Xuemin Shen
Heather Yu
John Buford
Mursalin Akon
Editors

Handbook of Peer-to-Peer Networking

Springer

Handbook of Peer-to-Peer Networking

Xuemin Shen · Heather Yu · John Buford ·
Mursalin Akon
Editors

Handbook of Peer-to-Peer Networking

Springer

Editors
Xuemin Shen
Department of Electrical & Computer Engineering
University of Waterloo
200 University Avenue W.
Waterloo, ON N2L 3G1
Canada
xshen@bbcr.uwaterloo.ca

John Buford
Avaya Labs Research
233 Mount Airy Road
Basking Ridge, NJ 07920
USA
buford@samrg.org

Heather Yu
Huawei Technologies USA
400 Someset Corp Blvd.
Bridgewater, NJ 08807
USA
heathery@ieee.org

Mursalin Akon
Department of Electrical & Computer Engineering
University of Waterloo
200 University Avenue W.
Waterloo, ON N2L 3G1
Canada
mursalin@ieee.org

ISBN 978-0-387-09750-3 e-ISBN 978-0-387-09751-0
DOI 10.1007/978-0-387-09751-0
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2009937578

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

For our parents.
-Xuemin Shen, Heather Yu,
John F. Buford and Mursalin Akon

Preface

Peer-to-peer networking is a disruptive technology for large scale distributed applications that has recently gained wide interest due to the successes of peer-to-peer (P2P) content sharing, media streaming, and telephony applications. There are a large range of other applications under development or being proposed. The underlying architectures share features such as decentralizaton, sharing of end system resources, autonomy, virtualization, and self-organization. These features constitute the P2P paradigm. This handbook broadly addresses a large cross-section of current research and state-of-the-art reports on the nature of this paradigm from a large number of experts in the field.

Several trends in information and network technology such as increased performance and deployment of broadband networking, wireless networking, and mobile devices are synergistic with and reinforcing the capabilities of the P2P paradigm. There is general expectation in the technical community that P2P networking will continue to be an important tool for networked applications and impact the evolution of the Internet. A large amount of research activity has resulted in a relatively short time, and a growing community of researchers has developed.

The *Handbook of Peer-to-Peer Networking* is dedicated to discussions on P2P networks and their applications. This is a comprehensive book on P2P computing. It addresses all issues currently developed as well as under development including P2P architectures, search and queries, incentive mechanisms, multimedia streaming, service oriented architectures, collaboration to share non-storage resources, mobile P2P, theory and analysis, and P2P databases. In addition, it covers rather practical perspectives such as traffic characteristics and trends of P2P applications, the E-business model in P2P applications, and software characteristics. Finally, the book contains chapters on emerging P2P concepts and applications.

The goal of this handbook is to provide an exhaustive view of the state-of-the-art of the P2P networking field. In organizing the book, the following objectives were followed:

- Provide an overview of the fundamentals of P2P networks
- Describe the current practice in P2P applications and industries

- Comprehensively cover the areas of interest
- Give the most recent perspective from the P2P research community

This book is written for researchers, professionals, and computer science and engineering students at the advanced undergraduate level and higher who are familiar with networking and network protocol concepts and basic ideas about algorithms. For the more advanced parts of the book, the reader should have general familiarity with Internet protocols such as TCP and IP routing. For some sections of the book such as discussions of mobility or multicasting, familiarity with mobility in IP and IP multicasting will be helpful but is not required.

The *Handbook of Peer-to-Peer Networking* is intended to provide readers with a comprehensive reference for the most current developments in the field. It offers broad coverage P2P networking with fifty chapters written by international experts. In addition, we hope the book becomes an important reference to those who are active in the field. The fifty chapters of the *Handbook of Peer-to-Peer Networking* are organized into the following sections:

- **An Introduction to Peer-to-Peer Networking** – This section contains background chapters accessible to the general reader, and covers the basic models, applications, and usage.
- **Unstructured P2P Overlay Architectures** – The majority of deployed P2P applications use unstructured overlays. These chapters cover the organizational principles and discuss a variety of examples, including overlays using social and semantic relationships.
- **Structured P2P Overlay Architectures** – Structured overlays have been a widely studied alternative approach, with over fifty different designs having been proposed. These chapters describe some of the leading models and their algorithms, as well as dynamics, bootstrapping, formalization, and stabilization.
- **Search and Query Processing** – Search is perhaps the most fundamental service in an overlay. A wide range of techniques are discussed, from basic keyword search to semantic search, and database query processing and indexing mechanisms applied to the P2P architecture.
- **Incentive Mechanisms** – In practice, peers are not altruistic, so techniques to ensure fair and mutual resource sharing have been proposed. An important category is incentive mechanisms which allows peers to participate proportional to their resource contribution to other peers.
- **Trust, Anonymity, and Privacy** – In most P2P overlays, peers are in autonomous security domains, and have no a priori basis for safe cooperation. Peer reputation management is an important category of enabling trust, and is discussed here in three chapters. In addition, work on anonymity in P2P networks and private P2P networks are presented in two chapters.
- **Broadcast and Multicast Services** – Multicast services are some of the earliest use of overlays and are often described as application layer or end system multicast to distinguish them from network layer multicast. This section has a rich coverage of work in both multicast and broadcast mechanisms, include gossip-based approaches and hybrid designs.

- **Multimedia Content Delivery** – The inherent scalability of the P2P paradigm makes it an attractive choice for large scale media streaming. The first chapter in this section examines key business models for P2P content delivery. The remaining chapters address recent work in IPTV and Video-on-Demand in P2P networks.
- **Mobile P2P** – When peers roam or are operating in ad hoc networks, the efficiency and stability of the overlay is effected. The growing importance of mobile and ad hoc networking for many applications has attracted research on the use of P2P overlays in MANETs, which is discussed in three chapters in this section.
- **Fault Tolerance in P2P Networks** – Uneven workloads and instability due to churn are some of the practical issues in operating large-scale P2P systems. Various load balancing techniques have been studied for adapting to uneven workloads, and are surveyed in one chapter in this section. Stabilization of the overlay and automatically correcting network partitions are topics of two other chapters.
- **Measurement and P2P Traffic Characteristics** – Another topic of great practical interest is the network traffic of P2P applications, which has become the dominant flow on the Internet. This issue affects both network operators, as discussed in one chapter, and the design of the overlay, as described in three other chapters.
- **Advanced P2P Computing and Networking** – Four special topics represent areas of P2P research that will gain more attention in future years are discussed in this section: formal models of P2P software, the use of web services in the P2P architecture; support for publish-subscribe and event-driven processing; and enabling collaborative applications in a P2P overlay.

Acknowledgements

We would like to thank those individuals who reviewed the original proposal and made important comments about structure, topics, and emphasis. Special thanks to Mohammad Islam who always lent an extra hand.

Thanks are due to the staff at Springer Science+Business Media: Melissa Fearon the senior editor for this book; others on the production side ...

Finally, we thank our families for their support and understanding while we worked on this book.

Waterloo, Canada	Xuemin (Sherman) Shen
Princeton, USA	Heather Yu
Princeton, USA	John F. Buford
Waterloo, Canada	Mursalin Akon
	March 2009

Contents

Part I Introduction to Peer-to-Peer Networking

Part II Unstructured P2P Overlay Architectures

Part III Structured P2P Overlay Architectures

Part IV Search and Query Processing

Part V Incentive Mechanisms

Part VI Trust, Anonymity, and Privacy

Part VII Broadcast and Multicast Services

Part VIII Multimedia Content Delivery

Part IX Mobile P2P

Part X Fault Tolerance in P2P Networks

Part XI Measurement and P2P Traffic Characteristics

Part XII Advanced P2P Computing and Networking

List of Contributors

Adam Wierzbicki
Polish Japanese Institute of Information Technology, Warsaw, Poland,
e-mail: `adamw@pjwstk.edu.pl`

Ahmed Serhrouchni
Institut Telecom, TELECOM-ParisTech, 46 rue Barrault, 75013 Paris, France,
e-mail: `Ahmed.Serhrouchni@telecom-paristech.fr`

Ajit Singh
Department of ECE, University of Waterloo, ON, Canada, N2L 3G1,
e-mail: `asingh@uwaterloo.ca`

Akrivi Vlachou
Department of Informatics, Athens University of Economics and Business (AUEB),
10434 Athens, Greece, e-mail: `avlachou@aueb.gr`

Alexander Löser
Technische Universität Berlin, Einsteinufer 17, 10587 Berlin, Germany,
e-mail: `alexander.loeser@tu-berlin.de`

Ali Ghodsi
Swedish Institute of Computer Science (SICS), Box 1263, 164 29 Kista, Sweden,
e-mail: `ali@sics.se`

Andrea Glorioso
European Commission DG Information Society and Media, Avenue de Beaulieu 33
B-1049, Bruxelles, e-mail: `Andrea.Glorioso@ec.europa.eu`

Andreas J. Kassler
Department of Computer Science, Karlstads University, Universitetsgatan 2,
SE-651 88, Karlstad, Sweden, e-mail: `Andreas.Kassler@kau.se`

Anwitaman Datta
Nanyang Technological University, Nanyang Avenue, Singapore,
e-mail: anwitaman@ntu.edu.sg

Bo Xu
Department of Computer Science, University of Illinois at Chicago, USA

C.-F. Michael Chan
Stanford University, Stanford, CA, USA, e-mail: mcfchan@stanford.edu

Carla-Fabiana Chiasserini
Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy, e-mail: chiasserini@polito.it

Christoph Tempich
DETECON international GmbH, Oberkasselerstrasse 2, 53227 Bonn, Germany,
e-mail: christoph@tempich.com

Christos Doulkeridis
Department of Informatics, Athens University of Economics and Business (AUEB), 10434 Athens, Greece, e-mail: cdoulk@aueb.gr

Claudio Casetti
Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy, e-mail: casetti@polito.it

Cyrus Shahabi
University of Southern California, Department of Computer Science, Los Angeles, CA 90089-0781, USA, e-mail: shahabi@usc.edu

Daniel A. G. Manzato
University of Campinas, Institute of Computing, Av. Albert Einstein, 1251, P.O. Box 6176, 13084-971, Campinas, SP, Brazil,
e-mail: dmanzato@ic.unicamp.br

Daniel Stutzbach
Stutzbach Enterprises, Dallas, Texas, USA,
e-mail: daniel@stutzbachenterprises.com

Daniela Saladino
Department of Information Engineering, University of Modena and Reggio Emilia, Italy, e-mail: daniela.saladino@unimore.it

Danny H. K. Tsang
Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, e-mail: eetsang@ece.ust.hk

Dingyi Han
Apex Data & Knowledge Management Lab, Department of Computer Science & Engineering, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai, 200240, China, e-mail: handy@apex.sjtu.edu.cn

Dirk De Grooff
Centre for UX Research / IBBT, K.U.Leuven, Parkstraat 45 Bus 3605 - 3000
Leuven - Belgium, e-mail: dirk.degrooff@soc.kuleuven.be

Duncan Russell
School of Computing, University of Leeds, Leeds, LS2 9JT, UK,
e-mail: duncanr@comp.leeds.ac.uk

Fabien mathieu
Orange Labs, 38–40 rue du général Leclerc, 92794 Issy-les-Moulineaux, France,
e-mail: fabien.mathieu@orange-ftgroup.com

Farnoush Banaei-Kashani
University of Southern California, Department of Computer Science, Los Angeles,
CA 90089-0781, USA, e-mail: banaeika@usc.edu

Félix Gómez Mármol
Departamento de Ingeniería de la Información y las Comunicaciones, University
of Murcia, Facultad de Informática, Campus de Espinardo, 30.071 Murcia, Spain,
e-mail: felixgm@um.es

Fotios C. Harmantzis
Stevens Institute of Technology, Hoboken, NJ 07030, USA,
e-mail: Fotios.Harmantzis@stevens.edu

Gerhard Hasslinger
T-Systems, Deutsche-Telekom-Allee 7, D-64295 Darmstadt, Germany,
e-mail: gerhard.hasslinger@telekom.de

Giancarlo Ruffo
Department of Computer Science, University of Torino, Corso Svizzera, 185,
Torino, Italy, e-mail: giancarlo.ruffo@unito.it

Gregorio Martínez Pérez
Departamento de Ingeniería de la Información y las Comunicaciones, University
of Murcia, Facultad de Informática, Campus de Espinardo, 30.071 Murcia, Spain,
e-mail: gregorio@um.es

Guillaume Raschia
LINA, 2 rue de la Houssiniere 44322 Nantes, France,
e-mail: guillaume.raschia@univ-nantes.fr

Haiying Shen
University of Arkansas, Fayetteville Avenue, AR, USA,
e-mail: hshen@uark.edu

Heather Yu
Huawei Technologies, Bridgewater, NJ, USA, e-mail: heathery@ieee.org

Ibrahim Korpeoglu
Department of Computer Engineering, Bilkent University, 06800 Ankara, Turkey,
e-mail: korpe@cs.bilkent.edu.tr

Jian-bin Han
Lehigh University, Department of Computer Science and Engineering, 19 Memorial Drive West, Lehigh University, Bethlehem, PA 18015,
e-mail: jih206@lehigh.edu

Jie Xu
School of Computing, University of Leeds, Leeds, LS2 9JT, UK,
e-mail: jxu@comp.leeds.ac.uk

João Leitão
INESC-ID / IST, Lisbon, Portugal, e-mail: jleitao@gsd.inesc-id.pt

João V. P. Gomes
Nokia Siemens Networks Portugal, S. A., Rua Irmãos Siemens, 1, Alfragide, 2720-093 Amadora, Portugal, e-mail: jgomes@penhas.di.ubi.pt

John F. Buford
Avaya Labs Research, Lincroft, NJ, USA, e-mail: buford@samrg.org

Jorn De Boever
Centre for UX Research / IBBT, K.U.Leuven, Parkstraat 45 Bus 3605 - 3000 Leuven, Belgium, e-mail: jorn.deboever@soc.kuleuven.be

José Pereira
University of Minho, Braga, Portugal, e-mail: jop@di.uminho.pt

Josemaria Malgosa-Sanahuja
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain,
e-mail: josem.malgosa@upct.es

Juan Carlos Sanchez-Aarnoutse
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain,
e-mail: juanc.sanchez@upct.es

Juan Pedro Muñoz-Gea
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain,
e-mail: juanp.gea@upct.es

Kin-WahKwong
Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA, e-mail: kkw@seas.upenn.edu

Kjetil Nørvåg
Department of Computer and Information Science, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway,
e-mail: Kjetil.Norvag@idi.ntnu.no

Kolja Eger
Siemens AG, Corporate Technology Information and Communications, Otto-Hahn-Ring 6, 81739 Munich, Germany, e-mail: kolja.eger@siemens.com

Kostas G. Anagnostakis
Institute for Infocomm Research, Heng Mui Keng Terrace, 119613, Singapore,
e-mail: kostas@i2r.a-star.edu.sg

Krishna Dhara
Avaya Labs Research, Lincroft, NJ, USA

Krzysztof Rzadca
Nanyang Technological University, Nanyang Avenue, Singapore,
e-mail: krz@pjwstk.edu.pl

Loubna Mekouar
Loubna Mekouar, University of Waterloo, Waterloo, Canada,
e-mail: lmekouar@bbcr.uwaterloo.ca

Lu Liu
School of Computing, University of Leeds, Leeds, LS2 9JT, UK,
e-mail: luliu@comp.leeds.ac.uk

Lu Yan
School of Computer Science, University of Hertfordshire, Hatfield, Hertfordshire AL10 9AB, UK, e-mail: lu.yan@ieee.org

Luís Rodrigues
INESC-ID / IST, Lisbon, Portugal, e-mail: ler@ist.utl.pt

Łukasz Żaczek
Polish Japanese Institute of Information Technology, Warsaw, Poland,
e-mail: lzaczek@pjwstk.edu.pl

Manaf Zghaibeh
Stevens Institute of Technology, Hoboken, NJ 07030, USA,
e-mail: Manaf.Zghaibeh@stevens.edu

Manuela Pereira
Institute of Telecommunications - Networks and Multimedia Group, Department of Computer Science, University of Beira Interior, Rua Marquês de Ávila e Bolama, 6201-001 Covilhã, Portugal, e-mail: mpereira@di.ubi.pt

Marcel C. Castro Department of Computer Science, Karlstads University, Universitetsgatan 2, SE-651 88, Karlstad, Sweden,
e-mail: Marcel.Cavalcanti@kau.se · Marguerite Fayçal
Institut Telecom, TELECOM-ParisTech, 46 rue Barrault, 75013 Paris, France,
e-mail: Marguerite.Faycal@telecom-paristech.fr

Maria Luisa Merani
Department of Information Engineering, University of Modena and Reggio Emilia, Modena and Reggio Emilia, Italy,
e-mail: marialuisa.merani@unimore.it

Mario Kolberg
University of Stirling, Stirling, UK, e-mail: mko@cs.stir.ac.uk

Mário M. Freire
Institute of Telecommunications - Networks and Multimedia Group, Department of Computer Science, University of Beira Interior, Rua Marquês de Ávila e Bolama, 6201-001 Covilhã, Portugal, e-mail: mario@di.ubi.pt

Matthias Wählisch
HAW Hamburg, Dept. Informatik, Berliner Tor 7, 20099 Hamburg, Germany
Freie Universität Berlin, Institut für Informatik, Takustr. 9, 14195 Berlin, Germany,
e-mail: waehlisch@ieee.org

Michael Rogers
University College London, London WC1E 6BT, UK,
e-mail: m.rogers@cs.ucl.ac.uk

Michalis Vazirgiannis
Department of Informatics, Athens University of Economics and Business (AUEB), 10434 Athens, Greece, e-mail: mvazirg@aueb.gr

Mooi-Choo Chuah
Lehigh University, Department of Computer Science and Engineering, 19 Memorial Drive West, Lehigh University, Bethlehem, PA 18015, USA,
e-mail: chuah@cse.lehigh.edu

Mursalin Akon
Department of ECE, University of Waterloo, ON, Canada, N2L 3G1,
e-mail: mmakon@uwaterloo.ca

Nelson L. S. da Fonseca
University of Campinas, Institute of Computing, Av. Albert Einstein, 1251, P.O. Box 6176, 13084-971, Campinas, SP, Brazil,
e-mail: nfonseca@ic.unicamp.br

Nick Antonopoulos
Department of Computing, University of Surrey, Guildford, Surrey, GU2 7XH, UK, e-mail: n.antonopoulos@surrey.ac.uk

Noureddine Mouaddib
LINA, 2 rue de la Houssiniere, 44322 Nantes, France, e-mail: Noureddine.Mouaddib@univ-nantes.fr

Nuno A. Carvalho
University of Minho, Braga, Portugal,
e-mail: nuno.carvalho@di.uminho.pt

Ouri Wolfson
Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA

Patrick Valduriez
INRIA, 2 rue de la Houssiniere 44322, Nantes, France,
e-mail: Patrick.Valduriez@inria.fr

Paulo P. Monteiro
Nokia Siemens Networks Portugal, S. A., Rua Irmãos Siemens, 1, Alfragide, 2720-093 Amadora, Portugal, e-mail: paulo.1.monteiro@nsn.com

Pedro R. M. Inácio
Nokia Siemens Networks Portugal, S. A., Rua Irmãos Siemens, 1, Alfragide, 2720-093 Amadora, Portugal,
e-mail: pedro.inacio@penhas.di.ubi.pt

Peter Kersch
High Speed Networks Laboratory, Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary, e-mail: kersch@tmit.bme.hu

Pilar Manzanares-Lopez
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain,
e-mail: pilar.manzanares@upct.es

Rabab Hayek
LINA, 2 rue de la Houssiniere 44322, Nantes, France,
e-mail: rabab.hayek@univ-nantes.fr

Raouf Boutaba
University of Waterloo, Ontario, Canada,
e-mail: rboutaba@bbcr.uwaterloo.ca

Reaz Ahmed
Bangladesh University of Engineering and Technlogy, Dhaka, Bangladesh,
e-mail: reaz@cse.buet.ac.bd

Reza Rejaie
University of Oregon, Eugene, Oregon, e-mail: reza@cs.uoregon.edu

Robert Szabo
High Speed Networks Laboratory, Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Hungary,
e-mail: szabo@tmit.bme.hu

Rui Oliveira
University of Minho, Braga, Portugal, e-mail: rco@di.uminho.pt

S.-H. Gary Chan
Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, e-mail: gchan@cse.ust.hk

Saleem Bhatti
University of St Andrews, Fife KY16 9SS, UK,
e-mail: saleem@cs.st-andrews.ac.uk

Seif Haridi
Royal Institute of Technology (KTH), Electrum 229, 164 40 Kista, Sweden,
e-mail: haridi@kth.se

Steffen Staab
University of Koblenz-Landau, Universitätsstrasse 1, 56070 Koblenz, Germany,
e-mail: staab@uni-koblenz.de

Tallat M. Shafaat
Royal Institute of Technology (KTH), Electrum 229, 164 40 Kista, Sweden,
e-mail: tallat@kth.se

Thomas C. Schmidt
HAW Hamburg, Dept. Informatik, Berliner Tor 7, 20099 Hamburg, Germany,
e-mail: t.schmidt@ieee.org

Towhidul Islam
Department of ECE, University of Waterloo, ON, Canada, N2L 3G1,
e-mail: mtislam@uwaterloo.ca

Ulrich Killat
Hamburg University of Technology (TUHH), Institute of Communication Networks, 21071 Hamburg, Germany, e-mail: killat@tuhh.de

Ugo Pagallo
Department of Law, University of Torino, Via Sant´Ottavio, Torino, Italy,
e-mail: ugo.pagallo@unito.it

Xiaotao Wu
Avaya Labs Research, Lincroft, NJ, USA

Xuemin (Sherman) Shen
Department of ECE, University of Waterloo, ON, Canada, N2L 3G1,
e-mail: xshen@bbcr.uwaterloo.ca

Yan Luo
Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA

Yang Guo
Corporate Research, Thomson, Princeton, NJ, USA

Yingwu Zhu
Seattle University, Seattle, WA 98122, USA, e-mail: zhuy@seattleu.edu

Yong Yu
Apex Data & Knowledge Management Lab, Department of Computer Science & Engineering, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai, 200240, China, e-mail: yyu@apex.sjtu.edu.cn

Youssef Iraqi
Dhofar University, Salalah, Oman, e-mail: y_iraqi@du.edu.om

Zongyang Luo
School of Computing, University of Leeds, Leeds, LS2 9JT, UK,
e-mail: scszluo@leeds.ac.uk

Zoran Despotovic
DOCOMO Communications Laboratories Europe GmbH, Munich, Germany,
e-mail: despotovic@docomolab-euro.com

Part I
Introduction to Peer-to-Peer Networking

Peer-to-Peer Networking and Applications: Synopsis and Research Directions

John F. Buford and Heather Yu

Abstract Peer-to-peer computing and networking are important developments for large-scale distributed systems design and the evolution of Internet architecture. Widely used applications have demonstrated their feasibility and economic potential for services involving millions of users. A great deal of research has followed to formalize and improve on the empirical results. This introductory chapter surveys the key results of the field, introduces terminology, and identifies open issues which are likely to be important research directions.

1 Introduction

1.1 Significance and Emergence

Several important desktop computing applications have emerged in recent years that use an Internet-scale decentralized architecture to simultaneously connect millions of users to share content, form social groups and communicate with their contacts. These applications are classified as peer-to-peer because of the elimination of servers to mediate between end systems on which the applications run, and their network behavior is described as an overlay network because the peer protocols form a virtualized network over the physical network.

While peer-to-peer (P2P) applications have had a rapid ascent and wide impact, in the future P2P overlays are likely to enable important new applications following from these technology trends:

John F. Buford
Avaya Labs Research, Basking Ridge, NJ 07920, USA, e-mail: `buford@samrg.org`

Heather Yu
Huawei Technologies, Bridgewater, NJ, USA, e-mail: `heathery@ieee.org`

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_1, © Springer Science+Business Media, LLC 2010

- Continued improvements in the fidelity of the consumer entertainment experience and network and computing capacity of the associated entertainment devices
- The development of dense and ubiquitous sensing grids with real-time data collection of all types of phenomena
- The wide deployment of broadband wireless networks (WiMax, 802.11n, UWB, LTE)
- The proliferation of mobile smart phones and other broadband-enabled mobile devices
- The use of personal networks, body-area networks, and vehicle networks, to connect both real-time sensors and embedded computing devices.

The wide adoption of these technologies will enable high-fidelity and pervasive information collection, content publishing and distribution, and sharing of environmental and personal real-time sensed data and information on a global scale. The benefits of this include increased awareness of one's personal environment, more precise context-awareness in interactions with others, and enhanced situation-awareness for applications ranging from immersive entertainment and recreation, environment management, homeland security, and disaster recovery. Peer-to-peer overlays are an important component of this future vision, due to their high scalability, flexibility for different types of applications, and low barrier of entry. The evolution of contemporary P2P overlays to enable this future vision is an important research direction.

The use of application layer protocols to form overlays to deliver Internet services has a long history (Table 1). However until relatively recently, these types of overlays used specifically designed protocols, and were used to interconnect infrastructure servers rather than end systems. In addition, the address space of the overlay was typically not virtualized, and dealing with churn was not a primary design point. Nevertheless, such service overlays continue to be an important part of the Internet architecture [1–3], and there is growing interest in using the end-to-end and resource virtualization capabilities of overlays in the evolution of the Internet architecture. Example research efforts in this direction include SpoVNet [4] and SATO [5].

Table 1 Specialized overlay networks for internet services

Type	Example	First use or definition
Email	SMTP	1970s
Internet news	NNTP	1986
Multicast	MBone	1992
Web caching	Internet cache protocol	1995
Content delivery network	Akamai	1999
Anonymous communication	FreeNet	1999
Application layer multicast	Narada	2000
Routing	RON	2001

The well-known popularization of P2P file sharing systems beginning with the hybrid Napster and followed by other content dowloading P2P systems such as Gnutella, FastTrack, KaZaa, and BitTorrent invigorated the interest of the research community to develop solutions to the perceived deficiences of these systems. The subsequent availability of P2PTV and VoP2P applications discussed in the next section were further catalysts for research in real-time media streaming over P2P overlay networks.

Notable underpinnings of the research in improving P2P overlays was the early work on distributed hash tables by Devine [6] as well as Litwin et al. [7, 8]. Plaxton, Rajaraman, and Richa (PRR) [9] presented the first algorithms for distributed object location and routing, using a suffix forwarding scheme. PRR was the basis for subsequent influential designs such as Tapestry and Pastry. Karger, et al. [10] formalized consistent hashing which is the basis for many DHT designs.

1.2 Key Applications

The first widely used file sharing system, Napster, featured a hybrid architecture in which the directory was stored on a server, but peers directly transferred files between them. Napster became the first legal test case for file sharing of licensed content, and was subsequently forced to change to protect such content. A number of peer-to-peer file sharing systems were developed (Table 2) to avoid the legal challenges faced by Napster. The majority of these second-generation file sharing systems were based on unstructured overlays. While these systems had no mechanisms for protecting the rights of content owners, in some cases the P2P application developers obtained revenue by either selling their applications or by embedding

Table 2 Example file sharing applications

Client application(s)	Protocol	Description
KaZaA grokster imesh	FastTrack	Proprietary unstructured overlay with encrypted protocol, high capacity peers become superpeers; features connection shuffling
Limewire	Gnutella	Superpeer unstructured overlay with flooding query propagation
eDonkey	Overnet	Structured overlay based on Kademlia
eMule	Kad	Structured overlay based on Kademlia
BitTorrent	BitTorrent	An unstructured overlay used for distributing large files in pieces using mutual distribution of the pieces between a set of peers called a swarm. Uses a server to store the torrent and another server called a tracker to identify the swarm members

spyware in the clients. Recent research studies of P2P file sharing systems include [11–16].

Several new ventures such as QTrax, SpiralFrog, and TurnItUp have proposed incorporating DRM in to the file sharing applications or ad-based revenue models in which advertisements are delivered during media playback.

The founders of KaZaA subsequently launched the first widely used voice-over P2P (VoP2P) application, Skype. Currently Skype connects around 15 M concurrent users and provides a variety of services including P2P voice and video calls, voice calls to PSTN endpoints, presence, and instant messaging. Like KaZaA, the Skype protocol is encrypted and the definition of the protocol has not been released. Some studies have shown that Skype uses a superpeer model, and the superpeers support NAT traversal for connecting peers behind NATs. In addition, superpeers also act as media relays. Recent research studies of Skype include [17–21, 101].

In contrast to file sharing systems which exhibit the free rider behavior, in P2P telephony users are motivated to stay connected both to be able to receive calls and view the current status of their buddies. Long application lifetimes mean a low churn rate, which makes the operation of the overlay more stable. Experimental studies of Skype have shown a significantly higher node lifetime compared to P2P file sharing sytems.

The distribution of streaming video referred to as P2PTV has also become an important application of P2P. Various models are used, including torrent-style distribution, application layer multicasting, and hybrid CDNs (content delivery networks). Example PPTV applications include Babelgum, Joost, PPLive, PPStream, SopCast, TVants, TVUPlayer, Veoh TV, and Zattoo. P2PTV is expected to play an important role in future IPTV deployments. A summary of P2PTV related research is discussed later in this chapter.

1.3 Definition and Properties of P2P Systems

Peer-to-peer systems have been defined in many papers. Here are two definitions that cover the concepts of resource sharing, self-organization, decentralization, and interconnection:

> “A distributed network architecture may be called a peer-to-peer network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers). These shared resources are necessary to provide the Service and content offered by the network (e.g. file sharing or shared workspaces for collaboration). They are accessible by other peers.”[22]
>
> “Peer-to-peer systems are distributed systems consisting of interconnected nodes able to self-organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority.” [23]

We have also defined an overlay network [26]:

> "An application layer virtual or logical network in which end points are addressable and that provides connectivity, routing, and messaging between end points. Overlay networks are frequently used as a substrate for deploying new network services, or for providing a routing topology not available from the underlying physical network. Many peer-to-peer systems are overlay networks that run on top of the Internet."

The following properties are characteristics found in most P2P systems.

Resource sharing each peer contributes system resources to the operation of the P2P system. Ideally this resource sharing is proportional to the peer's use of the P2P system, but many systems suffer from the free rider problem.

Networked all nodes are interconnected with other nodes in the P2P system, and the full set of nodes are members of a connected graph. When the graph is no longer connected, the overlay is said to be partitioned.

Decentralization the behavior of the P2P system is determined by the collective actions of peer nodes, and there is no central control point. Some systems however secure the P2P system using a central login server. The ability to manage the overlay [24] and monetize its operation may require centralized elements.

Symmetry nodes assume equal roles in the operation of the P2P system. In many designs this property is relaxed by the use of special peer roles such as super peers or relay peers.

Autonomy participation of the peer in the P2P system is determined locally, and there is no single administrative context for the P2P system.

Self-organization the organization of the P2P system increases over time using local knowledge and local operations at each peer, and no peer dominates the system. Biskupski, Dowling, and Sacha [25] argue that existing P2P systems do not exhibit most of the properties of self-organization.

Scalable This is a pre-requisite of operating P2P systems with millions of simultaneous nodes, and means that the resources used at each peer exhibit a growth rate as a function of overlay size that is less than linear. It also means that the response time doesn't grow more than linearly as a function of overlay size.

Stability Within a maximum churn rate, the P2P system should be stable, i.e., it should maintain its connected graph and be able to route deterministically within a practical hop-count bounds.

1.4 Business Models

P2P file sharing applications have been monetized by their operators by sales of the P2P client software or by embedding spyware into the application. Content licensing has not been as successful to date. The leading VoP2P application, Skype, provides basic P2P telephony for free, and receives revenue for add-on services such as voice mail or peer-to-PSTN calls. The primary model for P2PTV operators is similar to existing cable and broadcast TV – embedded advertising with measurable viewership.

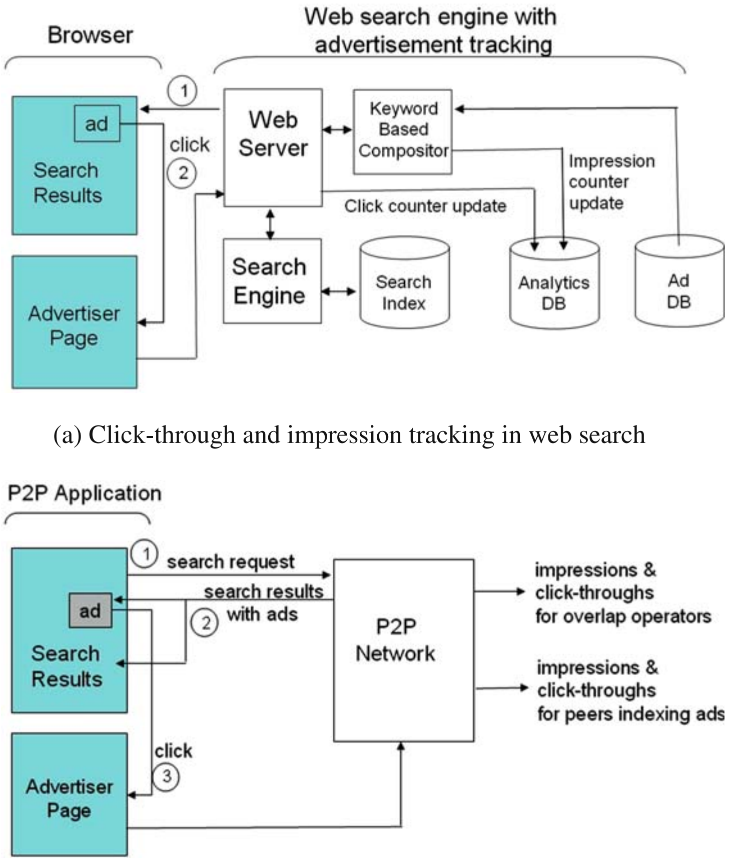
The advertising mechanisms used in web search (impressions, cost-per-click, and placements) are more difficult to implement in P2P applications since verification can not rely on a centralized collection point. Figure 1a shows a simplified model for impression and click counting in web search. The search results are produced by the search engine, and in parallel the search keywords and other criteria are used to select ads which will be composed with the search results, resulting in an impression for each displayed ad. Each time an impression occurs, a counter is updated in the advertisement analytics database. If the user clicks on the ad, the embedded url leads to the advertiser's web page. A click counter must be updated in the advertisement analytics databse. Two ways to obtain this are to indirectly forward the ad url via a specific search engine web server (e.g., www.searchengine.com?advertiser.com), or to embed a script in the advertiser's web page which invokes the search engine's url when loaded. The advertiser can also use third party services to track web site hits with analytics reports produced by the search vendor.

In the P2P case, assume ads are selected for display on the P2P application user interface by association with searches initiated through the application user interface. (Of course, the P2P application developer could sell banner ad space on the P2P applications that would point to advertiser's web sites, but these would not be specifically targeted according to the user's activities or application use). Figure 1b shows the basic flow. A search request is propagated through the P2P network, returning one or more search results from various peers. Ads which match the search criteria are returned with the search results. There are several ways this could be done, using the P2P network or using a separate index, and monetization could be obtained by both the overlay operator and the operators of the peers that return the search results. For this discussion, serving advertisements through peers raises the question of how to validate impression counts maintained at the peers, as well as click-through counts at the peer application.

1.5 Technology Drivers

The P2P value proposition [26] "…for the user is to exchange excess computational, storage, and network resources for something else of value to the user, such as access to other resources, services, content, or participation in a social network." The rapid gains in computer capacity and wide adoption of broadband access have thus fueled the growth of P2P applications. As these trends are expected to continue, we raise the following questions:

– Given the limited ability of search engines to index all of the web, can P2P search compete with or augment web search, and at what scale and cost?



(a) Click-through and impression tracking in web search

(b) Click-through and impression tracking in P2P networks

Fig. 1 Click through and impression tracking

- Currently the largest online peer populations are in the range of 10–20 M. Can P2P networks support continued growth to say 10x or 100x peer population increase? What are the limits?
- Given the emergence of broadband wireless and the likely domination of peer population by mobile wireless devices versus fixed desktop devices, how will the stability and operation of P2P networks be effected?
- Will HDTV and high-definition video lead to new P2P applications?
- Will deployment of large-scale sensor grids create new applications for P2P networks, and what are appropriate architectures for interconnecting such networks in a global overlay?

1.6 Structure of the Chapter

The remainder of the chapter is organized as follows. First we survey overlay design, and provide a taxonomy for understanding the many different overlays that have been proposed. This survey includes unstructured, structured, hierarchical, service, semantic, and sensor overlays. The next section summarizes results on overlay dynamics, including mobility and overlays for MANETs, and variable hop overlays. Search, overlay multicast, content delivery, and security are summarized in subsequent sections.

2 Overlay Basics

2.1 Classification and Taxonomy

The many different designs for P2P networks have led to various proposals for classification. For example, file sharing systems have been divided in to generations. First generation are hybrid designs that combine servers with P2P routing, and second generation are decentralized architectures. Anonymized P2P systems such as Freenet and I2P are sometimes referred to as third generation. Categorization by generations has several shortcomings. It leaves out many other important dimensions and doesn't explain what subsequent generations are likely to provide. Further, systems of all three generations were in use at the same time.

Another common distinction is to divide P2P overlays into unstructured and structured types. Unstructured overlays are usually further distinguished by how search requests are propagated, distribution of node degree in the peer population, and by differences in link formation with neighbor peers. Structured overlays are differentiated according to a variety of dimensions such as:

- maximum number of hops for routing a request (e.g., multi-hop, one-hop, variable hop)
- routing algorithm (e.g., prefix, XOR, geometric distance, address space difference, semantic distance)
- node degree with size of overlay (e.g., constant degree, logarithmic degree)
- overlay geometry
- lookup type (iterative vs recursive, and serial vs parallel)

Beyond the unstructured and structured categories, we find other categories, such as hierarchical overlays, federated overlays, overlays for deploying network services called service overlays, overlays for sensor grids called sensor overlays, overlays which route queries by semantic relationships called semantic overlays, and overlays providing support for mobile nodes in IP and ad hoc networks. We describe these categories in more detail in the following sections. Figure 2 shows a classification tree for many of these categories of P2P overlays. Classifications for mobile-enabled overlays, services overlays, and sensor overlays will be discussed in later sections.

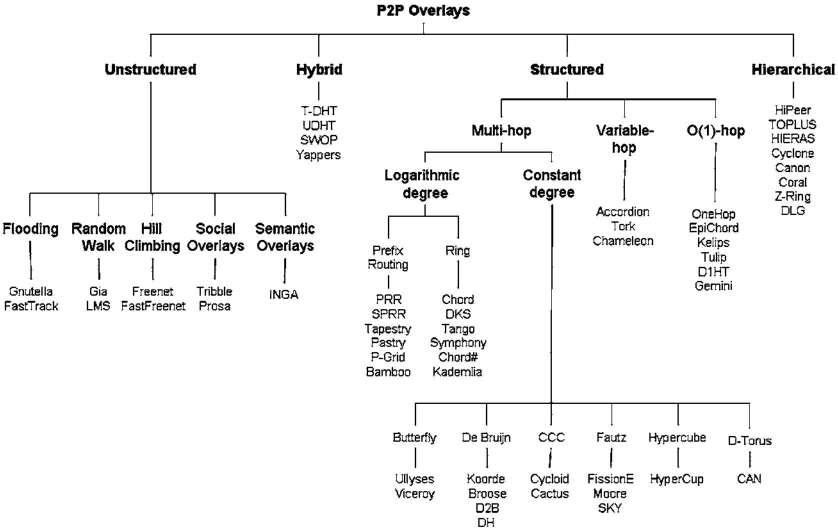


Fig. 2 Classification of P2P overlays

2.2 Unstructured Overlays

An unstructured overlay is "an overlay in which a node relies only on its adjacent nodes for delivery of messages to other nodes in the overlay. Example message propagation strategies are flooding and random walk" [26]. The graph structure formed by unstructured overlays can be compared to that of random graphs, scale-free or power law random graphs, graphs exhibiting small world phenomena, and other social networks. An important research focus for unstructured overlays has been the design of efficient search, including query propagation, object placement, and query processing. More details about search are discussed in a later section.

Another important research focus has been the optimal graph structure for unstructured overlays, and decentralized algorithms to form and maintain these graphs structures under changing peer and object populations. Influential unstructured overlay designs are listed in Table 3.

Table 3 Example unstructured overlays

Type	Design	Features	References
Hill climbing backtracking	Freenet	Routing using hill climbing with backing, and providing security, anonymity, and deniability	[27, 28]
Hill climbing backtracking	Fast freenet	Extension to Freenet in which objects stored at each peer are summarized and summaries are shared with neighbors	[29]
Hill climbing backtracking	Small world freenet	Freenet augmented with links emulating small world model	[30]
Flooding	Gnutella	Superpeers use flooding of requests on behalf of regular peers	[31, 32]
Random Walk	Gia	Uses techniques such as dynamic topology adaptation, active flow control, one-hop index replication, and biased random walk to improve performance	[33]
Flooding	FastTrack	Proprietary protocol with superpeer architecture that uses connection shuffling	[34]
Random Walk	LMS	Local minima search proactively replicates objects using consistent hashing of object identifiers to place objects at close node identifiers	[35]
Hybrid	SWOP	Structured overlay with additional cluster and long links to emulate small-world properties	[36]
Semantic routing	INGA	Semantic overlay in which each peer organizes a semantic index for its content, and queries are routed according to the associated topic, and evaluated using a semantic matching function	[37]
Preference directed queries	Tribler	Social-based overlay in which peers exchange preference lists to exploit social affinity between peers with similar preferences	[38]
Relevance directed queries	PROSA	P2P resource organization by social acquaintances manages peer links according to semantic strength of the relationship of the respective peer interests	[39]
Hybrid	Yappers	Combines local nodes in to small DHTs, and routes between DHTs using unstructured links	[40]

2.3 Structured Overlays

A structured overlay is: “an overlay in which nodes cooperatively maintain routing information about how to reach all nodes in the overlay” [26]. Compared to unstructured overlays, structured overlays provide a limit on the number of messages needed to find any object in the overlay. This is particularly important when searching for infrequently occurring or low popularity objects. In order to provide deterministic routing, peers are placed into a virtualized address space, the overlay is organized into a specific geometry, and a converging distance function over the combined object and node identifier space is defined for the routing forwarding algorithm.

Each peer has a local routing table which is used by the forwarding algorithm. The peer’s routing table is initialized when the peer joins the overlay, using a specified bootstrap procedure. Peers periodically exchange routing table changes as part of overlay maintenance. Overlay maintenance is discussed in a later section.

The majority of structured overlays use key-based routing in which “a set of keys is associated with addresses in the address space such that the nearest peer to an address stores the values for the associated keys, and the routing algorithm treats keys as addresses” [26]. A distributed hash table (DHT) is a structured overlay that uses key-based routing for put and get index operations and in which each peer is assigned to maintain a portion of the DHT index.

Because the address space is virtualized and peer addresses are typically randomly assigned, peers which are neighbors in the overlay can be distant in the underlying network. While this improves the fault tolerance of the overlay, it causes significant performance loss. Consequently, *topology-aware overlays* use measurements of proximity of peers in the underlying network to create neighbor peers in the overlay.

There has been some interest in the efficient support of broadcast in structured overlays. Broadcast can be used for group communication, blind search, and overlay configuration. Example approaches are defined in [41–44].

Table 4 summarizes many of the designs for structured overlays.

Table 4 Structured overlays by category

Type	Design	Features	References
Prefix routing	PRR	First DOLR algorithm, used suffix based routing	[45]
	SPRR	Added join/leave and maintenance to PRR	[46]
	Tapestry	Based on PRR with an added join/leave and maintenance mechanism	[47]
	Pastry	Prefix routing with last hop using neighbor table	[48]
	P-Grid	Prefix routing	[49]
	Bamboo	Variation of Pastry used in OpenDHT	[50]
	Z-Ring	Hierarchical address space with large base to reduce latency	[51]
Logarithmic degree	Chord	Logarithmic spaced links to neighbors around predecessor-successor ring, consistent hashing, unidirectional requests	[52]
	DKS(n,k,f)	Distributed k-ary search with routing region at each hop divided into k regions. $k=2$ similar to Chord	[53]
	Tango	Variation of DKS which reduces links to increasing scalability	[54]
	Chord #	Modification of Chord which replaces consistent hashing with key-order preserving indexing	[55]
	Symphony	Bi-directional routing with added long-links to shorten lookup hop count	[56]
	Kademlia	XOR distance function, parallel requests	[57]
Fixed degree	CAN	D-torus with cartesian coordinate system	[58]
	Viceroy	Butterfly	[59]
	Ulysses	Butterfly	[60]
	Cycloid	Cube connected cycle, prefix-style routing	[61]
	Cactus	2 trees combined with cube connected cycle	[62]
	Koorde	de Bruijn, based on Chord	[63]
	Broose	de Bruijn	[64]
	D2B	de Bruijn	[65]
	DH	de Bruijn	[66]
	Hi-Peer	Multi-ring de Bruijn	[67]
	Hypercup	Hypercube	[68]
	FissionE	Kautz graph	[69]
	Moore	Kautz graph	[70]
	SKY	Kautz graph	[71]
	DLG	A universal framework for building DHTs based on arbitrary constant-degree graphs, using distributed line graphs	[72]

(continued on following page)

Table 4 (continued) Structured overlays by category

Type	Design	Features	References
O(1)-hop	EpiChord	Iterative parallel lookup with opportunistic maintenance	[73]
	OneHop	Peers organized into slices and units, requests routed through slice and unit leaders, with active maintenance	[74]
	Kelips	Epidemic multicast protocol for overlay maintenance	[75]
	Tulip	2-hop overlay similar to Kelips which adds locality awareness	[76]
	Gemini	2-hop with high probability, combines suffix and prefix routing	[77]
	D1HT	Uses active maintenance algorithm EDRA where all join/leave events are forwarded to all other peers in logarithmic time	[78]
Variable Hop	Accordion	Recursive parallel lookup with bandwidth adaptive maintenance	[79]
	Chameleon	Hybrid of EpiChord and D1HT	[80]
	Tork	Hybrid of EpiChord and D1HT	[81]
Hierarchical	Multiple rings	Multiple overlays connect to a super-ring overlay, and use hierarchical routing to route requests between overlays	[82]
	TOPLUS	Peers organized into groups, each with own overlay; higher-level overlay is defined among the groups; intra-group routing used to route between overlays	[83]
	HIERAS	Overlay divided into several rings, with peers in low level rings selected according to locality	[84]
	Cyclone	Leaf overlays are connected in one horizontal overlay	[85]
	Z-Ring	Prefix routing with base = 4096, overlay organized in to groups for reduced maintenance	[86]
	Canon	At each successive level an overlay is formed which contains all peers in the subsumed overlays in the lower levels	[87]
	Coral	Peers are members of successive clusters of overlays, and lookups use distributed sloppy hash tables (DSHTs)	[88]

2.4 Hierarchical and Federated Overlays

A *hierarchical overlay* is an overlay architecture that uses multiple overlays arranged in a nested fashion, and the nested overlays are interconnected in a tree. A message to a peer in a different overlay is forwarded to the nearest common parent overlay in the hierarchy. When large scale distributed systems exhibit locality in their operation, hierarchical structure can increase overall performance. Hierarchical overlays exhibit hierarchical organization in addressing and routing. Different overlay regions in the hierarchy may use different routing algorithms. Important requirements for efficient operation of hierarchical overlays include avoiding bottlenecks and keeping the hierarchical structure balanced. Examples of hierarchical overlays include [82–88].

A *federated overlay* [89] is an overlay that is formed from a collection of independent overlays, each implemented by a separate administrative domain, and which may use different routing algorithms and addressing mechanisms in each domain. Each overlay is autonomous, and messaging operations between overlays require peering arrangements. Each domain manages the authentication, authorization and other management tasks for its overlay. Federation offers one mechanism by which overlay operators can offer specialized services to their customers while still providing the benefits of scale. An important requirement for federation is the trust relationship between each domain. Similar to the peering relationships between service providers for the Internet backbone, security breaches in one network have the potential to cascade to other overlays through the peering points. Thus the least secure overlay in the federation becomes a vulnerability for the remaining overlays.

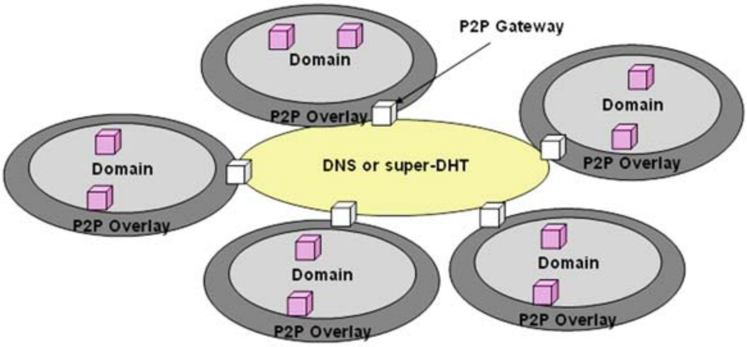


Fig. 3 Example of a federated overlay [89]

Individual overlays operate as usual according to the specific overlay algorithm. Pairs of overlays connect through gateway peers (Fig. 3). Gateway peers can discover other gateway peers for other overlays by different means such as a lookup in an interconnecting overlay or by DNS. A peer sending a message to a remote overlay first discovers a gateway peer and sends the message to it for forwarding. A multipart address scheme is used to distinguish objects and peers in separate overlays. Figure 4 shows an example message forwarding in a federated overlay using different types of overlays. Let's compare the routing in a federated overlay, each of size n peers, with that where all peers belong to a single overlay of size $N = \sum n$.

Case 1: A look-up within the same overlay will perform better than in the global overlay due to the smaller number of hops.

Case 2: A look-up crossing multiple federated overlays will perform worse as the number of overlays increases due to the overhead of address resolution and routing at each peering point. Increasing the number of direct peering relationships improves performance at the cost of additional complexity in forming and managing each peering point.

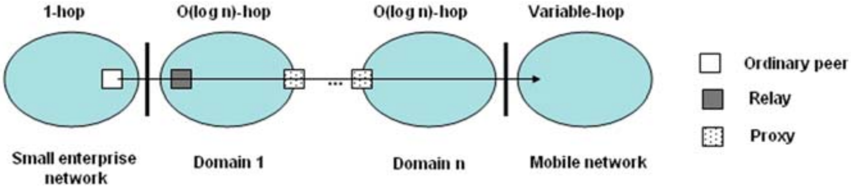
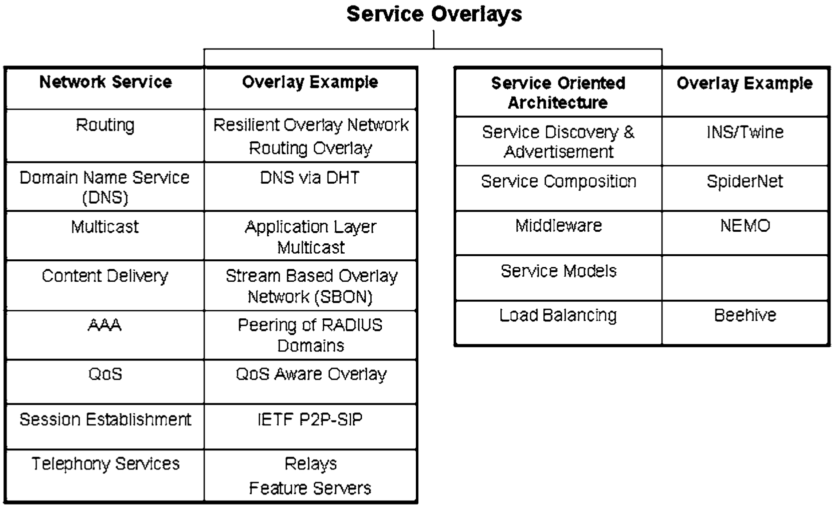


Fig. 4 Example multi-overlay messaging in a federated overlay

2.5 Service Overlays

As the Internet has grown, the need for new network services has also increased. However there is usually a long delay in developing and deploying any new service or extension to an existing service if it requires changes to network layer protocols, routers or other network infrastructure. This is due to the need to insure interoperability and avoid the introduction of new security vulnerabilities. To accelerate the deployment of new services and avoid changing the network infrastructure, many network services have been implemented as application layer protocols using end systems attached to the network. This includes multicasting, VoIP, and content delivery networks (CDNs). When an overlay is used as the basis for such application layer services, it is referred to as a *service overlay*.

In addition, service orientation is a new paradigm for architecting web applications and distributed enterprise systems. One of the shortcomings of P2P applications to date is that each application has had a dedicated overlay designed specifically for it. The idea of applying service orientation to P2P overlays is also referred as a service overlay. From this perspective, P2P applications are modularized as services which operate at a layer above the overlay, each service interface is defined using a service description, and the overlay provides a generic service discovery and advertisement mechanism. As the number of P2P applications increases, this has the advantage of enabling the reuse of the same P2P infrastructure for a collection of



Service Overlays

Network Service	Overlay Example
Routing	Resilient Overlay Network Routing Overlay
Domain Name Service (DNS)	DNS via DHT
Multicast	Application Layer Multicast
Content Delivery	Stream Based Overlay Network (SBON)
AAA	Peering of RADIUS Domains
QoS	QoS Aware Overlay
Session Establishment	IETF P2P-SIP
Telephony Services	Relays Feature Servers

Service Oriented Architecture	Overlay Example
Service Discovery & Advertisement	INS/Twine
Service Composition	SpiderNet
Middleware	NEMO
Service Models	
Load Balancing	Beehive

Fig. 5 Types of service overlays [26]

applications. In an open P2P platform, it could also lead to the delivery of many new 3rd party applications.

Figure 5 shows examples of these two categories of service overlays. Further information on service overlays for routing, called resilient overlays, can be found in [90–94], and research on using overlays for DNS is discussed in [95–98].

2.6 Semantic Overlays

A semantic overlay is an overlay network in which routing topology is organized according to the semantic associations and relationship of information being stored in the overlay. Similar to the semantic web, content can be stored and accessed using a semantic model that is more convenient for the user. Several semantic overlays have been proposed such as Inga [37]. Challenges facing semantic overlays include:

- Agreeing on common ontologies within a community of peers
- Updating the ontology across the distributed set of peers when new concepts and relationships become important.
- Efficient semantic matching for object placement and search
- Implementing semantic behavior directly in the overlay routing mechanism versus layering it on top of an existing DHT or unstructured overlay query mechanism

2.7 Sensor Overlays

A sensor overlay is an overlay network that connects elements of a sensor infrastructure or grid. The purpose of a sensor overlay is to hide physical layer network constraints from applications, and to make data collection and infrastructure control logically separate from the physical layer routing. In addition, in a large sensor grid, there may be multiple planes, each with different physical routing layers and data collection. The sensor overlay can potentially unify these and make integration with conventional overlays simpler. An example sensor overlay is PIAX [99].

2.8 Research Directions

Many variations of overlays have been studied to date. Efforts continue to reduce latency, better adapt the overlay routing to the semantics of the applications, improve load balancing and response to flash crowds and similar dynamics, adapt to changing network conditions, and increase the ability of the overlay to self-organize.

3 Overlay Dynamics, Heterogeneity and Mobility

3.1 Churn and Overlay Maintenance

Peers may join or leave the overlay at any time. Overlays use join and leave protocols so that neighbors can update their routing state, and so that newly joined peers can quickly make connections with active neighbors. A candidate peer needs to discover an existing peer by which to join the overlay. The process of discovering and contacting an existing peer is called *peer bootstrap*, and involves mechanisms outside of the overlay such as contacting a well-known bootstrap server or making local broadcast announcements. When a peer joins the overlay, it typically receives its initial routing and object state from one or more peers designated by the bootstrap peer. After that, the peer modifies its state based on the operation of the overlay protocols. When a peer leaves the overlay it may signal its neighbors using a leave protocol. The neighbors then make changes to their routing state, and object state may be migrated or replicated as well. If a peer is disconnected without notification, neighbors use a heartbeat mechanism to detect the departure and trigger the corresponding routing and object state updates.

Churn is the arrival and departure of peers to and from the overlay, which changes the peer population of the overlay. *Overlay maintenance* is the operation of the overlay to repair and stabilize the overlay routing state in response to churn. The overhead for overlay maintenance increases as the churn rate increases. It also increases proportional to the routing state maintained by each peer, which is in turn proportional to the size of the overlay and the degree of each peer. There are techniques to reduce churn itself, such as incentives for peers to stay connected to the overlay. In addition, newly joined nodes can be quarantined, treated as client-only nodes, either due to limited capacity or until the peer reaches a lifetime threshold. This relies on the peer lifetime distribution being heavy tailed, which has been found to occur in practice.

Empirical data on churn in operational P2P applications has been gathered by deploying overlay crawlers. Crawlers connect to many other peers in the overlay in order to gather a snapshot of overlay membership. By continuing the process, information about membership changes and peer connectivity can be gathered. For P2P file sharing systems, measurements show a median peer lifetime of less than 1 hour and as little as 3 minutes, while as many as 2% of the peers will have a lifetime as long as one day [100]. On the other hand, peer lifetime measurements for Skype superpeers show a much longer median lifetime of about 5.5 hours [100].

Overlay maintenance can be classified as active or opportunistic. In active maintenance, routing table maintenance operations are triggered on peer join and leave events. An example active maintenance algorithm is EDRA (Event Detection and Recording Algorithm) used in D1HT [78]. As the churn rate changes, the routing state updates change proportionally. In opportunistic maintenance, routing table maintenance is performed as part of peer request routing or if the routing state falls below a minimum threshold. If peer request rates are high, for example for object lookups or inserts, then the routing state will be updated more frequently. An example opportunistic maintenance algorithm is that used in EpiChord [73].

While most analysis of overlays assume steady state is reached in which overlay maintenance matches the churn rate and the routing state enables the desired overlay geometry, there is growing thought that such an overlay state is not reached in practice, particularly for large overlays, due to the continuous changes to peer membership and time needed to propagate membership changes throughout the overlay. More likely, peers are not only out of sync with respect to the actual membership of the overlay but also have inconsistent routing state with other peers. To more accurately reflect the dynamics of the overlay, stochastic models of overlay membership have been developed for specific overlays, for example, [102, 103].

3.2 Mobility in P2P Overlays

While mobile nodes represent a small percentage of overlay peers today, in the future as the capability and network bandwidth increases and the population of such devices grows, this situation may be reversed. Thus the impact of mobility on the performance of the overlay is an important question [119].

Mobile devices have four properties that affect their interaction with the overlay in ways different from conventional desktop computers: roaming, energy limitations, node heterogeneity, and multi-homed interfaces. Network roaming causes IP address changes, and in conventional overlays, re-binding the overlay address to the IP address is effectively a leave-join sequence, leading to mobility-induced churn.

Approaches to mitigating mobility-induced churn include use of Mobile IP at the native layer, treating mobile nodes as stealth nodes [104], and designating a non-mobile node as virtual home agents for a mobile node [105]. Energy limitations of nodes increase the likelihood of a node going into a stand-by state. In today's overlays, this is likely to cause a node disconnect from the overlay.

Node heterogeneity means that nodes will not have equal capacity to store objects, participate in overlay maintenance, relay traffic from other nodes, and so forth. Variable hop overlays are one way to address these variations, and are discussed later in this section.

Multi-homed nodes are nodes that can connect to two or more different network interfaces at the same time. This could be used to provide redundant paths for peers to send and receive messages, which might reduce the impact of mobility induced churn.

3.3 Overlays for MANETs and Ad Hoc Networks

A mobile ad hoc network (MANET) is a set of mobile nodes which act as both routers and hosts in an ad hoc wireless network. The nodes route messages to other nodes without using a network infrastructure. Because of their limited power and capacity, MANET nodes transmit in range-limited broadcast messages which reach only nearby nodes. The MANET topology may change rapidly and in unpredictable ways.

As discussed in the introduction section of this chapter, integration of sensor grids, personal area networks, vehicular networks, and other ad hoc networks with Internet-based overlays is an important requirement for future global overlay based applications. Due to similarities between MANETs and the P2P model at both the application and network layer, there has been significant interest in adapting P2P overlays to work efficiency with MANET routing protocols. For example, many research systems have integrated flooding style unstructured overlays with MANETs. A summary of research activities is given in Table 5.

3.4 Heterogeneity and Variable Hop Overlays

A *variable-hop structured overlay* is a structured overlay that adapts the hop-count performance of the overlay according to the peer's network bandwidth budget so that at higher bandwidth budget the average hop count decreases and at lower bandwidth budget the average hop count increases. The performance of structured overlays depends on the accuracy and completeness of peers' routing tables, but more accurate and larger routing tables require more maintenance traffic. In addition, maintenance traffic grows with the churn rate and with the size of the overlay. Due to differences in nodes' network and computational capacity, different approaches have been proposed to avoid having all nodes operate at the least common denominator level. The superpeer architecture which elevates the more capable and more reliable nodes to full status is a common approach. Variable hop overlays as demonstrated in [115] are another important direction.

Variable hop overlays take advantage of the ability of the overlay protocol to adapt its bandwidth utilization through changing configuration parameters. Each peer adjusts its routing table size and accuracy according to the available bandwidth at that peer. During periods where the nodes have low bandwidth capabilities, overlay routing performance may reach that of multi-hop overlays while for higher bandwidth, routing performance reaches one-hop.

In addition to Accordion [115], other proposals for variable hop overlays include Tork [116] and Chameleon [117, 118].

Table 5 Features of P2P overlays for MANETs [26]

System	MANET routing algorithm	P2P overlay	Lookups	Evaluation size (nodes)	Node speed (m/s) and range
MHT [106]	GPSR	None	Key maps to node's path	1000 to 100,000	10–15 m/s $2000 \times 2000\ \text{m}^2$
Ekta [107]	DSR	Pastry	Prefix key-based	50	1–19 m/s $1500 \times 300\ \text{m}^2$
MPP [108]	Extended DSR	Gnutella	Flooding	50 ... 200	0–5 m/s $\leq 2000 \times 2000\ \text{m}^2$
XL-Gnutella [109]	OLSR	Gnutella	Flooding with superpeers	50	≤ 15 m/s not stated
MADPastry [110]	AODV	Pastry	Prefix key-based with clustering around landmarks	100 and 250	1.4 m/s $1000 \times 1000\ \text{m}^2$
FastTrack over AODV [111]	AODV	FastTrack	Flooding with superpeers	50	0–20 m/s $1500 \times 320\ \text{m}^2$
ORION [112]	Neutral, AODV and SMB	Unstructured	Flooding	40	0–2 m/s $1000 \times 1000\ \text{m}^2$
ISPRP [113]	DSR	Chord	Key-based	1000	NA
Dynamic P2P source routing [114]	DSR	Pastry	DP2PSR	800	9–19 m/s NA

3.5 Research Directions

Understanding the dynamics of large overlays in the face of changing peer populations, peer heterogeneity, peer mobility involves many challenging problems. While designing for heterogeneous peer populations involves relatively static peer distinctions, the ability to efficiently adapt peers to changing capacity and network conditions will involve considerations at a much smaller time scale. In addition, it remains an open question as to the practicality of large overlays if the majority of the devices are mobile. Finally, different models of node heterogeneity in terms of distribution and density in the overlay may become important in the design of overlay adaptation mechanisms.

4 P2P Content Access and Delivery

In recent years, the number of digital content, such as music and video, available on the Internet, the number of users accessing digital content through the Internet, and the number of digital video being streamed over the Internet each day are all growing exponentially. This obviously is placing an intense demand on the network bandwidth at the Internet backbone as well as on the servers that are offering the digital video and audio services. To improve content accessing scalability, Content Delivery Network (CDN) was invented and widely deployed in the last ten years. The evolution of CDN where a single sited content server is replaced by a set of distributed content servers placed strategically to provide not only better distribution of files but also better streaming of real-time media has many advantages. First of all, network congestion can be significantly reduced since servers are geographically distributed to better serve clients in given regions with low latency. Second, with servers placed at the edge of the network and closer to users, better quality of experiences can be expected for real time media streaming. The advantages of pushing content closer to end users suggest a natural extension of conventional CDN, P2P content delivery where contents are moved all the way to end users. It can potentially offer more advantages over traditional CDN, although it must be balanced with security, resource contention, and DRM issues.

Multicast is an effective content delivery method with reduced network bandwidth requirement. Due to cost issues, IP multicast has not been widely deployed. The explosion of streaming media applications thus offers another fertile ground for P2P based content delivery to blossom.

Taking advantage of the high scalability and the low cost in implementation properties of P2P networks, today P2P content access and delivery has become one of the most popular P2P applications. This includes P2P music sharing, P2P video sharing, P2PTV, P2P radio, P2P video streaming, etc. Just like in centralized content delivery systems, content can be delivered via downloading or steaming to the end users in P2P networks. At delivery, a media stream is segmented into data blocks that are delivered via flooding, random walk, or via a topology defined specific route in the P2P overlay network. Depending on the network topology, content blocks may be forwarded along a distribution tree rooted at the source peer or flown through a mesh network. And differentiated by the initiator of the content delivery, the content blocks may be pulled or pushed from the source peer to the destination peer.

Although many P2P content delivery applications are seen today, many fundamental technical issues are still not fully resolved, for instance in content search, content streaming, and content caching and replication.

4.1 Content Search

Searching is a key step in data access. It is certainly the case in P2P content access as well. P2P networks take advantage of the distributed resources at peer nodes. Contents are scattered and duplicated in the P2P network in a distributed fashion. Hence, content retrieval in a P2P network needs to contemplate the specific network model as well as the characteristics of the content being accessed. Ideally, a P2P content search algorithm should comprise support of complex queries, low cost in implementation, and fast and high accuracy query return capabilities. Today, most structured P2P networks support static key and ID based object lookups while unstructured P2P networks can handle certain complex type of queries, such as range queries. Although semantic query and content based query can enrich user experiences in content search, they are hardly supported by any P2P content delivery systems today. This is because those types of queries are still posing significant technical challenges.

Content search schemes and capabilities are largely dependent on the content indexing and management schemes as well as the P2P network topology. Table 6 summarizes P2P indexing schemes. In a centralized indexing system where the index is kept at a centralized location in the P2P system, content searching is generally done by forwarding the query message to the centralized indexing server to facilitate object lookup. The server returns the lookup result which contains the location of the desired content object. Content is transmitted then in a P2P fashion. Localized, distributed, and hybrid indexing schemes can effectively reduce the risk of network disruption owing to their distributed nature. Care must be taken when designing the query scheme to reduce the cost associated with query message forwarding and flooding.

Many people associate DHT with P2P search. This is because DHT based object lookup is a widely adopted search scheme in structured P2P networks. Most DHT based schemes rely on numerical keys to index and query objects in the P2P network. Object searching is accomplished using key distance and routing towards the peer that has the closest key to the querying object key. It offers efficient key or ID based exact match lookups with guaranteed query returns. However, managing a consistent DHT requires considerable effort due to the dynamics of the network topology. Another drawback of DHT-based system is its inability to support complex queries. In most applications, obviously, keyword, range, and semantic queries are more useful than key or ID-based exact match search.

Table 6 P2P indexing schemes

Indexing schemes	**Index location**	**Query propagation**	**Content object delivery**	**Key limitations**
Centralized indexing	Central server	Query is sent to the central server directly and resolved at central server	Querying peer obtains the content object location, i.e. the address of the source peer who has the content object; it then sends a request for delivery to the source peer directly; content object can now be delivered from the source peer to the destination peer directly	Vulnerability to attacks on the server and the possibility of bottleneck effect at the server
Localized indexing	Local peers	Query is propagated from peer to peer until the desired content object index is found	Content object may be delivered directly from the source peer to the query peer upon localization of the object	High cost associated with query flooding and low object retrieval efficiency
Distributed indexing	Distributed among peers	Query is forwarded to neighborhood peers based on peer routing table until the target object index is found	Querying peer obtains the location of the content object, sends a request to the source peer, and then receives the content object from the source peer	Possible delay at peer joining due to index set up
Hybrid indexing	Super nodes; super nodes and local peer nodes	Query is first searched locally at the local peer and the super node that is connected to the local peer directly. If content object is not found in local indices, query is propagated to other super nodes according to the routing table until it is found or a predefined Time-To-Live (TTL) threshold is reached	Querying peer obtains the location of the content object, sends a request to the source peer, and then receives the content object from the source peer	Powerful super nodes to sustain frequent query flooding are needed

Keyword search may be realized in P2P networks using the popular vector space model and/or inverted indexing based approaches. High cost associated with query flooding is one of the key drawbacks of those approaches if a non-centralized indexing scheme is employed. When the indices are distributed over peers, a simple query may cause a large amount of data being transmitted over the network. To reduce cost, one might take advantage of conventional information retrieval schemes. For instance, content summary based inverted indexing was proposed in [120]. Since a query can be processed by transmitting a much smaller candidate list, bandwidth demand for query flooding can be significantly reduced. The trade off is the additional storage space requirement and most importantly the reduction in recall rate in DHT based structured overlay. Today, how to implement efficient keyword search remains a challenge problem in P2P.

Compared to structured overlay, unstructured overlay has more freedom in implementing complex queries. Flooding, iterative depending, and random walk are commonly adopted searching approaches in unstructured P2P. Table 7 list the properties of these types of searching schemes in unstructured P2P networks.

Table 7 A comparison of searching schemes in unstructured P2P

Search scheme	Characteristics	Cost
Flooding	Query requests are flooded through the P2P network with the querying peer being the center of the flood	High. Massive amount of query messages are being transmitted for a single query
Iterative deepening	A growing ring is used to iteratively deepen the query flooding range until the target object is found [121]	High but lower than flooding based approach. Massive amount of query messages are being transmitted for a single query
Random walk	The querying node forwards (walk) the query message (walker) to one randomly selected neighbor which randomly selects its neighbor to forward the query message until the target object is located	Low to medium. In a K-random walk scheme, the cost is proportional to K while the delay is inverse proportional to K
Guided search	"Guidance" on where the query message should be forwarded is employed to improve query efficiency. Keyword vector, query similarity function, peer ranking and profile [122] may be used to guide the query forwarding	Low. Though recall rate can be significantly reduced if the "guidance" function is flawed

A query that retrieves all objects between an upper and a lower bound, i.e., an exclusive range, is called a range query. Likewise, a query that retrieves all objects within a multi-dimensional range is called a multi-dimensional range query. Methods to resolve range queries in classical database problems can be easily imported into unstructured P2P networks. However, it is relatively harder to achieve in structured P2P due to the "clear-cut" nature of a DHT. If range attuned numerical keys can be generated, range query could be supported in DHT based system. Locality Sensitive Hashing (LSH) [123] is one such approach to hash similar data partitions to nearby identifiers and similar ranges to the same peer with high probability. Noticeably, LSH has poor scalability. SkipIndex [124], another partition based scheme that offers a solution to range query also demonstrated impressive results in small scale P2P networks. How to design a scalable range query scheme and how to design a scheme that can offer scalable and efficient multi-dimensional range query support for DHT based P2P remain challenges.

In a traditional information system, the most challenging types of searches are semantic search and content based search. This is certainly true in P2P networks as well. How to support efficient semantic search and content based search for multimedia content access will need considerable effort and continuous investigation.

4.2 P2P Streaming and Multicasting

Content delivery, based on the way the content is transported and consumed, can be categorized into downloading and streaming modes. Streaming refers to the delivery method where content is being consumed while it is being transported. Compared to download based delivery, streaming poses significant challenge due to the time bounded requirement.

One-to-one, one-to-many, and many-to-many are possible configurations in different P2P streaming applications. For instance, a live remote personal video sharing could be one-to-one or one-to-many, an Internet video application often takes advantage of a one-to-many configuration, and a video conferencing application is likely to involve many-to-many communications. Consequently, unicast, broadcast, or multicast protocols may be employed in different streaming applications.

In a P2P network, content can be streamed via a tree based or a mesh based overlay. In tree based approach, content, rooted at the source node, is pushed along the tree to the destination peers. Mesh-based overlays implement mesh distribution graphs for content streaming. In the mesh distribution graph, each new node first obtains a content block availability map where a set of randomly selected peers who have the desired content blocks are listed, it then contacts a subset of those 'good' peers to request for streaming, and obtains the content blocks from those peers based on a predefined protocol. PPLive and Coolstreaming, for example, both take advantage of mesh based streaming. While the mesh based pull model offers better load balancing capability, it often introduces additional delays due to the exchange of buffer maps.

Tree based schemes on the other hand entail considerable control overhead at peer churns. If any interior member leaves the group (the tree,) the tree is broken and the children of the failure or departure node need to be reconnected to the tree. These entail additional group management cost. Tree based system is also inherently unbalanced. It does not utilize the bandwidth of the leaf nodes, causing a burden of duplicating and forwarding multicast traffic carried by a small subset of peers that are interior nodes of the multicast tree. This violates the fairness in resource and load sharing requirement in a P2P system. To improve fairness in resource sharing, multiple trees may be built to deliver different sub-streams. Splitstream [125], for instance, is one such scheme. Often this type of algorithms can work beautifully for a small scale P2P video streaming application. However, in a large scale P2P system, considerable complexity in building multiple balanced trees and tree reconnection at peer churns may significantly affect the system performance.

When there are multiple clients (receivers) simultaneously requesting/receiving the same media stream in a streaming application, multicast can be implemented. Multicast is a special type of streaming where protocols are defined to delivery a packet to a group of destinations at the same time using efficient strategies. Multicast can be deployed at different network layers. IP multicast which implements multicast at the IP routing level is generally high in implementation cost. P2P overlay multicast was invented to reduce deployment cost and improve scalability. A P2P overlay multicast system should implement [26]:

- Session identification
- Session initiation/creation
- Session subscription/join
- Session leave/graceful departure
- Session message dissemination/data forwarding
- Session fault tolerance/tree reformation at peer failure
- Session termination
- Session admission control
- Content access control and security

A comparison between P2P overlay multicast and IP multicast is given in Table 8. Obviously P2P networks offer considerable advantages, such as high scalability and low cost in implementation, for content streaming applications. The most considerable drawbacks in many commercially available P2P streaming and multicast video application systems such as PPLive include long startup delay and playback jittering. Quality of Experience (QoE), which indicates user experience and satisfaction, is a popular way today to measure the success of a content delivery service. Start-up delay and playback jittering are two important factors affecting user experiences. To reduce join and reconnection latencies in overlay multicast services, proximity based routing which improves arbitrarily long distances in routing hops is introduced [126]. Another approach [26] utilizes proactive step-parent selection to reduce the reconnection time in tree based multicast systems. That is, each peer locates its potential step-parent in advance. At tree reformation, a node that is a candidate parent immediately takes over the role of parenting. This cuts down real-time messaging needed for tree reconstruction, thus reducing the probability of playback jittering at the affected end hosts.

It was also shown that implicit protocols [127] where the control and data paths are defined simultaneously can support both latency-sensitive and high-bandwidth applications as well as very large group sizes.

Today, video streaming, one of the most popular means for content distribution, still suffer from several drawbacks when built on top of a fully distributed P2P overlay. These include high stress on the ISP links, dependency on high bandwidth peers, uneven quality distribution, lack of content security mechanisms and authentication capability, and long startup delay and channel switching delay. As a result, hybrid approaches are gaining considerable attention in the industry lately. Will hybrid solutions be able to offer reduced ISP stress and improved security as well as performance for large scale P2P streaming applications? Will hybrid systems ultimately solve the billing and accounting problem? These along with many other questions need to be resolved before P2P streaming takes on a full spin in commercial content delivery applications.

Table 8 Characteristics comparison: P2P overlay multicast and IP multicast

Metric	P2P overlay multicast	IP multicast
Efficiency	Relatively low	High
Stress on ISP	Relatively high	Low
Server bandwidth requirement	Significantly lower	High
Control overhead	Considerably higher	Low
Robustness	Generally lower	High
Lag between customers	Can be high	Low
Deployment cost	Usually very low	High

4.3 Caching and Replication

In P2P networks, data objects may be duplicated and saved temporarily or permanently on multiple peers. Caching and replication play key roles in reducing network bandwidth usage and origin server load and bandwidth requirement, reducing client side latency, and improving load balance, data availability, system reliability, and data access latency in a P2P network. Nevertheless, data consistency and synchronization issues need to be handled properly to reduce miss rate without high cost. The number of requests issued to peers for a particular content blocks and the frequency of cache replacement for instance, can affect the number of messages and network traffic pattern. Intuitively, flooding could guarantee object synchronization with the cost on additional communication messages and bandwidth requirement. Synchronization on demand could effectively reduce the communication cost, however, it may only offer a weak guarantee. Can a joint flooding and on demand approach offer reasonable guarantee with acceptable overhead? This still awaits investigation.

Tables 9 and 10 compare several different replication schemes and caching schemes respectively.

Caching and replication in structured P2P could be tricky. Some of the structured P2P systems associate an object to the object identifier which is also the key to discover the location of the object. Effectivecaching and replication of

Table 9 P2P replication schemes

Replication scheme	**Characteristics**
Full replication	Data are replicated on all peers in a P2P network. All data are available to read locally. The cost to maintain the replicas is high
Partial replication	A portion of the content is replicated at some peers. Relatively lower maintenance cost with longer seek time
Synchronous replication	All replicas are simultaneously changed. Higher hit rate with high cost in replication synchronization is expected
Asynchronous replication	Delay in change at a remote replica is allowed. Cost in replication synchronization is reduced with reduced hit rate
Static replication	Replicas are fixed at all times
Dynamic replication	Location and number of replicas change by time, system condition, transactional status, etc.
Active replication	Locations of replica are predefined. Query request is sent to all replica servers
Passive replication	Peer nodes request and copy content from one and another. Passive replication processes query request sequentially and synchronizes the replicated copies of objects periodically
Random replication	Replicas are placed randomly at peers
Query path based replication	Content (data) object is replicated on all peer nodes on the path from the query destination back to the query source
Adaptive query path based replication	Object shall not be uniformly replicated on all peer nodes on the path. Instead, the object popularity, peer resources, etc. shall be taken into consideration to decide where the object will be replicated
Neighborhood replication	Data objects are replicated at some or all neighbor peers of a peer that holds the objects
Object location replication	The location of data objects are replicated in the neighbors of the peer that holds the data object whereas the data objects are not replicated

Table 10 P2P caching schemes

Caching scheme	Characteristics
Just-in-Time (JiT) caching	Immediately after a request is received from the client, the cache pulls the content from the server with the content sent to the cache and the requesting client simultaneously
Pre-Caching (PreC)	Contents are often cached before a request is received at the proxy

objects requires additional mechanisms. In Tapestry [47], replica roots identified with random keys which are generated using a replication function are used for object replication.

4.4 Summary of Design Issues

Cost, implementation complexity, efficiency, robustness, scalability, and quality of services and experiences are some of the key design criteria for P2P content delivery services. For instance, in a video conferencing service, the system has to meet the delay bound constraint to offer acceptable customer experience. Video applications are often resource demanding. Thus, system control overhead may have direct impact on a video application system's performance. Furthermore, system capability to cope with churn and network dynamics is imperative in any P2P content delivery systems. An efficient system that can take advantage of the P2P network resources in a fair and balanced way can have a strong impact on system scalability and performance.

4.5 Research Issues

Noticeably, many popular P2P related brand names are associated with content delivery. For instance, Kazaa offers music sharing, PPLive provides P2P based TV service, and Pando presents video downloading, streaming, and sharing capabilities to its customers. Although tens of similar P2P based content delivery services are available today, major content providers, network service providers, or telco companies have not been deploying P2P based content delivery systems. Why? From technology point of view, there are many technical issues still need to be resolved before P2P network takes on a full spin in content delivery. Security, efficiency in searching and delivery, fairness in resource sharing, and billing and accounting, for instance, are some popular issues.

5 Security

5.1 The P2P Security Concern

Security is not just an issue in P2P content delivery, but also an important issue in almost all types of P2P applications. In fact, information security has been a daunting subject area in today's networked world. Information security aims at safeguarding information and information systems through a range of policies, strategies, security products, technologies and procedures. It includes the protection of information and system's availability, confidentiality, privacy, and integrity. Today, relying on personal computer and the Internet for information storage, retrieval and asset management is becoming an everyday practice for many individuals; whereas for many organizations, the network is a primary and mission critical components whose day-to-day operation must be fully warranted. Hence network vulnerabilities have significant impact on enterprise as well as personal information security today. With the growing frequency and types of threats, from viruses to Trojan horses, from adware to spyware, from denial of service to distributed denial of service, from fraud to identity theft, ... people are more and more aware of the security threats and more and more concerned with various security threats that are presented to them via different networks and applications.

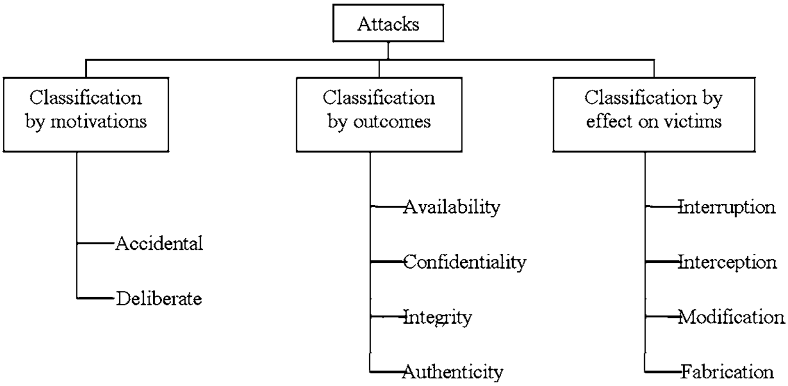


Fig. 6 Sample categories of attacks

With no central authority governing the authenticity and integrity of the sharing content and peers and with limited mechanisms for protecting the rights of content owners or the security of the client systems, P2P overlay network and applications add another dimension of security concern. Obviously, sharing files or computing resources on one's device with unknown peers over the Internet goes against many of the basic principles of securing your information. It could open up new doors for cyber criminals to steal confidential information, to damage personal or enterprise properties, and to poison the network for criminal intents.

5.2 Basic Classifications of P2P Network Security Threats

Through the years, many terms describing the security threats of networks or information systems were used. Some may be widely adopted and some may be confusing. Figure 6 and Table 11 list several popular categories and sample types of threats [26, 128–130].

Attacks on P2P network and systems may target at the overlay or the application layers [26]. Content theft, confidential information theft, service attacks, and computing and network resource theft and attacks are just some of the many types of threats P2P networks and application users are facing. Table 12 shows several unique P2P overlay security attacks defined in [131]. Those attacks could potentially

Table 11 Sample types of attacks

Category	Description	Sample attacks
Accidental	Security leakage caused by accidents	Accidental bandwidth clogging
Deliberate	Attacks are deliberate for criminal intent	Theft
Availability	Attacks that compromise the availability of the system or information, i.e. attacks that cause system or data unavailability to authorized or normal usage	Denial of service, bandwidth clogging, worm (e.g., using up the computer's resources and possibly shutting the system down)
Confidentiality	Attacks that compromise the confidentiality of the system or information, i.e. unauthorized disclosure of information	Theft, Trojan (e.g., one that logs keystrokes to steal information, RAT(Remote Access Trojan)) Spyware
Integrity	Attacks that compromise the integrity of the system or information, i.e. attacks that modify data without authorization	Virus
Authenticity	Attacks that compromise the authenticity of the system or information, i.e. attacks that target at or affect the genuineness of the information or the system	ID attacks, password attacks
Interruption	Unauthorized disruption	Distributed denial of service, bandwidth clogging
Interception	Unauthorized access	Theft, password attack, IP spoofing, nodeId attacks, Sybil attack
Modification	Unauthorized tampering	Reverse engineering
Fabrication	Unauthorized creation	Content/email spoofing

Table 12 Sample types of P2P overlay attacks [26, 131]

Category	Description
NodeId attack	Nodes obtain specific nodeId(s) (node identification) for malicious intent
Sybil attack	A small number of entities counterfeiting multiple peer identities so as to compromise a disproportionate share of the system
Message forwarding attack	Attacks that alter the route or the content of the message being forwarded for malicious intent
Routing table attack	Attacks that manipulate routing table entries for malicious intent
DDoS attack	Nodes work together to prevent a system from performing its task

jeopardize the availability, confidentiality, integrity, and/or authenticity of content, data, a system, a network, or a peer.

A P2P network is a particularly attractive platform for attackers to steal confidential information and to spread viruses. It often opens up a back door for hackers to easily gain access to devices and information that normally can not be accessed. For instance, a hacker can use a software tool, such as Wrapster, to disguise a confidential document. The confidential document, now appear to be a legitimate media content, such as an MP3 file, bypasses the enterprise security mechanisms and policies and is shared and transmitted through a P2P file sharing system. The receiver outside of the enterprise network can now unwrap the file and convert it into its original format. A piece of code, the virus, could also appear to be a popular file-sharing program and subsequently when downloaded, the virus gains access to the peers' data, information, and software on the device. Modifying data and files and destroying the file system are just two of the many damages a virus could cause. P2P networks also provided a fertile ground for attacks to cash in on a collection of peer resources to achieve malicious means. Distributed Denial of Service (DDoS) attack and Sybil attack are two most representative ones in this category. Denial of Service (DoS) attacks could cause service breakdown through disruption of physical network components; consumption of resources such as storage, computation, or bandwidth resources; obstruction of communications; and interference with configuration and state information. For example, a DoS attacker may use malware to max out a user's CPU time or crash a system by triggering errors in instructions.

5.3 Counter Measures

Leaking confidential information through P2P networks and applications is a primary concern at many organizations. In addition, bandwidth clogging, viruses, copyright infringement, etc. are also serious threats at the enterprise network level. Detecting and stopping P2P applications at the enterprise network level is a straightforward practice and is implemented by many organizations today. Notice though if a laptop is used in a P2P application while disconnected from the enterprise network, it may still introduce security breaches when it is reconnected into the enterprise network. For instance, P2P software vendors or application service providers may offer free P2P file sharing or other services. In order to generate revenue, they may bundle advertiser applications or activity trackers to its P2P application and services. The advertisement or activity tracking application software maybe piggybacked in its application program, much like an adware or spyware could be. Obviously, other types of adware and spyware may also be piggybacked. They are often downloaded without users' knowledge and may run in the background even when the peer machine is disconnected from the P2P network. This opens up a window for hackers and introduces various potential risks. For instance, the adware or spyware may bypass the enterprise network firewall when the machine enters the enterprise network. It may contain viruses or worms that will spread around the enterprise network once the machine is reconnected. It may contain other security flaws and it certainly will use additional resources including computational, storage, and even bandwidth resources. To better address security concerns from P2P, one must be able to stop P2P activities on their networked systems and every component of the systems completely. To do this, strong policies should be implemented to allow automated protection on each and every component of the network. It includes protecting all components from becoming nodes in P2P networks while they are on and off the enterprise network.

While banning P2P maybe an easy solution to protect ones network from attacks caused by P2P networks and applications, it's certainly not a viable solution for all networks. To fight against P2P attacks while maintaining the privilege to employ some P2P applications, specific counter measures may be designed and implemented. A semi-decentralized P2P system, for instance, may help to reduce many types of P2P security risks. In a semi-decentralized P2P system, a centralized trust entity maybe utilized for security administration. Noticeably, the centralized authority can also become the victim of P2P attacks, such as a DDoS attack, if proper counter measure is not taken. Similarly, hybrid P2P can effectively reduce some P2P security risks while the super peers in the hybrid P2P network may become the victim of DDoS attacks if proper counter measure is not employed.

Obviously, fully distributed security mechanisms are needed in fully decentralized P2P systems. Castro [132] introduced secure routing table maintenance, secure nodeId assignment, and secure message forward as several primitives for secure message routing in structured overlay. By imposing strong constraints on routing table, binding nodeIds to node IP address, message authentication, and some other mechanisms, improved security at message routing in structured overlay is expected.

Studies on P2P DDoS attacks show that pattern detection and advanced filtering mechanism may be helpful in detecting DDoS attacks, the explosion of new types of DDoS attacks making it one of the toughest to defend [133, 134]. In [133], Mirkovic suggests to deploy comprehensive protocol, system security mechanisms and abundant resources to improve the system resilience to DDoS attacks.

5.4 Fairness, Trust and Privacy Issues

Adar [135] reported a 70% free rider testing result in Gnutella. Gnutella is certainly not alone in experiencing P2P free riding where peers are consuming resources without fair contribution of their resources. To improve fairness in P2P resource sharing, auditing, incentive, and micro-payment based mechanisms are proposed.

A P2P system relies heavily on a set of distributed peers working properly and fairly together. Today, a large scale P2P system can be thousands to millions in size with peers interacting with unknown peers. How can peers establish and maintain trust between one and another in a P2P system especially a large P2P system? How to perform peer authentication? A centralized trust management entity could solve the problem and yet it tends to be a single point of attack. A hybrid system although does not suggest a single point of attack, could still become impaired when attacks are targeted on several super peers at the same time. Distributed trust management, on the other hand, could impose high cost and overhead for pair-wise peer authentication.

Privacy in P2P networks is another constantly raised issue today. Most P2P networks do not implement appropriate privacy governance mechanisms. While anonymous communication offer means to protect privacy, it is offset by the risks in trust and reduction in communication efficiency.

5.5 More on P2P Security

Due to the autonomous and distributed nature and the wide availability of replicated objects, making a P2P network secure is a big challenge. Exposure to theft, distributed viruses, worms, Trojan horses, spyware, or DDoS attacks are just some of the many types of P2P attacks we are facing today. With decentralized security mechanisms not fully in place and traditional server-based security schemes not offering suitable means for fully decentralized P2P network protection, P2P security remains a daunting subject in the P2P research field.

As introduced earlier in this chapter, many P2P systems are designed for a specific application. A systematic counter measure with clearly defined security goals and security schemes that are application and system driven and carefully designed protocols and systems based on appropriate security policies, coupled with security educated P2P system and application users, perhaps can be expected to improve P2P system security and defend against various attacks. To further understand the security risks and counter measures in P2P networks, additional discussions and references can be found in Chapter 14 of [26].

6 Summary

There has been a great deal of research to devise and improve on a large range of overlay-related design dimensions. Also the number of applications of overlays is increasing due to growing capacity of wireless networks and end devices, and the popularity of P2P applications among end users. We have attempted here to summarize and highlight the key results to date. Nevertheless there are certain major questions about the P2P paradigm and its use that remain including:

- What is the significance of the P2P paradigm as a general distributed systems architecture, and how will it evolve in practice with respect to the client-server paradigm?
- What are the barriers to adoption of the many research results by deployed systems, and how can these be avoided?
- Is the first mover advantage in P2P applications surmountable, and can balkanization of the P2P landscape be avoided?
- What is the likely long-term impact on internet architecture and service providers?

Further sources of surveys on the field of P2P networks include [136–139] as well as the books [1, 140].

References

Introduction

1. D. Clark, B. Lehr, S. Bauer, P. Faratin, R. Sami, and J. Wroclawski. Overlay networks and the future of the Internet, Communications & Strategies 63, 3Q2006.
2. D. Clark, B. Lehr, S. Bauer, P. Faratin, R. Sami, and J. Wroclawski. The growth of Internet overlay networks: implications for architecture, industry structure and policy, 33rd Telecommunications Policy Research Conference, Sept. 2005, available at http://web.si.umich.edu/tprc/papers/2005/466/TPRC_Overlays_9_8_05.pdf.
3. L. Peterson, T. Anderson, S. Shenker, and J. Turner. Overcoming the Internet impasse through virtualization, IEEE Computer, Apr. 2005, 62-69.
4. The SpoVNet Consortium, SpoVNet: An architecture for supporting future Internet applications, www.spovnet.de.
5. Ambient Networks Deliverable, System design of SATO & ASI, www.ambientnetworks.org/Files/deliverables/D12-F.1_PU.pdf, Sept. 2007.
6. R. Devine. Design and implementation of DDH: a distributed dynamic hashing algorithm, Proceedings of the 4th international Conference on Foundations of Data Organization and Algorithms, Oct. 13-15, 1993, D. B. Lomet (ed.), Lecture Notes in Computer Science, Vol. 730, Springer-Verlag, 101–114.
7. W. Litwin, M.-A. Neimat, and D. A. Schneider. LH: linear hashing for distributed files, ACM SIGMOD Record, 22(2), June 1, 1993, 327–336.
8. W. Litwin, M.-A. Neimat, and D. A. Schneider. LH: a scalable, distributed data structure, ACM Transactions on Database Systems (TODS), 21(4), Dec. 1996, 480–525.
9. C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment, Proceedings of the 9th annual ACM symposium on parallel algorithms and architectures, Newport, RI, June 23-25, 1997, 311–320.

10. D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web, Proceedings of the 29th Annual ACM Symposium on theory of Computing, El Paso, Texas, May 4-6, 1997, STOC '97, ACM Press, 654–663.
11. M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design, IEEE Internet Computing, 6(1), February 2002.
12. S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In Proceedings of Multimedia Computing and Networking 2002 (MMCN'02) (San Jose, CA, Jan. 2002).
13. S. Sen and J. Wang. Analyzing Peer-to-Peer Traffic Across Large Networks. In Proceedings of the ACM SIGCOMM Internet Measurement Workshop 2002 (Marseille, France, Nov. 2002).
14. Y. Chawathe, S. Ratnasamy, L. Breslau, S. Shenker, and N. Lanham. GIA: Making Gnutella-like P2P Systems. Scalable. ACM SIGCOMM 2003.
15. Y. Qiao and F. Bustamante. Structured and unstructured overlays under the microscope: a measurement-based view of two P2P systems that people use. In Proceedings of the Annual Technical Conference on Usenix'06 Annual Technical Conference (Boston, MA, May 30 – June 03, 2006). USENIX Association, Berkeley, CA, 31–31.
16. J. Liang, R. Kumar, K. Ross. The FastTrack Overlay: A Measurement Study. Computer Networks, 50, 2006, 842–858.
17. S. Baset and H. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol Columbia University, Department of Computer Science, Technical Report cucs-039-04. 2004.
18. S. Guha, N. Daswani, and R. Jain. An experimental study of the Skype peer-to-peer VoIP system. In IPTPS, 2006.
19. T. Hofeld. Measurement and Analysis of Skype VoIP Traffic in 3G UMTS Systems. 4th International Workshop on Internet Performance, Simulation, Monitoring and Measurement, IPS-MoMe 2006, Salzburg, Austria, February 2006.
20. K. Suh, D. R. Figueiredo, J. Kurose, and D. Towsley. Characterizing and detecting skype-relayed traffic, in Proceedings of IEEE Infocom (Infocom 2006), Barcelona, Spain, April, 2006.
21. H. Xie and Y. Yang. A Measurement-based Study of the Skype Peer-to-Peer VoIP Performance The Sixth International Workshop on Peer-to-Peer Systems. IPTPS 2007 (Feb 2007).
22. R. Schollmeier. A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. Peer-to-Peer Computing 2001.
23. S. Androutsellis-Theotokis and D. Spinellis. A Survey of Content Distribution Technologies. ACM Computing Surveys, 36(4), December 2004.
24. J. Buford. Management of peer-to-peer overlays, International Journal of Internet Protocol Technology, Special Issue on Management of IP Networks and Services, 3(1), 2008, 2–12.
25. B. Biskupski, J. Dowling, and J. Sacha. Properties and mechanisms of self-organizing MANET and P2P systems, ACM Transactions on Autonomous and Adaptive Systems 2(1), March 2007, 34 pp.
26. J. Buford, H. Yu, and E. K. Lua. P2P Networking and Applications. Morgan Kaufmann 2008.

Unstructured Overlays

27. I. Clarke. A Distributed Decentralized Information Storage and Retrieval System. Unpublished Report. Division of Informatics, University of Edinburgh. 1999. Online: http://www.freenetproject.org.

28. I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system, in Proc. ICSI Workshop Design Issues in Anonymity and Unobservability, Berkeley, CA, June 2000. (also: Hannes Federrath (Ed.): Designing Privacy Enhancing Technologies. International Workshop on Design Issues in Anonymity and Unobservability. Lecture Notes in Computer Science VOL. 2009, Springer-Verlag, Berlin/ Heidelberg 2001.
29. H.-E. Skogh, J. Haeggstrom, A. Ghodsi, and R. Ayani. Fast Freenet: Improving Freenet Performance by Preferential Partition Routing and File Mesh Propagation. In Proceedings of the 6th International Workshop on Global and Peer-To-Peer Computing on Large Scale Distributed Systems (CCGRID'06), p. 9. IEEE Computer Society, 2006.
30. H. Zhang, A. Goel, and R. Govindan. Using the small-world model to improve Freenet performance. Comput. Networks 46(4), Nov. 2004, 555–574.
31. Gnutella Protocol Specification version 0.6.
Online at http://gnutella-specs.rakjar.de/index.php/ Main_Page, accessed Nov 2007.
32. M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design, IEEE Internet Computing, 6(1), February 2002.
33. Y. Chawathe, S. Ratnasamy, L. Breslau, S. Shenker, and N. Lanham. GIA: Making Gnutella-like P2P Systems. Scalable. ACM SIGCOMM 2003.
34. J. Liang, R. Kumar, and K. Ross. The FastTrack Overlay: A Measurement Study. Computer Networks, 50, 2006, 842–858.
35. R. Morselli, B. Bhattacharjee, A. Srinivasan, and M. Marsh. Efficient lookup on unstructured topologies. Proceedings of the Twenty-Fourth Annual ACM Symposium on Principles of Distributed Computing (Las Vegas, NV, USA, July 17 – 20, 2005). PODC '05. ACM Press, New York, NY, 77–86.
36. K. Hui, J. Lui, and D. Yau. Small-world overlay P2P networks: construction, management and handling of dynamic flash crowds. Comput. Networks, 50(15), Oct. 2006, 2727–2746.
37. A. Löser, S. Staab, and C. Tempich. Semantic Social Overlay Networks. IEEE J. Sel. Areas. Communications, 25(1), 2007, 5–14.
38. J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. van Steen, and H. Sips. Tribler: A Social-based Peer-to-Peer system. Proc. of the 5th International Workshop on Peer-to-Peer Systems (IPTPS'06).
39. G. Mangioni, V. Carchiolo, M. Malgeri, and V. Nicosia, Evaluating the Dynamic Behaviour of PROSA P2P Network, International Symposium on Parallel and Distributed Processing and Applications 2006, ISPA06, 2006.
40. P. Ganesan, Q. Sun, and H. Garcia-Molina. Yappers: A peer-to-peer lookup service over arbitray topology. Infocom'03, Apr. 2003.

Broadcast in Structured Overlays

41. M. Lue, C. King, and H. Fang. Scoped broadcast in structured P2P networks. Proc. of the 1st International Conference on Scalable information Systems (Hong Kong, May 30 – June 01, 2006). InfoScale '06, vol. 152. ACM Press, New York, NY, 51.
42. V. Vischnevsky, A. Safonov, M. Yakimov, E. Shim, and A. Gelman. Scalable Blind Search and Broadcasting in Peer-to-Peer Networks, Sixth IEEE Intl. Conference on Peer-to-Peer Computing, Sept. 2006.
43. J. Li, K. Sollins, and D. Lim. Implementing aggregation and broadcast over Distributed Hash Tables. SIGCOMM Computer Communication Review, 35(1), Jan. 2005, 81–92.
44. H.-C. Hsiao and C.-T. King. Scoped broadcast in dynamic peer-to-peer networks. 29th Annual International Computer Software and Applications Conference, 2005. COMPSAC 2005. 26–28 July 2005, pp. 533–538 Vol. 2.

Structured Overlays

45. C. Greg Plaxton, R. Rajaraman, and A. Richa. Accessing nearby copies of replicated objects in a distributed environment, Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures, p. 311–320, June 23–25, 1997, Newport, Rhode Island, United States.
46. X. Li, J. Misra, and C. G. Plaxton. Active and Concurrent Topology Maintenance. In Proceedings of the 18th International Conference on Distributed Computing (DISC 04), p. 320–334, London, UK, 2004. Springer-Verlag.
47. B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment, IEEE Journal on Selected Areas in Communications, 22(1), pp. 41–53, Jan. 2004.
48. A. Rowstron, P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems, Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, p.329–350, November 12–16, 2001
49. K. Aberer, A. Datta, M. Hauswirth. Efficient, self-contained handling of identity in Peer-to-Peer systems, IEEE Transactions on Knowledge and Data Engineering 16(7), July 2004.
50. S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. OpenDHT: A Public DHT Service and Its Uses. Proceedings of ACM SIGCOMM 2005, August 2005.
51. Q. Lian, Z. Zhang, S. Wu, and B. Zhao. Z-Ring: Fast Prefix Routing via a Low Maintenance Membership Protocol. In Proceedings of the 13TH IEEE international Conference on Network Protocols (Icnp'05) – Volume 00 (November 06–09, 2005). ICNP. IEEE Computer Society, Washington, DC, 132–146.
52. I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Transactions on Networking 11(1) Feb. 2003, 17–32.
53. L. Alima et al., Dks(n,k,f): A Family of Low Communication, Scalable and Fault-Tolerant Infrastructures for P2P Applications, Proc. 3rd IEEE/ACM Int'l. Symp. Cluster Comp. and the Grid, Monterey, California, USA, 2003, pp. 344–50.
54. B. Carton, V. Mesaros, and P. Van Roy. Improving the scalability of logarithmic-degree DHT-based peer-to-peer networks,Proc. of Euro-Par, Aug.-Sept. 2004.
55. T. Schutt, F. Schintke, and A. Reinefeld. Structured Overlay without Consistent Hashing: Empirical Results. In Proceedings of the Sixth IEEE international Symposium on Cluster Computing and the Grid (Ccgrid'06) – Volume 00 (May 16 – 19, 2006). CCGRID. IEEE Computer Society, Washington, DC, 8.
56. G. Manku, M. Bawa, and P. Raghavan. Symphony: distributed hashing in a small world. In Proceedings of the 4th Conference on USENIX Symposium on internet Technologies and Systems – Volume 4 (Seattle, WA, March 26 – 28, 2003). USENIX Association, Berkeley, CA, 10–10.
57. P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In Proc of IPTPS02, Cambridge, USA, March 2002. 7.5 Constant Degree Overlays.
58. S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker. A scalable content-addressable network, Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, p.161–172, August 2001, San Diego, California.
59. D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: a scalable and dynamic emulation of the butterfly. In Proceedings of the Twenty-First Annual Symposium on Principles of Distributed Computing (Monterey, California, July 21 – 24, 2002). PODC '02. ACM Press, New York, NY, 183–192.
60. A. Kumar, S. Merugu, J. Xu, and X. Yu. Ulysses: A Robust, Low-Diameter, Low-Latency Peer-to-peer Network, in Proc. of IEEE ICNP 2003.

61. H. Shen, C.-Z. Xu, and G. Chen. Cycloid: A scalable constant-degree lookup-efficient P2P overlay network, Journal of Performance Evaluation's Special Issue on Peer-to-Peer Networks (6/29), 2005.
62. C. Shui, H. Wang, P. Zhou, and Y. Jia. Cactus: A New Constant-Degree and Fault Tolerate P2P Overlay. PRIMA 2006: 386–397.
63. F. Kaashoek and D. R. Karger. Koorde: A Simple Degree-optimal Hash Table, IPTPS, February 2003.
64. A.-T. Gai and L. Viennot. Broose: a practical distributed hashtable based on the de-bruijn topology, in Fourth International Conference on Peer-to-Peer Computing, 2004, Aug. 2004, pp. 167–174.
65. P. Fraigniaud and P. Gauron. D2B: a de Bruijn based content-addressable network. Theor. Comput. Sci. 355, 1 (Apr. 2006), 65–79.
66. M. Naor and U. Wieder. Novel architectures for P2P applications: The continuous-discrete approach. ACM Trans. Algorithms 3, 3 (Aug. 2007), 34.
67. G. Wepiw and P. Simeonov. HiPeer: A Highly Reliable P2P System. IEICE – Transactions on Information and Systems, E89-D(2), Feb. 2006, 570–580.
68. H. Rostami, J. Habibi, and A. Rahnama. Semantic HyperCup. In Proceedings of the 39th Annual Hawaii international Conference on System Sciences – Volume 09 (January 04 – 07, 2006). HICSS. IEEE Computer Society, Washington, DC, 223.
69. D. Li, X. Lu, and J. Wu. FISSIONE: a scalable constant degree and low congestion DHT scheme based on Kautz graphs. Proc. IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005), 3, 13–17 March 2005, 1677–1688.
70. D. Guo, J. Wu, H. Chen, and X. Luo. Moore: An Extendable Peer-to-Peer Network Based on Incomplete Kautz Digraph with Constant Degree. INFOCOM 2007.
71. Y. Zhang, X. Lu, and D. Li. SKY: efficient peer-to-peer networks based on distributed Kautz graphs. Journal Science in China Series F: Science in China Press, co-published with Springer-Verlag GmbH. Dec. 2008.
72. Y. Zhang, L. Liu, D. Li, and X. Lu. Distributed Line Graphs: A Universal Framework for Building DHTs Based on Arbitrary Constant-Degree Graphs. Proceedings of the 2008 the 28th international Conference on Distributed Computing Systems – Volume 00 (June 17 – 20, 2008). ICDCS. IEEE Computer Society, Washington, DC, 152–159.

O(1)-Hop Overlays

73. B. Leong, B. Liskov, and E. D. Demaine. EpiChord: Parallelizing the Chord Lookup Algorithm with Reactive Routing State Management. Computer Communications, Elsevier Science, 29, 2006, 1243–1259.
74. A. Gupta, B. Liskov, and R. Rodrigues. One Hop Lookups for peer-to-peer overlays Proceedings of the 9th Workshop on Hot Topics in Operating Systems (HotOS-IX), Lihue, Hawaii, May 2003.
75. I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse. Kelips: building an efficient and stable P2P DHT through increased memory and background overhead. Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03). 2003.
76. I. Abraham, A. Badola, D. Bickson, D. Malkhi, S. Maloo, and S. Ron. Practical Locality-Awareness for Large Scale Information Sharing. The 4th Annual International Workshop on Peer-To-Peer Systems (IPTPS '05). 2005.
77. M. Li, J. Hu, D. Wang, and W. Zheng. Gemini: Probabilistic Routing Algorithm in Structured P2P Overlay. The 3rd International Conference on Grid and Cooperative Computing (GCC 2004), Wuhan, China, Oct 2004.
78. L. Monnerat and C. Amorim. D1HT: A Distributed One Hop Hash Table. In Proc of the 20th IEEE Intl Parallel & Distributed Processing Symposium (IPDPS), April 2006. 7.7 Variable Hop Overlays.

79. J. Li, J. Stribling, R. Morris, and M. F. Kaashoek. Bandwidth-efficient Management of DHT Routing Tables, NSDI 2005.
80. A. Brown, M. Kolberg, and J. Buford. Chameleon: An adaptable 2-tier variable hop overlay. IEEE CCNC 2009, Jan. 2009.
81. A. Brown, J. Buford, and M. Kolberg. Tork: A Variable-Hop Overlay for Heterogeneous Networks. Fourth Workshop on Mobile Peer-to-Peer 2007. March 2007.

Hierarchical and Federated Overlays

82. M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. One ring to rule them all: service discovery and binding in structured peer-to-peer overlay networks, Proc. of the 10th workshop on ACM SIGOPS European workshop: beyond the PC, July 01-01, 2002, Saint-Emilion, France.
83. L. Garces-Erce, K.W. Ross, E. Biersack, P. Felber, and G. Urvoy-Keller. TOPLUS: Topology Centric Lookup Service, Fifth International Workshop on Networked Group Communications (NGC'03), Munich, September 2003.
84. Z. Xu, R. Min, and Y. Hu. HIERAS: a DHT based hierarchical P2P routing algorithm, International Conference on Parallel Processing (ICPP'03), Kaohsiung, Taiwan, 2003.
85. M. Artigas , P. Lopez , J. Ahullo, and A. Skarmeta. Cyclone: A Novel Design Schema for Hierarchical DHTs, Proc. of the Fifth IEEE International Conference on Peer-to-Peer Computing, p.49–56, August 31-September 02, 2005.
86. Q. Lian, Z. Zhang, S. Wu, and B. Zhao. Z-Ring: Fast Prefix Routing via a Low Maintenance Membership Protocol. In Proceedings of the 13th IEEE international Conference on Network Protocols (Icnp'05) – Volume 00 (November 06 - 09, 2005). ICNP. IEEE Computer Society, Washington, DC, 132–146.
87. P. Ganesan, K. Gummadi, and H. Garcia-Molina. Canon in G Major: Designing DHTs with Hierarchical Structure, IEEE International Conference on Distributed Computing Systems (ICDCS 2004), Tokyo, Japan, pp. 263–272, 2004.
88. M. Freedman and D. Mazières. Sloppy Hashing and Self-Organizing Clusters. Proc. 2nd Intl. Workshop on Peer-to-Peer Systems (IPTPS '03) Berkeley, CA, February 2003.
89. D. Braun, J. Buford, et al. UP2P: A Peer-to-Peer, Overlay Architecture for Ubiquitous Communications and Networking. IEEE Communications Magazine. 12/2008.

Service Overlays

90. D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. Proc. 18th ACM Symposium on Operating Systems Principles (SOSP) (Banff, Canada, Oct. 2001), pp. 131–145.
91. D. Andersen, A. Snoeren, and H. Balakrishnan. Best-path vs. multi-path overlay routing. Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (IMC'03), 2003.
92. S. Qazi and T. Moors. Scalable Resilient Overlay Networks Using Destination-Guided Detouring. Proc. IEEE International Conference on Communications (ICC), Jun. 2007.
93. Y. Zhu, C. Dovrolis, and M. Ammar. Proactive and reactive bandwidth driven overlay routing: A simulation study. Computer Networks, 50(6), April 2006, 742–762.
94. S.-J. Lee, S. Banerjee, P. Sharma, P. Yalagandula, and S. Basu. Bandwidth-Aware Routing in Overlay Networks IEEE INFOCOM 2008. The 27th Conference on Computer Communications 13–18 April 2008, p. 1732–1740.
95. R. Cox, A. Muthitacharoen, and R. Morris. Serving DNS Using a Peer-to-Peer Lookup Service. In Revised Papers From the First international Workshop on Peer-To-Peer Systems

(March 07 – 08, 2002). P. Druschel, M. F. Kaashoek, and A. I. Rowstron, Eds. Lecture Notes In Computer Science, vol. 2429. Springer-Verlag, London, 155–165.

96. J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS performance and the effectiveness of caching. Proceedings of the ACM SIGCOMM Internet Measurement Workshop '01, San Francisco, California, November 2001.
97. V. Pappas, D. Massey, A. Terzis, L. Zhang. A Comparative Study of Current DNS with DHT-Based Alternatives. IEEE INFOCOM 2006, April 2006.
98. V. Ramasubramanian and E. Sirer. The design and implementation of a next generation name service for the internet. In Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications (Portland, Oregon, USA, August 30 – September 03, 2004). SIGCOMM '04. ACM, New York, NY, 331–342.

Sensor Overlays

99. PIAX. piax.org

Churn and Overlay Maintenance

100. D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. In Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement Conference 2006 (IMC 2006), October 25–27, 2006.
101. S. Guha, N. Daswani, and R. Jain. An experimental study of the Skype peer-to-peer VoIP system, 5th Annual International Workshop on Peer-to-Peer Systems (IPTPS'06), 2006.
102. P. Kersch, R. Szabo, L. Cheng, K. Jean, and A. Galis. Stochastic maintenance of overlays in structured P2P systems. Elsevier Journal of Computer Communications, Special Issue: Disruptive networking with peer-to-peer systems, 31(3), 25 Feb. 2008, 603–619.
103. D. Wu, Y. Tian, K.-W. Ng and A. Datta. Stochastic analysis of the interplay between object maintenance and churn. Elsevier Journal of Computer Communications, 31(3), Feb. 2008.

Mobile P2P

104. A. MacQuire, A. Brampton, I. Rai and L. Mathy. Performance Analysis of Stealth DHT with Mobile Nodes. In Proceedings of the 4th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW 2006), March 2006.
105. H.-C. Hsiao and C.-T. King. Bristle: A Mobile Structured Peer-to-Peer Architecture. International Parallel and Distributed Processing Symposium (IPDPS'03), 2003. 7.13 Overlays for MANETs.
106. O. Landsiedel, S. Götz, and K. Wehrle. Towards Scalable Mobility in Distributed Hash Tables. Sixth International IEEE Conference on Peer-to-Peer-Computing, Cambridge, UK, August / September 2006.
107. H. Pucha, S. Das, and Y. C. Hu. Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks. Proc. 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), 2004, pp. 163–173.
108. I. Gruber, R. Schollmeier, and W. Kellerer. Performance evaluation of the mobile peer-to-peer service. Proceedings IEEE CCGrid 2004, 2004, pp. 363–371.
109. M. Conti, E. Gregori, and G. Turi. A crosslayer optimization of gnutella for mobile ad hoc networks. Proc. of the 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc05), pp. 343–354, Urbana-Champaign, IL, USA, 2005. ACM Press.

110. T. Zahn and J. Schiller. Designing Structured Peer-to-Peer Overlays as a Platform for Distributed Network Applications in Mobile Ad Hoc Networks. Compute Communications, 31(2), 5 February 2008, 643–654.
111. B. Tang, Z. Zhou, A. Kashyap, and T. Chiueh. An Integrated Approach for P2P File Sharing on Multi-hopWireless Networks. In Proc. of the IEEE Int. Conf. on Wireless and Mobile Computing, Networking and Communication (WIMOB'05), Montreal (Canada), August 2005.
112. A. Klemm, C. Lindemann, and O. Waldhorst. A special-purpose peer-to-peer file sharing system for mobile ad hoc networks. Proceeding Workshop on Mobile Ad Hoc Networking and Computing (MADNET 2003), 2003, 41–49
113. C. Cramer and T. Fuhrmann. ISPRP: A Message-Efficient Protocol for Initializing Structured P2P Networks. Proc. 24th IEEE International Performance, Computing, and Communications Conference (IPCCC 2005), 2005, pp. 365–370.
114. H. Pucha, S. M. Das, and Y. C. Hu. Imposed Route Reuse in Ad Hoc Network Routing Protocols Using Structured Peer-to-Peer Overlay Routing. IEEE Transactions on Parallel Distributed Systems, 17(12), Dec. 2006, 1452–1467. 7.14 Variable Hop Overlays
115. J. Li, J. Stribling, R. Morris, and M. F. Kaashoek. Bandwidth-efficient management of DHT routing tables. In the Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI '05), Boston, MA, 2005.
116. A. Brown, J. Buford, and M. Kolberg. Tork: A Variable-Hop Overlay for Heterogeneous Networks. Fourth Workshop on Mobile Peer-to-Peer 2007. March 2007.
117. A. Brown, M. Kolberg, and J. Buford. An Adaptable Service Overlay for Wide-Area Network Service Discovery. IEEE Globecom 2007 Workshop – Enabling the Future Service-Oriented Internet. Nov. 2007.
118. A. Brown, M. Kolberg, and J. Buford. Chameleon: An adaptable 2-tier variable hop overlay. IEEE CCNC 2009, Jan. 2009.
119. J. Buford. Mobile P2P after Five Years – Where are we and where are we headed? (Contributed Talk) Fifth IEEE Workshop on Mobile Peer-to-Peer. March 2008.

Content Access and Delivery

120. K. Yang and J. Ho, Proof: A DHT-Based Peer-to-Peer Search Engine, In Proceedings the IEEE International Conference on Web Intelligence, 2006, Hong Kong: IEEE Computer Society, p. 702–708.
121. B. Yang and H. Garcia-Molina. Efficient search in peer-to-peer networks, in Proceedings the 22nd International Conference on Distributed Computing, 2002.
122. V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-yazti. A local search mechanism for peer-to-peer networks, in Proceedings of the Eleventh ACM Conference on Information and Knowledge Management, 2002.
123. G. Gupta, D. Agrawal, and A. E. Abbadi. "Approximate range selection queries in peer-to-peer systems," in Proceedings of the First Biennial Conference on Innovative Data Systems Research, Asilomar CA, USA, 2003.
124. C. Zhang, A. Krishnamurthy, and R. Y. Wang. "SkipIndex: Towards a Scalable Peer-to-Peer Index Service for High Dimensional Data." Available at http://www.cs.princeton.edu/chizhang/skipindex.pdf
125. M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. "Splitstream: Highbandwidth multicast in cooperative environments," in Proceedings, the 20th ACM Symp. on Operating Sys. Principles (SOSP 2003), Oct. 2003.
126. H. Yu and J. Buford. Peer-to-peer overlay multicast, Book Chapter, in Encyclopedia of Wireless and Mobile Communications, Auerbach Publications, Florida, USA, to appear November, 2007.

127. S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast, in Proceedings, ACM SIGCOMM2002, September 2002.

Security

128. Web Application Security Consortium, Web Application Security Consortium: Threat Classification, Version 1.00, 2004. Available at http://www.webappsec.org/projects/threat/
129. S. Hansman and R. Hunt. A taxonomy of network and computer attacks, Computers & Security, 24(1), 2005, 31–43.
130. A. Simmonds, P. Sandilands, and L. van Ekert. An ontology for network security attacks, International Journal of Information Security and Privacy, 1(4), 2007, 1–23.
131. D. Wallach. A survey of peer-to-peer security issues, in Proceedings, International Symposium on Software Security - Theories and Systems, Tokyo, Japan, November 2002, p.42–57.
132. M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks, in Proceedings of 5th Symp. on Operating Sys. Design and Impl., Boston, MA, December 2002, 299–314.
133. J. Mirkovic and P. Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms, Computer Communications Review, 34(2), April 2004.
134. N. Naoumov and K. Ross. Exploiting P2P systems for DDoS attacks, in Proceedings of the 1st international conference on Scalable information systems, Hong Kong, 2006.
135. E. Adar and B. Huberman, Free riding on Gnutella, FirstMonday, 2000.

Surveys

136. S. Androutsellis-Theotokis and D. Spinellis. A Survey of Content Distribution Technologies. ACM Computing Surveys, 36(4), December 2004.
137. E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. IEEE Communications Surveys and Tutorials, Second Quarter, 7(2), 2005.
138. J. Risson and T. Moors. Survey of research towards robust peer-to-peer networks: search methods. Computer Networks, 50(17), Dec. 2006, 3485–3521.
139. S. El-Ansary, S. Haridi. An Overview of Structured P2P Overlay Networks. Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks (ed. J. Wu). Auerbach Publications, 2006, 665–683.
140. R. Steinmetz and K. Wehrle (Eds.). Peer-to-Peer Systems and Applications. Lecture Notes in Computer Science 3485 Springer 2005.

The Social Impact of P2P Systems

Andrea Glorioso, Ugo Pagallo, and Giancarlo Ruffo

Abstract The chapter deals with the social impact of P2P systems in light of a bidirectional connection by which technological developments influence, in a complex and often unpredictable way, the social environment whereas the dynamic evolution of the latter does affect technological progress. From this perspective, the aim is to deepen legal issues, sociological trends, economical aspects, and political dimensions of P2P technology, along with some of its next possible outputs, in order to assess one of the most compelling alternatives to the traditional frame of highly centralized human interaction.

1 Introduction

Although known to computer scientists and networking professionals for decades, peer-to-peer (P2P) systems only became widely popular in the late 1990s with the Napster case and, then, with the U.S. Court's decision in *MGM v. Grokster* in 2005. These developments, of course, affected legal scholars, who have paid increasing attention to P2P systems in recent years. However, their impact goes beyond the law and also raises issues of relevance for economics, sociology, and political science. This is particularly true if we understand P2P systems not only as simple "sharing

Andrea Glorioso
Any opinion expressed by A. Glorioso in this article is the sole responsibility of the author and does not necessarily represent the views of any organisation he has worked or is working with.
e-mail: andrea@digitalpolicy.it

Ugo Pagallo
Department of Law, University of Torino, Via Sant'Ottavio, 54, Torino, Italy,
e-mail: ugo.pagallo@unito.it

Giancarlo Ruffo
Department of Computer Science, University of Torino, Corso Svizzera, 185, Torino, Italy,
e-mail: giancarlo.ruffo@unito.it

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_2, © Springer Science+Business Media, LLC 2010

networks", but also, at a more abstract level, as massively distributed platforms for information storage and retrieval.

Therefore, the aim of the chapter is to highlight, in a necessarily non-exhaustive manner, the main research developments in the above-mentioned fields. We suggest that there is a clear bidirectional connection between the technological features of P2P systems and their various social effects. Technological developments influence, in a complex and often unpredictable way, the social environment (i.e., economics, sociology, politics, and law) whereas a dynamic situation in a given social environment also shapes the way in which technological evolution proceeds, favoring certain architectural choices over others. In short, each affects or gives feedback to the other in a continuous cycle. For example, a strongly decentralized and encrypted P2P architecture providing plausible anonymity for its participants will have different social ramifications and effects than those flowing from a weakly decentralized system in which the origin and destination of information can be traced with relative ease. At the same time, the technological developments that make one system rather than another possible will also be affected by the social environment, that is legal frameworks, economic decisions (including public funding), political orientations, and the sociological and cultural perceptions found in the society where the technological research is taking place and its results are diffused.

These inter-related developments are examined in five sections, each of which focuses on a specific field, namely legal issues (Section 2), sociological trends (Section 3), economic analyses (Section 4), political dimensions (Section 5), and, last but not least, some new horizons and perspectives that our society will be forced to cope with in the future (Section 6).

Of course, this overview does not purport to exhaustively discuss all the possible influences of P2P systems over the fields of human action, nor all the ways in which the complex composition of such different elements as law, economics, politics, sociology, and culture, tends to encourage, steer or limit the rate of development of certain P2P architectures over others (or, if one is pessimistic, the development of P2P architectures altogether, as opposed to centralized, one-way communication systems [67]).

Rather, our attempt should be taken for what it is: a humble, but hopefully stimulating, contribution to a better understanding of P2P systems as technological products that cannot actually exist in a vacuum, but in a complex environment with its own set of rules, constraints and possibilities.

2 Legal Issues

Among the most traumatic impacts of P2P systems on the legal field since the late 1990s, copyright issues have undoubtedly taken the forefront for a while. Technology has in fact changed societal behaviors of copying, distributing and, in general, handling information and information-based goods and services, according to a transformation that has deeply influenced legislative initiatives and scholarly discussions on the topic. Digital copyright is mostly a matter of access and control over information in electronic environments, so that legislators and courts have been obliged to rethink the traditional relationship between creators, producers, and consumers of informational goods. For instance, the U.S. Congress approved both the *Digital Millennium Copyright Act* (DMCA) [39] and the so called *Sonny Bono Act* in 1998 [37]; three years later, the European Parliament and the Council enacted the directive on "copyright and related rights in the information society" [36]. Then, in the U.S., the Congress passed the *Consumer Broadband and Digital Television Promotion Act* (2002), the *Family Entertainment Copyright Act* (2005), and the *Net Neutrality Bill* (2006), while, in Europe, the IPRED saga has been ruling: the first directive on the enforcement of intellectual propriety rights is from 2004 (n. 48), and later the European Parliament supported a new version (IPRED-2) on April 25, 2007.

Courts have been very active, too: the first important decision on copyright and P2P systems was in July 2000, when the U.S. District Judge Marilyn Patel granted the *Recording Industry Association of America* (RIAA)'s request to stop making copyrighted recordings available for download thanks to the Napster services. Even if the San Mateo company did not store any information such as the recordings on its own computers, it was considered against the law to provide the information of the songs available on the computers of the community logged on [39]. It is not enough, in other words, to affirm that the DMCA grants immunity to ISP providers for what their customers do, because this kind of protection does not include "contributory infringers" (as the District Court of Appeals determined in its own decision on Napster, in February 2001).

Four years later, it was the turn of the U.S. Supreme Court in *MGM v. Grokster* to consider P2P systems, such as Streamcast or Grokster, as a form of technology promoting the "ease of infringing on copyrights," so that its producers "can be sued for inducing copyright infringement committed by their users." Notwithstanding this unanimous holding by the Court, the legal consequences on further developments of P2P technology remained unclear. The Supreme Court justices were indeed divided in their arguments, between the need to protect every technology "capable of substantial non infringing uses" as they declared in *Sony v. Universal City Studios* from 1984, and the necessity to provide remedies against new ways of copyright infringement. So far, in the U.S., the problem is to determine whether software is creating "shared files folders" to make "available for distribution" that very information protected by copyright that would be shared via those folders [40].

In *Elektra v. Barker*, (see [33]), for example, Judge Kenneth Karas from the Manhattan federal court rejected the RIAA's "making available" theory in January 2008, even if he admitted the sufficiency of the allegations of "downloading" and "distributing", giving accordingly RIAA an opportunity to reformulate pleadings. Whereas Karas' idea is to represent the issue with the legal hypothesis of "offering to distribute for purposes of redistribution," it seems more fruitful to remark that the suit, in *Elektra v. Baker*, was based upon a report of an Internet investigator who claimed to have detected the "illegal files folder" we presented above.

In fact, the process of investigating alleged copyright infringements by P2P networks raises another central legal issue besides copyright, that is privacy. In the quest for new tools to help their fight against illegal file sharing, both the movie and music industry have posed serious dilemmas to Internet users, as it occurred some years ago with a highly controversial decision involving an American ISP, Verizon, and, again, the RIAA. The reason is that "the privacy of Internet users participating in P2P file-sharing practices is threatened under certain interpretations of the DMCA [as] a new form of *panoptic surveillance* that can be carried out by organizations such as the RIAA." [26]

Again, this privacy deadlock took place in summer 2007, when the *Motion Picture Association of America* (MPAA) asked and, according to federal judge Florence-Marie Cooper, it should have obtained the IP addresses of those connecting to TorrentSpy files via their service in the U.S. Forced to enable server logging against its own privacy policy, it is noteworthy, however, the different conclusion of the case: TorrentSpy, which servers are physically located in the Netherlands, announced its decision to stop doing business in the U.S. on August 27, 2007.

Despite the general tendency to reach harmonization in copyright law, there is indeed a clear difference of approach to P2P-related privacy issues between the U.S. and the European Union [49]. If a property standpoint prevails in the former legal system, privacy is widely considered as a fundamental right in the latter, as proclaimed by both the European Convention from 1950 and the EU Charter of Nice in 2000, let aside the specific constitutional traditions of each Member State. Conceived as a matter of protection, access and control over personal data, the traditional concept of privacy as the "right to be let alone" has evolved due to the same technological revolution that has conducted from traditional copyright to digital copyright. Thus, digital privacy has become one of the most relevant legal issues debated in recent cases, insofar as, at least in Europe, access and control over information allegedly reserved to copyright holders must respect P2P users' personal data.

For example, this is what took place in Italy in 2007 with the so-called "Peppermint" case. Although the legal reasoning of the court is rather complex [11–13] it is possible to summarize some key elements of its decisions in order to show the central role played by data protection laws. The facts are technically (but not legally) simple. A German record company (Peppermint Jam Records GmbH) suspected a large number of Italian users were downloading, without proper authorization, music from its catalogue via P2P networks. Hence, the copyright holder commissioned a Swiss company (Logistep AG) to monitor the activities of all the computers connected to those P2P networks. Logistep used a "crawler" which was able to connect to any P2P network as a normal client; but rather than simply exchanging digital files with other clients, Logistep's program recorded in its database additional data – including the IP addresses as well as date and time of the connection – of all the users who were sharing a file. Consequently, we have a twofold problem.

First, it is not entirely clear whether Logistep has been monitoring only users who were uploading (or rather, making available) Peppermint's songs on the Internet, or whether it was checking those who were downloading as well.

Secondly, it is likely that the database created by Logistep was, and still is, hosted in Switzerland, which raises a set of major legal issues insofar as the data "travelled" outside Italy and, so, outside the European Union.

In any case, the story went on partially as in the aforementioned TorrentSpy case. The plaintiff had indeed required before a section of the Tribunal in Rome to get from the Internet Service Providers responsible for collecting the IP addresses, the identity of the subscriber(s) using a specific IP address at a particular time and date. The tribunal of Rome accepted the request twice, therefore forcing the relevant ISPs to disclose such identities. So that, in spring 2007, a Bozen law firm representing Peppermint could send to every of the 3000 identified subscribers a "personalized" letter in which it was required a sum of EUR 330 for settling the case as a "partial compensation for damages, legal and technical expenses."

Nevertheless, a subsequent decision by another section of the same Court changed the whole picture on June 16, 2007. In a nutshell, the court affirmed that claims of the plaintiff, i.e., of the copyright owner of the songs being exchanged on P2P networks, should be duly balanced against the rights of P2P users, namely by protecting personal data as a fundamental right. Again, legal technicalities of the court's reasoning remain complex as they involve an analysis of the legislation on data protection, copyright and telecommunications, both at the Italian and European Union levels, along with various considerations on procedural safeguards which were not respected by allowing a private company to perform what amounts to be a full-blown police investigation. Still, the core element of this new decision seems evident: while personal data protection has been recognized by the Italian Constitutional Court's jurisprudence as a fundamental right, it is not admissible to conceive copyright protection as a justification for extremely invasive monitoring techniques.

In order to confirm how data protection laws are extremely relevant when defining cases on P2P file sharing systems, let us finally recall a recent decision made by the European Court of Justice in the "Promusicae" case (C-275/06) on January 29, 2008. In fact, the court was asked by a Spanish tribunal to clarify whether national law was compatible with the law of the European Union, because it did not seem to oblige ISPs to disclose identities of their subscribers for alleged violations of copyright law (as Promusicae, an association of Spanish music producers, was claiming). Indeed, the Spanish court was not asking the European Court of Justice to take a decision on the specific facts of the case, for it would simply have gone beyond the ECJ jurisdiction. Rather, it demanded the correct interpretation of EU law on this matter, pursuant to article 234 of the EC Treaty.

Put it shortly, the decision has been a double one: on one hand, it states that it is not required for any legal system of EU Member States, such as Spain, to make such disclosure compulsory. But, on the other hand, such a compulsory disclosure would not be incompatible with the law of the European Union. As the European Court of Justice affirmed, "when transposing those directives, the Member States [must] take care to rely on an interpretation of them which allows a fair balance to be struck between the various fundamental rights protected by the Community legal order. Further, when implementing the measures transposing those directives, the authorities and courts of the Member States must not only interpret their national law in a manner consistent with those directives but also make sure that they do not rely on an interpretation of them which would be in conflict with those fundamental rights or with the other general principles of Community law, such as the principle of proportionality."

So, the conclusion is that a delicate balance must be found here, insofar as P2P systems do not only involve private claims about possible copyright infringements, but also privacy concerns about data protection in digital environments. If we stressed how legal scholars still discuss, in the U.S., on the possibility to ascertain whether P2Ps are a technology capable of substantial non infringing uses, it is thus clear that, at least in Europe, such a copyright protection must go along with the fair respect of P2P user's personal data. A different constitutional approach to the informational nature of human beings and their relationships in digital environments leads in fact to a divergent way of understanding the most relevant legal issues created by this technology. As a first striking example of that bidirectional connection remarked since the introduction, the social impact of P2P systems involves different legal outputs that shape the very evolution of this technology.

3 Sociological Aspects

The legal misadventures of Napster and its ruin in 2002 are a good standpoint to introduce the sociological aspects of the analysis on P2P systems. Indeed, Napster was a centralized P2P network with a number of servers keeping information on the peers – namely, the files that each peer made available for distribution over the Napster network which answered to clients' searches for a particular file.

As we saw in the previous section, it was in fact this architecture to be considered illegal by the Courts: The operators of the central server(s) used to index each peer's files so that they could have intervened to stop copyright infringements. Of course, it can be questioned that the simple exchange of a file would automatically imply copyright infringements, therefore ignoring all of the defenses afforded by "fair use" and other similar doctrines. However, the point is to remark how the following generation of P2P systems has moved toward a more massively distributed way of sharing and exchanging information on the Internet.

Instead of a hierarchical network in which one central hub still represents its vertex, the horizontal nature of post-Napster P2P systems makes it difficult to impose centralized controls over the dissemination of information, reflecting, in some ways, the original design for the Internet. Besides, this horizontal architecture creates wider opportunities, both in scope and quantity, for the production of information on the Web. Yochai Benkler calls this process "commons-based peer production" to define collaborative projects such as free software [8–10], while Michel Bauwens proposes to interpret it as "the relational dynamic of distributed networks" where hubs may exist but are not obligatory as it happens with the Internet [7]. From a sociological viewpoint, however, it is important to stress, first of all, that several scientific papers have demonstrated the existence of so-called "small world" networks at various levels of any P2P system.

This new research relies on and deepens previous sociological work by such luminaries as Stanley Milgram [44] and Mark Granovetter [24, 25] on small world-social networks, or Friedrich Hayek's research [29] on spontaneous evolution of social systems. In fact, as in Milgram's "small world problem," the degrees of separation between nodes are significantly reduced in P2P systems, so the diameter of the network is quite shorter than that of a typical regular system. The reason was anticipated by Granovetter's seminal work on "the strength of weak ties," insofar as a small number of random links reduces exponentially the diameter of the network. If you consider for example a regular network with twenty nodes, each of which has four links, nodes would require an average of five steps in order to reach each other. But, it is sufficient to randomly rewire three nodes to decrease the degree of separation from five to three. This means that in a system of six billion people as contemporary world can be represented, it is enough to have only two random links out of ten thousand regular connections to determine the degree of separation in the network as small as eight. When random links increase from two to three, the degree of separation reduces from eight to five, i.e., something very close to Milgram's first approximation more than forty years ago!

This paradoxical property of small world-systems with clustering coefficients higher than those of random networks, and diameters shorter than those of regular ones, characterizes the spontaneous evolution of many complex networks. Let aside biological corroborations, as in the case of the C-Elegans' neural network studied by Duncan Watts and Steven Strogatz in 1998 [62], this occurs with the Internet as shown by Albert Làzslo Barabási in 2002 [6], or with both the jurisprudence and the majority opinions of the U.S. Supreme Court analyzed by Seth Chandler [14], James Fowler, and Sangick Jeon in 2005 [21], or, last but not least, with contemporary research that has found significant evidence of spontaneous clustering of users according to content that is distributed in P2P systems such as Gnutella or KaZaA [46, 47].

Furthermore, this latter research on P2P systems proposes several methods that can be used to detect the small world-phenomenon in various networks, including "data-sharing graphs" as in [31] or "affinity networks" as in [54]. The reason why lots of real complex social networks are spontaneously developing this way, can thus be understood according to Hayek's ideas on cosmos [29] with the dynamics and evolution of spontaneous orders, for high clustering coefficients and short diameters are the key parameters to understand how the distribution of the information is optimized in complex social systems (see [45]).

Yet, we are still missing a fundamental point. In Barabási's terms [6], real networks like the Web present a power law distribution that goes along with its characteristic "long tail" of information. It is in fact a tiny fraction of nodes extremely connected in the network, namely its hubs, that produces a peculiar scale-free effect dubbed as the "rich gets richer"-phenomenon, or, in Barabási's jargon, the "anti-democratic" nature of the hubs. This way of distributing information in a network has partly been confirmed by the aforementioned work on the U.S. Supreme Court jurisprudence, and on P2P systems as well. As the topological research of all the papers published in the history of U.S. legal journals confirms, most of the articles are scarcely consulted even by their authors, and only a very small percentage of essays is massively quoted by scholars! So, as it occurs with the phenomenon of globalization, the legitimacy of the hubs depends on the clustering coefficients of the network. What P2P systems obtain, most of the times, spontaneously on the Internet, is precisely what contemporary globalization lacks: shortening the diameter of the network via its hubs does not mean you have got higher clustering coefficients in the system [48].

However, some other scholars claim that proper P2P systems do not need any "subcenter," hub, or Super-peer, as it occurs with decentralized P2P networks, index authorum, or the jurisprudence of the U.S. Supreme Court. The idea is that only the relational dynamic that arises in distributed networks can properly be called "peer-to-peer" (see [7] 2.1.A). This is not, of course, a simple matter of definition or of fantasy, in a world with no "authoritative" nodes anymore. Rather, it is entwined with the evolution of collective intelligence [41, 63], cognitive capitalism [16, 60], and even netarchy [7].

So, after "spontaneous evolution" of P2P systems with their legal problems, between cosmos and taxis in Hayek's phrasing, it is time to shed further light on such a topic of classical philosophy via the economic analysis of P2P systems. The idea is in fact to deepen both the impact of this technology in contemporary society and the dynamic situation through which social environment shapes the way technology evolves, insofar as it was just economics, after all, the first field in which Hayek developed his analysis on cosmos and taxis, namely on the very connection between spontaneous orders and human planning.

4 Economic Trends

Let us start our economic section on P2P systems as we did with its legal issues, that is by focusing, first of all, on copyright topics: in fact, the impact of recent technology on society and economy can be introduced with current debate on DRM (Digital Rights Management [42]) vs. DRM-free systems. In particular, some time ago, Steve Jobs [32], in his public speech "Thoughts on Music," criticized DRM systems that Apple has been imposing in order to protect its music against piracy. This approach brings however to a paradox: while DRM-protected digital music is prevented from being played by devices of different producers, DRM-free content, which uses open formats (e.g., MP3 for music and MPEG4 for movies), can conversely be downloaded, distributed, copied and played on different devices. This diversity involves an implicit disincentive to legally buy copy-protected digital content, because only DRM-free files are mostly interoperable: after all, 97% of the music filling iPods is unprotected and of unclear origins.

Jobs' conclusions – echoing theses already expressed elsewhere [20] – are thus surprising, inasmuch as they suggest both to abolish DRM systems and to sell music encoded in open formats. Yet, even if the dream of a "DRM-free" world would finally occur, there is no obvious reason to believe that copyright infringement, one of the main reasons that has been alleged for the introduction of DRM systems, would decrease as such. Rather, it is likely that future legal market-models will have to consider serious, scalable, efficient, secure, and reliable alternatives to DRM-based centralized on-line stores.

The P2P paradigm seems to provide a technologically mature framework for this domain, possibly making digital content-sharing applications a valid solution even for small vendors and emerging artists. In fact, small-medium actors in the marketplace could hardly afford production and maintenance costs that can be very high when distribution is provided by means of a resilient Content Delivery Network (CDN) architecture (as used, for example, by iTunes, Yahoo!, or Microsoft Media Shop).

Furthermore, the combination of DRM technologies with centralized systems might arise several problems. It suffices to stress the complicated procedures that users of Yahoo! Music Store will have to go through just to get either a reimbursement or a DRM-free copy of the songs they have purchased, due to the announced (at the time of this writing) closing down of the service and specifically of the related DRM servers [65]. While it is hard to say whether this represents either the failure of DRM technology as implemented in the market or simply of Yahoo! Music Industry, it urges to rethink both the media market and the role of P2P systems.

Despite their huge potentials, we already explained why P2P systems have become infamous as file sharing applications that make particularly easy for users to access copy-protected files for free. Indeed, it is very difficult to trace peers' activity, and identification of abuses cannot be easily performed, given the absence of a central authority. Moreover, a business model is hard to find. It is questionable who should be involved as a provider in the transaction: should it be only the owner of a given object or also other actors? In the same way, P2P distribution frameworks lead to technical advantages, but their economical benefits are far from clear. Consider the case of the purchaser who becomes distributor of that good later on: why should he/she provide properly the content if the owner wants to be reimbursed?

Another general perspective opens up by considering that P2P networks offer the ideal scenario to exploiting the Long Tail business [3, 4]. The reason is that they promote a perfect infrastructure to sell a large number of unique items in relatively small quantities. Up till now, it is well known that centralized architectures are efficient and dependable, when servers are replicated and content is fairly cached. Therefore, even if CDNs are still expensive and hard to maintain, they are preferred by (dominant) content providers for enabling a resilient market that can be easily controlled and managed. On the contrary, P2P systems are scalable in principle and they can largely help providers reduce administrative costs, the problem being to determine whether this technology is mature enough for a dependable, fair and profitable market place infrastructure.

Besides, the economic and technological convenience of P2P systems seems to depend on selfish or altruistic users' behaviors (as illustrated by the notorious free-riding phenomenon [1]). Of course, the introduction of some mechanisms that provide incentives (e.g., the BitTorrent's tic-tac-toe strategy [15]) can stimulate cooperative behavior between users in order to fight the negative outcomes of free-riding, like degradation of the system performance, unpredictable availability of resources, or, under the most unfavorable circumstances, the complete collapse of the system. But, again, it is far from clear how a P2P based-economic model could be certain to prevent all of these flaws.

In any case, when discussing on the economic role of P2P systems in the real world, we need to keep in mind that the entire product's value-chain has really been upset. This is the main reason why "P2P economics" is a very active field nowadays, with many attempts to define the human interaction which occurs in the digital environment, according to the theory of informational goods, the Internet distribution chain with its variable and marginal costs of production and distribution, along with network externalities [55], cognitive capitalism, and the very value of the "network information economy" [10]. Some scholars [7] have even claimed that their work on P2P systems shows that a new paradigm is emerging, thanks to the superiority of the "free software/open sources" model as key of a new P2P economics.

Within this context, let us grasp here some peculiarities of the P2P business model by focusing on roles and interactions identified in [30], and referring to the Oslo's tutorial on "Peer-to-Peer Market Places" [53]. Although different application/service styles can be taken into account for a generic discussion, for the sake of simplicity (and also for the relevance of the topic) we will focus mainly on digital content-sharing and distribution.

First of all, we need to compare the digital domain with the traditional value-chain of a media object (e.g., a movie, or a music album) that is characterized by five distinct services or phases: authoring, production, distribution, delivery, and consumption. Authoring is related to the creation of the artifact, and may include a pre-production phase, for example with the preparation of a demo tape to be presented to a content producer. Production starts with a contract between the content author and a business organization, like a record company, adding a relevant value to the object. In this phase, the artifact is usually refined by a team of professionals, and finally copied on a given physical support, like an audio CD. The record company starts the distribution of produced objects, according to a given marketing strategy. Physical objects are shipped all around the country (and, sometimes, the world), increasing the cost of the artifact by means of transportation service-suppliers that would be using airplanes, trains, and so on. All the objects are delivered to the distributed network of malls and shopping centers that provide the final link to the customers, fixing the ultimate price for each single object. Finally, the client consumes the product, ideally closing the chain by paying a fee that covers all of the costs augmented after every step of the cycle.

Digital technology literally disrupts this value-chain. In fact, distribution and delivery collapse in one single service that is provided by a fully integrated information system implemented by means of complex server architecture, such as a CDN. This is the case of popular on-line stores, like iTune or Amazon that connect customers and content owners more rapidly through the Internet. In order to understand roles and interactions between market's actors, we report in Fig. 1 the model presented in [30].

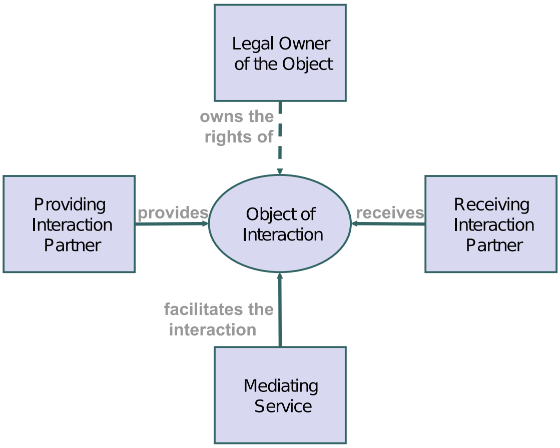

Legal Owner of the Object
owns the rights of
Providing Interaction Partner
provides
Object of Interaction
receives
Receiving Interaction Partner
facilitates the interaction
Mediating Service

Fig. 1 Roles and interactions

The main problem with P2P markets is that peers in the network can interchange all these roles. Quite trivially, a receiver can become a provider of a previously retrieved digital content in further interactions, due to the servent (both client and server) nature of a node. Nevertheless, in principle every user can access the P2P system to insert a digital object, and as a consequence the owner of the object can have the role of a provider, with or without third parties mediating between the authors and the market. Finally, a mediating service, such as a search engine, can be executed on a central server (e.g., Napster's search directory), or a decentralized one by using flooding strategies or other scalable and more sophisticated models (e.g., Gnutella [23], Kademlia [43], and so on). Hence, P2P disrupts the remaining traditional aspects that still survive in centralized digital market: distribution and delivery are provided by whomever manages the on line store, by subscribing a contract with the technology provider (e.g., Akamai for CDN infrastructures). If P2P is used instead, both services rely on contribution of the users, who may advocate their rights as authors, consumers, providers, and distributors.

From both an economic and a technological point of view, mediators are still necessary. Of course, production is a complicated process, and an artwork may sometimes be professionally arranged or refined. But if a revenue model is to be planned, other trusted third parties should be considered in the overall process, such as a bank, a credit card company, a payment broker when financial transactions may occur, or, finally, a certification authority when authenticated communications are needed.

In particular, users need a platform that has to be implemented and updated. Above all, they need a community where it is possible to exchange information, receive feedbacks, suggestions, comments, and so on. Such a community can also provide other mediating services, like a search engine or a financial broker (e.g., PayPal). Whereas we noted in Section 3 how often P2P systems spontaneously evolve in small world ways, it must be added that such a community may also plan a peculiar revenue model that can include membership or submission fees for sharing content, as well as mechanisms for enabling receivers to pay providers and/or mediators. However, all of these solutions may legally be problematic, because the legal owner is not (always) identical with the provider, let aside the hypotheses on who is responsible for the community to pay license fees, or to undersign an agreement with owners.

In conclusion, new mediators can release an open platform, represent the community, purchase licenses, digitalize content and insert items in the network (or enabling the users to insert items by their own). But the following points must be kept in mind, if the Grokster 's lesson has been learned:

First, the revenue for owners must be clear.
Second, the profit should be shared with providers and/or mediators, if the community owner wants to save both in terms of distribution and of delivery costs.
Finally, a dependable and scalable technical solution shall be adopted.

5 Political Aspects

The political dimensions of P2P systems play an obvious role in the further development of this technology, both at economical and legal levels. However, it is crucial, above all, to clarify what is meant by "political" in this context as well as the interpretation that should be given to the very term of "P2P systems."

Regarding the first point, we interpret the concept of "politics" rather broadly, in order to cope with all the ways in which people make decisions to get certain goals. This is certainly a wider definition than the one referred to the activities of civil governments, and in a certain sense it tends to include what is currently understood under the conceptual framework of "governance" [5].

Concerning the second point, we should distinguish between "P2P systems" as "technological systems" (according to the meaning which is usually given in engineering circles) and "P2P systems" as a metaphor for supposedly new emerging forms of human organization and participation [7, 10]. Although this latter interpretation is intellectually fascinating, it deserves a more thorough discussion than the one that can be offered here. Therefore, the focus will be on P2P as "technological systems" in order to highlight their relevance for political activities (in the broader sense specified above).

Indeed, the political role of "P2P systems" becomes a significant one as soon as we couple the technical definition of P2P networks as massively distributed platforms for information storage and retrieval, with a rather uncontroversial statement in politics. Namely, control of information and, hence, the possibility to obtain such information and to act upon it, are crucial issues for people making decisions to achieve definite ends.

This very possibility of sharing growing amounts of information via P2P networks empowers individuals with the information they need in order to "do politics"; such empowerment has a profoundly different nature as it regards mono-directional media such as TV, radio or traditional printed press, which are representatives of the "industrial information economy" [10]. In these one-to-many systems, in fact, it is much easier for a limited number of people to act as informational gatekeepers, effectively deciding what information should be distributed.

It is not possible to analyze in detail how control over information influences the sphere of political action. It is nonetheless rather intuitive that a large set of political decisions and of decisions with political implications is taken on the basis of certain information being available – or, much more importantly, being unavailable – to stakeholders concerned with the results of those decisions. Let aside particular well-defined circumstances, usually when a major threat to the wellbeing of the members of a political community is looming (as in time of war), modern democracies rest on the assumption that the availability of information to all citizens is a basic precondition for meaningful participation to decision-making processes and, therefore, to political life in the widest implication of the word.

So, P2P systems should be understood according to their capability to facilitate flow of information, particularly when such information is withheld by entities – including, but not necessarily limited to, the State or related entities – which intend to profit (economically or otherwise) from such control.

Obviously, it is not particularly useful to treat "P2P systems" as a single monolith when assessing their relevance – positive or negative – for political action. As we stressed in previous sections, it makes a real difference whether P2P systems are strongly centralized (e.g., Napster) or decentralized (e.g., Gnutella, Pastry [51], Kademlia [43]). In the same way, it is important to distinguish P2P systems which allow anonymous communications (e.g., Freenet [38] or Publius [61]); plausible deniability (e.g., Publius [61]); confidential communications, and so on.

For example, the presence of single points of failure in a centralized P2P system explains why it is likely that these networks suffer from disruption to their relevant information-sharing functions more easily than decentralized systems. This might happen when such a network is used by its participants to share information that is regarded by others as politically sensitive and/or damaging. The capability to clearly identify and target a limited number of nodes, either via legal means (e.g., court injunctions) or through other, less legal, ways (e.g., denial of service attacks), is obviously a strong advantage for the entity that considers such an information as damaging. Vice versa, a decentralized P2P system helps counter this threat, but it might render more difficult to clearly identify an authoritative source in sharing information, which is an important element in political activities. Besides, it may raise difficulties in coordinating such political relationships, notwithstanding the fact that information technologies themselves might provide tools, such as filtering, tagging and recommendation systems, for more effective coordination [10, 56].

Similar considerations can be applied to P2P systems that allow anonymous communication, that is making extremely difficult for an entity to stop spreading of sensitive information via identification and action upon the participants of the network. Anonymous P2P systems make the sharing of information, and possible political action based upon such a sharing, available (or less risky) even in those political regimes that have developed extensive monitoring infrastructures [17]. Nevertheless, full anonymity might also encourage irresponsible acts (in the strict sense of the word, i.e., acts whose responsible cannot be identified) or be used for criminal purposes. This hypothesis raises a lot of difficult questions, most of them related to the possible use of technologies allowing anonymous communications to criminal organizations. The usage of pseudonymous systems, such as OpenID (see on-line documentation at http://openid.net) might be a feasible partial solution to this complex challenge.

In the context of their usefulness for political action, the same debate would apply when examining P2P systems that provide both plausible deniability and high degree of confidentiality, usually by encrypting communication between peers. These characteristics are, strictly speaking, very different from the capability of a system to provide anonymity. In particular, the possibility to offer plausible deniability is very important in those political environments where the mere usage of tools allowing confidential communication might create serious issues to their participants. By the way, this is becoming familiar even in democratic regimes under the fallacious assumption that those who have "nothing to hide" will not want confidentiality in their communications [58] . In any case, it is hard to draw a precise line between the necessity of providing a venue for political information-sharing activities in countries where the surveillance of citizens is routine, and the possibility that such systems protect or even encourage criminal activities [18].

Another point that must be clearly stressed is that sharing information in a more efficient and/or effective way does not necessarily mean any action based upon that information would be taken. P2P systems can, as mentioned above, help lowering the transaction costs of group coordination, by easing the sharing of the specific information which is necessary, whether directly or indirectly, for such coordination [10, 56]. However, there are many other factors that play a role in facilitating or inhibiting political activities: let us mention here three of them.

First, the very distributed nature of P2P systems can represent a formidable obstacle to the information sharing which turns into real action. Indeed, geographical distances between persons who might share a common goal, are not, up to today, reflected in the topology of P2P systems (although, there are proposals of P2P systems that organize their topology on the basis of geographical proximity [34, 66]).

Secondly, the sheer amount of information to be transferred in P2P systems can turn into an obstacle to practical action, insofar as human peers may suffer from information overloading [57].

Finally, there is another possible problem that, to be fair, is not necessarily peculiar to P2P systems, but generally to decision-making processes in large or weakly connected social groups (which characterize some, but not all of, P2P systems.) In a nutshell, the suspicion is that participants will tend to coalesce around positions which are not necessarily the best ones, but rather the most widely accepted and/or least controversial ones [35].

The actual relevance of these issues is still widely debated, one central point of discussion being that P2P networks should not be compared to an utopian picture of both information sharing and political action, but rather to the current framework of highly centralized mass-media environment [10, 56]. In this way it is clear what remains to be assessed in terms of the political practices allowed by existing and emerging P2P systems along with their social impact.

Indeed, it is not necessary to insist, once again, on the political significance of any particular architecture, i.e., the bidirectional connection by which policy-making processes constrain or stimulate certain kinds of P2P architectures over others (as we saw in Sections 3 and 4), while digital architecture influences different kind of political actions (as we mention in this section).

Rather, it is crucial to analyse whether and how far information sharing, which is allowed by forthcoming P2P systems, actually turns into real action. Therefore, in order to enrich that bidirectional connection with which we are dealing throughout this chapter, a fruitful perspective is given by some technical details on the threefold levels of P2P services that are the subject of the next section.

6 Turning Forthcoming P2P Systems into Reality

The aim of this section is to let the P2P paradigm suit (some of) the legal, social, economic, and political issues underlined in previous parts of the chapter. We partially refer to the P2P Service Model defined in the Service Oriented Peer-to-Peer Service (SOPPS) [22], which comes from the Market Management of Peer-to-Peer Services (MMAPPS) European research project. In such scenario, the Service Model outlines the different types of services a peer can offer to other peers as well as the interfaces through which they can be accessed.

In particular, we present the service model introduced in [52] which shows a structure that can logically be divided in three different levels, that we present in a bottom-up order: (1) the *Overlay Level*, (2) the *Accounting Level*, and (3) the *Market Level*. These modules give different services to other modules, and even if we can think of them, at this present degree of abstraction, as a pile of protocols, applications can use functions and operations offered at different levels. Each module is defined in terms of the functions they export to the other parts of the framework (see Fig. 2):

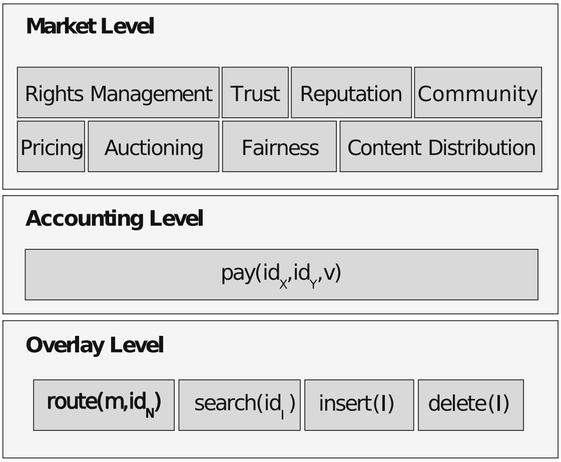


Fig. 2 Logical layers of a P2P service model

6.1 Overlay Level

The overlay network can be *unstructured* (e.g., Gnutella, Kazaa, and so on) or *structured* (e.g., Pastry [51], Chord [59] and Kademlia [43]). Scalable overlays are considered more scalable than the others. Very roughly, structured overlays are defined in terms of a *topology* (i.e., a forest, a ring, and so on), a *routing mechanism*, and an *identifier space*, which is used to uniquely locate nodes and resources in the network. The key idea of a *Distributed Hash Table* (DHT) system is that each node *N* is responsible for a set of resources whose identifiers are "closer" to *N*'s identifier than to the others; in fact, a *distance metric* is defined for each different system. Usually, a distributed storage service is defined over this basic layer. It is possible to insert a new resource in the network that will be assigned to the node of responsibility. Hence, given a generic node *N* and a resource *R*, respectively identified by id_N and id_R, the overlay layer should be able to export the following functions:

route(m, id_N): it routes message *m* to node *N*;
search(id_R): it looks for *R*, and returns a pointer to the node (or the set of nodes) that is responsible (or that caches) the searched resource.
insert(R) and *delete(R)*: these functions are used to store and to remove a given resource to a node. Some systems implement basic authentication mechanisms: for example, only a node with given credentials can access or modify a resource from the network (e.g., the Likir system defined on a Kademlia-like structure [2]).

The reader should observe that searches could be managed in a *centralized* (and largely efficient) manner. Napster is an example; however, as reminded in Sections 2 and 4, it might be considered illegal to centralize the provisioning of a directory service, even when the objects are stored on the computers of the community logged on. No one can implement a technology that prevents every kind of misuses. But, of course, a service provider (or a community owner) should avoid legal prosecution, through denying the accountability for other users' actions. On the contrary, decentralized searches in unstructured systems are not efficient (e.g., the flooding search strategy in Gnutella) or transfer the potential legal liability of this type on few super-peers (e.g., Kazaa-like policies). Additionally, a centralized system has a single point of failure presenting technical as well as political drawbacks, as we have shown in Section 5. This brings us to the first learned lesson that must be considered during the implementation of the given P2P service model:

Lesson Learned No. 1 *Structured overlays are to be preferred to unstructured P2P systems in order to let the community owners and/or the developers decline their responsibilities on users' actions. Moreover, such a system is free of single points of failure.*

Another important issue is given by anonymity and confidentiality, that should be granted as (optional) services to P2P users, as discussed in Section 5. It is critical that the solution is given at a lower level of the technological platform, because when using an overlay network, it is possible in principle to trace back to the content source (i.e., the user or the node that inserted a given object) and/or to the content provider (i.e., the user or the node that is storing/caching a given object). Secondly, the node identifier should not be coupled with the real identity of the user, in order to grant a given level of anonymity with the adoption of user generated pseudonyms. Other forms of identity management must be preferred instead (e.g., OpenId.)

Lesson Learned No. 2 *The Overlay level should integrate an authentication protocol and an identity verification policy that could accept pseudonyms instead of real generalities of users.*

The reader should observe that if peers communicate each other using authenticated channels, then they can easily cipher content before transmitting as well as storing and sharing it (for example using a Diffie-Hellman key exchange protocol). As a consequence, objects stored in the overlay system can be protected against not authorized access, without any responsibility of the content provider (when the latter is different from the source and/or the owner of the object).

6.2 Accounting Level

In order to satisfy economic constraints discussed in Section 4, security concerns at this level should be considered very seriously.We have to manage both macro-payment and micro-payment transactions; frauds are common; services and resources consumption must be accounted as well. The most part of actual markets adopts a central authority that manages economic transactions (e.g., PayPal) when services are provided or items are sold. At the resource level, many credit-based incentive mechanisms have been proposed over the last years. However, the accounting level must provide functions for *crediting* or *debiting* users (considering a *currency* with or without legal value). For the sake of simplicity, we define only the following function:

pay(id_X, id_Y, v): it invokes all the measures in order to securely provide a payment of value v from user X to user Y.

The *pay* method can be implemented by way of a central authority (very common, and maybe preferable, for managing macro-payments) or of a distributed system. In the latter class of proposals, many strategies can be further classified as *Local Accounting*, *Token-Based Accounting*, and *Remote Accounting*. For a more thorough discussion on different accounting strategies, see [30]

Let us stress that this accounting level is not necessarily related to the revenue policy that the community owner has decided to adopt, and that should be considered at the Market Level of the given model. In fact, several economic studies have sharpened how the flat-fee economic model can generate greater profits per good on respect to the pay-per-use model depicted in monetary systems (see for example [19]). Nevertheless, it has been clearly discussed how free-riding can be easily reduced if incentives for contributing are managed. Hence, crediting and debiting users introduce the need of a dependable business support, which must be scalable in terms of the many transactions that may occur in a P2P system.

Lesson Learned No. 3 *Crediting and debiting users can reduce the free-riding phenomenon, but they introduce the needing of an economic support. The execution of (many) $pay(id_X, id_Y, v)$ calls must be dependable, not repudiable, and scalable in terms of the number of transactions in a large community of users.*

Let us observe that commonly used Local Accounting strategies are trivially abstracted by the *pay* method: Indeed, both an eMule-like crediting system and a TicTacToe strategy (used in BitTorrent) can be seen in terms of a peer *paying* a fee to another peer, even if it results in a differentiated service (e.g., high priority in waiting queue of debtors) instead of real money transfers. Otherwise, other more reliable micro-payment systems must be used if the virtual coins correspond to real currency (see, for example, the Rivest's lottery scheme [50] and PPay proposed by Yang and Garcia-Molina [64]).

6.3 Market Level

This level includes most of the services as perceived by the final user. We list here a set of properties and functions that a framework could implement, in order to solve many of the issues we stressed in the previous sections of this chapter.

Pricing: We may define a pricing function that maps a service onto a tariff function or a scalar value that basically represents the price of the service consumption. It can be *fixed* or *competitive*; in the second case, each provider of the same service can offer it at different prices. For example, in a storage application, some peers can offer slices of its own disk space at a lower price than a competitor. PeerMint [27] gives an interesting approach to pricing mechanism in a P2P market.

Auctioning: When the good under sale is in limited number, the merchant can take different bids into consideration. In virtual markets that deal only with electronic sources (different replica can easily be produced), auctioning is less important. See PeerMart [28] as the state-of-the-art approach for enabling auctioning in a P2P system.

Intellectual Property Management: Even if there is no definitive solution to the DRM vs. DRM-free debate, we cannot underestimate the importance of intellectual property management, as discussed in Section 2. Although a flat-fee strategy is adopted, the community manager can be asked to account the content owner accordingly to the popularity of his/her own artifact.

Fairness: When the owner is credited under a given revenue model, then also the provider, who contributes to the system with her own bandwidth, cpu cycles, and disk space, should be (at least partially) credited as well. We think that this property has been deeply underestimated and a proper implementation of fair mechanisms can strongly incentive users to behave legally and bring success to a market place. The reader interested in such a topic, can refer to FairPeers [52].

Content Distribution: No electronic item must be delivered by traditional *shipping* methods (e.g., air mail). On the contrary, electronic content can alternatively be distributed depending on the kind of service. In fact, *delay-tolerant* (DT) connections (e.g., TCP based) can be used for file sharing applications, but *time-sensitive* (TS) mechanisms are needed when audio/video (e.g., a soccer match) is streamed to a set of paying users. If the task of content distribution is fairly shared by the participants, systems performance can optimize bandwidth available at the host side.

Trust and Reputation: When a transaction is completed (or even maliciously aborted), involved participants can be asked to submit a (positive or negative) feedback. Reputation management can be critical if pseudonyms are used instead of real identities: community owners may benefit from the users' active participation to isolate fraudulent users and to reduce misuses of the given platform.

Community Services: This higher level of abstraction can include some other services and functions (as a recommendation engine, social networking facilities, and so on), which can harness small world properties that characterize P2P community (see Section 3). A deeper discussion on such issues is out of the scope of this chapter. However, the reader can refer to *DeHinter*, a recommendation system for Gnutella network [54], that provides an example of a practical exploitation of spontaneous topological properties.

Therefore, a developer of a future P2P system must take care of each item in the given list, as it is stressed in the following learned lessons.

Lesson Learned No. 4 *In a P2P system every peer can be a merchant. The incremental exploitation of the P2P paradigm must include no trivial negotiation mechanisms, such as pricing and auctioning that are very strategic practices in the long tail business scenario (e.g., eBay.)*

Lesson Learned No. 5 *The U.S. Supreme Court decision in MGM v. Grokster case shows that in some parts of the world, developers can legally be prosecuted for inducing copyright infringement committed by their users. Intellectual property management must seriously be considered, and developers must provide a method for enabling legal sharing of copy-protected or otherwise governed information.*

Lesson Learned No. 6 *There are technological and political drawbacks in single points of failure. The needing of a distributed control over content and user's behavior makes mandatory the adoption of reputation and trust schemas.*

Lesson Learned No. 7 *The exploitation of the long tail scenario shows that the more the users are stimulated to contribute to the system, the more resources are available in the community. Fairness strategies can potentially expand the market as well as content distribution can benefit of user's enthusiastic participation.*

Lesson Learned No. 8 *Several scientific analyses have demonstrated the existence of small world networks at various levels of any P2P system (as well as other interesting patterns, such as power law distributions of nodes' degree). Spontaneous structures in network and community topologies can be exploited to enhance the performance of the given system (e.g. reducing the number of hops during searches) and to introduce new challenging social networking services (e.g., recommendation systems, lookup of neighbors of neighbors, and so on).*

7 Conclusions

Our analysis on the social impact of P2P systems has dealt with a work that obviously is still in progress. In Section 2, we mentioned the difficulties to precisely define the legal boundaries of a technology capable of substantial non infringing uses, along with the necessity to find out a fair (but difficult) balance between copyright and privacy issues. In Section 3, we referred to the spontaneous orders of P2P systems and their complex interaction, say, with the reality of taxis and legislation. In Section 4, we stressed how the potentialities of this technology do not have yet dissolved many doubts about its use as a dependable, fair and profitable marketplace infrastructure. In Section 5, it was the turn for political uncertainties which depend on geographical distances, overloading issues, or conformist risks, let aside the very possibility that such systems protect or even encourage criminal activities. Whereas, in Section 6, we depicted a plausible scenario for further developments of P2P systems, we were conscious that a key problem consists in turning forthcoming systems into reality (as the title of that section clearly warns).

Of course, all those open questions do not mean to forget merits and advantages of such a technology which has led many scholars to present it as a sort of new paradigm in social interaction. Rather, these issues prevent us from conceiving P2P systems as a form of panacea or utopia, insomuch as they invite to highlight the bidirectional connection stressed since the introduction and that has represented our personal thread of Ariadne throughout this chapter.

Indeed, we observed that legal networks, economical trends, and political debates are influencing P2P evolution, whereas, at the same time, those very systems have been transforming some key terms of our legal, economic, and political discourse. Hence, the picture we gave about a work that is still in progress should be also interpreted the other way around. In Section 2, we explained why initial apprehensions for copyright issues have left room to growing data protection concerns and, likely, new legal cases will arise in defense of freedom of research. In Section 3, we presented empirical work which proves P2P systems are part of a far more complex framework, i.e., Hayek's cosmos, irreducible as such to simple human programming as in the case of spontaneous affinity networks. In Section 4, it was the turn of classical political economics and how many of its central concepts have been upset, as it occurs with the value-chain of goods with their variable and marginal costs of production and distribution, networks externalities, or the very idea of informational goods. Then, in Section 5, we insisted on the new horizons opened up by P2P networks in politics, once properly understood in terms of information and, more precisely, in terms of control, access, and distribution for people making decisions to achieve definite ends. So, when we illustrated further developments of the technology as with the P2P service model in Section 6, the aim was not only to properly take into account the complex social environment with its own set of rules, constraints and possibilities, within which any technology has to be conceived. Rather, the aim was also to emphasize the way in which social interaction has already changed because of the introduction and functioning of P2P systems.

Finally, our analysis is obviously open to scientific debate and attempts of falsification; yet, our effort has been to ground it upon a historical balance of that continuous cycle in which each of its terms, namely technology and the social environment, affects or gives feedback to the other. While such a perspective means to take into proper consideration the social constraints within which any further development of P2P systems should be assessed, it also allows to ponder the consistency of some legal crusades, political queries, or economical misconceptions, still popular in current debate. After all, this is another good reason why it is so important to stress "the social impact of P2P systems".

References

1. Adar, E., Huberman, B.: Free riding on gnutella. Technical report, Xerox PARC, 2000.
2. Aiello, L.M., Milanesio, M., Ruffo, G., Schifanella, R.: Tempering Kademlia with a Robust Identity Based System, in Proc. of the 8th Int. Conf. on Peer-to-Peer Computing 2008 (P2P'08). IEEE Press.
3. Anderson, C.: The Long Tail, Wired, Oct. 2004.
4. Anderson, C.: The Long Tail: Why the Future of Business Is Selling Less of More, Hyperion, New York, 2006.
5. Apreda, R.: The semantics of governance (the common thread running through corporate, public, and global governance), University of CEMA Working Paper Series, number 245, 2003.
6. Barabási, A.-L.: Linked: The New Science of Networks, Perseus, New York, 2002.
7. Bauwens, M.: P2P and Human Evolution: Placing Peer to Peer Theory in an Integral Framework. http://integralvisioning.org/article.php?story=p2ptheory1. (Accessed July 30, 2008).
8. Benkler, Y.: Coase's Penguin, or Linux and the Nature of the Firm, in Yale Law Journal 112(3) 2004.
9. Benkler, Y.: Sharing Nicely: On Shareable Goods and the Emergence of Sharing as a Modality of Economic Production, in Yale Law Journal, 114(273) 2004.
10. Benkler, Y.: The Wealth of Networks, Yale University Press, New Haven, USA, 2006.
11. Blengino, C., Senor, M.: Caso Peppermint: file sharing e utilizzo di dati personali illecitamente trattati, in Altalex, 24 May 2007.
12. Blengino, C., Senor, M.: Caso Peppermint: la riservatezza delle comunicazioni prevale sul diritto d'autore, in Altalex, in Altalex, 12 September 2007.
13. Blengino, C., Senor, M.: Il caso "Peppermint": il prevedibile contrasto tra protezione del diritto d'autore e tutela della privacy nelle reti peer-to-peer, in XXIII(4–5) Il Diritto dell'Informazione e dell'Informatica, pages 835–850, Milano, Italy. Giuffré Editore, 2007.

14. Chandler, S.: The Network Structure of Supreme Court Jurisprudence. The University of Houston Working Paper Series, online at http://ssrn.com, 2005.
15. Cohen, B.: Incentives to build robustness in BitTorrent, in Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems, June 2003.
16. Cox, G., and Krysa, J.: Engineering Culture: On 'the Author as (digital) Producer', Autonomedia, 2005, p.74.
17. Deibert R. J., Palfrey J.G., Rohozinski R., Zittrain J.: Access Denied: The Practice and Policy of Global Internet Filtering, Cambridge (MA), USA, MIT Press, 2008.
18. Etzioni, A.: How Patriotic is the Patriot Act? Freedom versus Security in the Age of Terrorism, Routledge, New York-London, 2004.
19. Fishburn, P. C., and Odlyzko, A. M.: Competitive pricing of information goods: Subscription pricing versus pay-per-use. In *Economic Theory*, 13(2):447–470, 1999.
20. Fisher, W.: Promises to Keep: Technology, Law, and the Future of Entertainment, Stanford University Press, California, 2004.
21. Fowler, T., and Jeon, S.: The Authority of Supreme Court Precedent: A Network Analysis, online at http://jhfowler.ucdavis.edu, 2005.
22. Gerke, J., Hausheer, D., Mischke, J., Stiller, B.: An Architecture for a Service Oriented Peer-to-Peer System (SOPPS). Praxis der Informationsverarbeitung und Kommunikation (PIK) 26(2): 2003
23. Gnutella Protocol Development Group. The Gnutella Protocol Specification 0.4/0.6. `http://rfc-gnutella.sourceforge.net`.
24. Granovetter, M. S. : The Strenght of Weak Ties, in American Journal of Sociology, 78: 1360–1380, 1973.
25. Granovetter, M. S.: The Strength of Weak Ties: A Network Theory Revisited, in Sociological Theory, 1983, I, pp. 201–233.
26. Grodzinsky, F.S., Tavani, H.T.: P2P networks and the Verizon v. RIAA case : Implications for personal privacy and intellectual property, in Ethics and information technology, 7(4): 2005.
27. Hausheer, D., and Stiller, B.: Peermint: Decentralized and Secure Accounting for Peer-to-Peer Applications. In *IFIP Networking Conference*, pages 40–52, University of Waterloo, Waterloo, Ontario, Canada, 2005.
28. Hausheer, D.: *PeerMart: Secure Decentralized Pricing and Accounting for Peer-to-Peer Systems*. PhD thesis, Shaker Verlag, Aachen, Germany, ETH Zurich, TIK-Schriftenreihe No. 70, Diss. ETH Zurich No. 16200, 2006.
29. Hayek: Law, Legislation and Liberty: A New Statement of the Liberal Principles of Justice and Political Economy, Routledge & Kegan Paul, 1982.
30. Hummel, T., Muhle, S., Schoder, D.: Business Applications and Revenue Models, in: Steinmetz, R.; Wehrle, K. (eds.): Peer-to-Peer Systems and Applications (LNCS 3485), pages 473–489. Springer, 2005.
31. Iamnitchi, A., Ripeanu, M., and Foster, I.): Small-world file-sharing communities. In *The 23rd Conference of the IEEE Communications Society (InfoCom 2004)*, Hong Kong.
32. Jobs, S.: Thoughts on music. http://www.apple.com/hotnews/thoughtsonmusic/, February 6th, 2007. (Accessed July 30, 2008).
33. Kasunic, R.: Making Circumstantial Proof of Distribution Available, available at http://law.fordham.edu/publications/articles/200flspub12665.pdf
34. Kaune, S., Lauinger, T., and Kovacevic, A.,: Embracing the Peer Next Door: Proximity in Kademlia, in Proc. of the 8th Int. Conf. on Peer-to-Peer Computing 2008 (P2P'08). IEEE Press.
35. Keen, A.: The Cult of the Amateur: How blogs, MySpace, YouTube, and the rest of today's user-generated media are destroying our economy, our culture, and our values. Doubleday Business, 2008.
36. Kelling, D. T.: Intellectual Property Rights in EU Law: Free Movement and Competition Law. Oxford University Press, Oxford, 2003.
37. Landes, W. M., Posner, R.A.: The Political Economy Of Intellectual Property Law. American Enterprise Institute, 2004.

38. Langley, A.: Freenet, in Oram. A. (ed.), Peer-to-peer Harnessing the power of disruptive technologies, pages 123–132, Sebastopol (CA), USA. O'Reilly & Associates, Inc., 2001.
39. Lessig, L. *The Future of Ideas: the fate of the commons in a connected world* Vintage, 2002.
40. Lessig, L. *Code, and other laws of cyberspace, version 2.0* Basic Books, 2006.
41. Levy, P.: Collective Intelligence: Mankind's Emerging World in Cyberspace, translated by Robert Bononno, Plenum Trade, 1997.
42. Lyon, G.E.: The Internet Marketplace and Digital Rights Management, NIST Software Diagnostic and Conformance Testing Division 897, presented at the Conference on Infrastructure for e-Business, e-Education and e-Science on the Internet, L'Aquila, Italy, 6–12 August 2001.
43. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the xor metric. In IPTPS f01: Revised Papers from the First International Workshop on Peer-to-Peer Systems, pages 53V65, London, UK, Springer-Verlag, 2002.
44. Milgram, S.: The Small World Problem in Psychology Today, 60–67, May 1967.
45. Pagallo, U.: Teoria giuridica della complessità. Dalla "polis primitiva" di Socrate ai "mondi piccoli" dell'informatica – Un approccio evolutivo. Giappichelli, Torino, 2006.
46. Pagallo, U. and Ruffo G.: On the Growth of Collaborative and Competitive Networks: Opportunities and New Challenges, in Proc. of Ethicomp Working Conference 2007. In: S. Rogerson e H. Yang (eds). Yunnan University, 92–97, 2007.
47. Pagallo, U., and Ruffo G.: P2P Systems in Legal Networks: Another "Small World" Case, in: Eleventh International Conference on Artificial Intelligence and Law, Acm, Stanford, CA, 287–288, 2007.
48. Pagallo, U.: "Small World" Paradigm in Social Sciences: Problems and Perspectives, in Glocalisation: Bridging the Global Nature of Information and Communication Technology and the Local Nature of Human Beings, in: T. Ward Bynum, S. Rogerson, and K. Murata (eds.), e-SCM Research Center and University of Meiji, Tokyo, 456–465, 2007.
49. Pagallo, U.: La tutela della privacy negli Stati Uniti d'America e in Europa: modelli giuridici a confronto, Giuffrè, Milano, 2008.
50. Rivest, R. L.: Electronic lottery tickets as micropayments. In *FC '97: Proceedings of the First International Conference on Financial Cryptography*, pages 307–314, London, UK. Springer-Verlag, 1997.
51. Rowstron, A., and Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 329–350, 2001.
52. Ruffo, G., and Schifanella, R.: FairPeers: Efficient Profit Sharing in Fair Peer-to-Peer Market Places. Journal of Network and System Managegement 15(3): 355–382. Springer, 2007.
53. Ruffo, G.: Peer-to-Peer Market Places: Technical Issues and Revenue Models, in Inter-Domain Management, Proc. of the First Intern. Conf. on Autonomous Infrastructure, Management and Security, (AIMS 2007), Oslo, Norway, June 21–22, 2007, (LNCS 4543), Springer.
54. Ruffo, G., and Schifanella, R.: A Peer-to-Peer Recommender System Based on Spontaneous Affinities. ACM Trans. Internet Technol. (TOIT) 9 (1), Feb., 1–34. ACM Press, 2008.
55. Shapiro, C., and Varian, H.: Information Rules: A Strategic Guide to the Network Economy, Harvard Business School Press, Watertown (MA), USA, 1998.
56. Shirky, C.: Here Comes Everybody: The Power of Organizing Without Organizations. Penguin Press HC, 2008.
57. Shwartz, B.: The Paradox of Choice: Why More Is Less. Harper Perennial, 2005.
58. Solove, D.J. (2007): 'I've Got Nothing to Hide' and Other Misunderstandings of Privacy, San Diego Law Review 44, 745–772, 2007.
59. Stoica, I., Morris, R., Karger, D., Kaashoek, D., and Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *ACM SIGCOMM 2001*, San Diego, CA, 2001.
60. Topol, R., and Walliser, B. (eds): Cognitive Economics: New Trends, Emerald Group Publishing, 2007.

61. Waldman, M., Cranor, L.R., Rubin, A.: Publius, in Oram. A. (ed.), Peer-to-peer Harnessing the power of disruptive technologies, pages 145–158, Sebastopol (CA), USA. O'Reilly & Associates, Inc., 2001.
62. Watts, D. J., and Strogatz, S. H.: Collective Dynamics of "Small-World" Networks, in Nature, 393:440–442, 1998.
63. Wolpert, D. H.: Collective Intelligence, in Computational Intelligence: The Experts Speak, edited by David B. Fogel, and Charles J. Robinson, IEEE Neural Networks Society, Wiley-IEEE, 2002, pp. 245–260.
64. Yang, B., and Garcia-Molina, H. G.: Ppay: micropayments for peer-to-peer systems. In *Proc. of the 10th ACM CCS*. ACM Press, 2003.
65. Yoskowtiz, A.: Customers to be reimbursed over Yahoo DRM server shutdown, Afterdawn.com, http://www.afterdawn.com/news/archive/14916.cfm, 26 July 2008.
66. Zhao, B., Huang, L., Stribling, J., Rhea, S. C., Joseph, A.D., Kubiatowicz, J.D.: Tapestry: A Resilient Global-Scale Overlay for Service Deployment, in IEEE Journal on Selected Areas in Communications, (22), 41–53, 2004.
67. Zittrain, J.: The Future of the Internet And How to Stop It, Yale University Press, New Haven, USA, 2008.

From Client-Server to P2P Networking

Lu Liu and Nick Antonopoulos

Abstract Peer-to-peer (P2P) networks attract attentions worldwide with their great success in file sharing networks (such as Napster, Gnutella, Freenet, BitTorrent, Kazaa, and JXTA). An explosive increase in the popularity of P2P networks has been witnessed by millions of Internet users. In this chapter, an investigation of network architecture evolution, from client-server to P2P networking, will be given, underlining the benefits and the potential problems of existing approaches, which provides essential theoretical base to drive future generation of distributed systems.

1 Introduction

As a new design pattern, peer-to-peer (P2P) has been widely used in the design of large-scale distributed applications. An explosive increase in the popularity of P2P file sharing applications has been witnessed by millions of Internet users.

As an emerging technology, P2P networks attract attention worldwide, ranging from casual Internet users to venture capitalists. At the same time, the innovations of P2P networks also offer many interesting avenues of research for scientific communities. In the last few years, great research achievements have been made on P2P resource sharing and data transfer. Network architectures are starting to evolve from centralised client-server architectures to distributed P2P architectures or hybrid architectures between client-server and P2P.

Lu Liu
School of Computing, University of Leeds, Leeds, West Yorkshire, LS2 9JT, United Kingdom,
e-mail: luliu@comp.leeds.ac.uk

Nick Antonopoulos
Department of Computing, University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom,
e-mail: n.antonopoulos@surrey.ac.uk

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_3, © Springer Science+Business Media, LLC 2010

In this chapter, the network architecture evolution is discussed from client-server to P2P. A summary of recent solutions for resource discovery in P2P networks is also given underlining the benefits and the potential problems of these solutions.

2 Network Architecture Evolution

2.1 *Client-Server Architecture*

Many today's Internet applications, such as WWW, FTP, email, are distributed by using the client-server architecture (Fig. 1). In the client-server architecture, many clients request and receive services from the server(s). All the contents and services are stored and provided by a server(s). Content and services can be discovered and utilised efficiently by querying the centralised server(s), if the server(s) is available and capable of serving all clients at the same time.

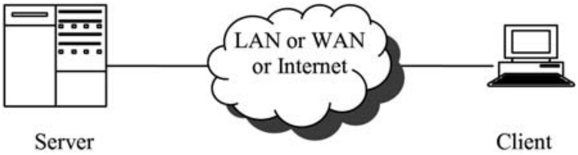


Fig. 1 Client-server architecture

However, such centralisation of the client-server architecture raises a series of issues which are caused by the limitation of resources at the server side, such as network bandwidth, CPU capability, Input/Output (I/O) speed and storage space. A server could be overloaded if too many requests are received. In order to cope with these limitations, the centralised server(s) needs to bear the high costs of providing sufficient resources. For instance, Google clusters more than 200,000 machines to give successful Web indexing services [1].

Moreover, the centralisation of the client-server architecture also leads to the problem of single-point-of-failure. If the centralised server(s) is removed or is not available for use, no alternative in the architecture can take its place and all services on the server(s) will be lost.

2.2 *Grid Architecture*

"Grid Computing" is rapidly emerging from the scientific and academic area to the industrial and commercial world. Current Grid computing systems are prominent implementations of client-server architecture for distributed computing.

The vision of the Grid Computing is to provide high performance computing and data infrastructure supporting flexible, secure and coordinated resource sharing among dynamic collections of individuals and institutions known as "virtual organizations" (VO) [2, 3]. The main focus of Grid architecture is on interoperability among resource providers and users in order to establish the sharing relationship, which needs common protocols at each layer of architecture. It is intended to offer seamless and uniform access to substantial resources without having to consider their geographical locations. Resources in the Grid can be high performance supercomputers, massive storage space, sensors, satellites, software applications, and data belonging to different institutions and connected through the Internet. The Grid provides the infrastructure that enables dispersed institutions (commercial companies, universities, government institutions, and laboratories) to form virtual organisations (VOs) that share resources and collaborate for the sake of solving common problems [2, 3].

2.3 Peer-to-Peer Architecture

P2P networks are decentralised distributed systems and enable computers to share and integrate their computing resources, data and services. Although concepts of P2P and Grids have a significant amount of overlap, they were originally proposed to address different domains. Whereas P2P is generally applied to a wide range of technologies that can greatly increase the utilization of collective natural resources at the edge of the Internet, such as information, bandwidth and computing resources, Grids are intended to promote interoperability and extensibility among various applications, platforms and frameworks [4].

In contrast to the existing Grid paradigms, P2P architecture does not rely on a centralised server to provide services (Fig. 2), which offers an appealing alternative to the existing Grid models especially for large-scale distributed applications. In the P2P model, each peer node (also known as servent) acts as both client and server, requesting resources from as well as routing queries and serving resources for other peer nodes. A P2P network is a logical overlay network over a physical infrastructure as illustrated in Fig. 2, which provides a virtual environment for P2P

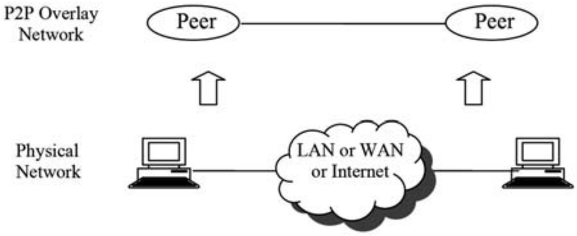


Fig. 2 Peer-to-peer architecture

developers to easily design and implement their own communication environment and protocols on the top of existing networks.

In recent years P2P networks have grown seemingly exponentially in popularity and utilisation. P2P file sharing has become one of the most popular Internet activities. Today's popular P2P file-sharing applications, such as Kazaa and Gnutella, have more than one million users each at any point of time [5]. According to research results from Cachelogic [6], about 50–65% of "downstream traffic" (i.e. from ISPs to endpoint devices) and 75–90% of "upstream traffic" (i.e. from endpoint devices to ISPs) on the Internet are the results of P2P file-sharing applications.

The popularity of P2P is motivated by the benefits it offers to end users. Compared to the client-server architecture, the advantages of P2P are listed below:

- P2P frees users from the traditional dependence on central servers, which enables end users to easily share resources (e.g. music, movies, games and other software). End users can share or retrieve resources directly from their connected machines without any further need to upload them to a centralised server.
- P2P applications are more resilient than those built on the client-server architecture by removing the single-point-of-failure.
- P2P distributes the responsibility of providing services from centralised servers to each individual peer node in the network.
- P2P exploits available bandwidth, processor, storage and other resources across the entire network. P2P interactions are only between individual peers which eliminate the bottleneck of centralised servers.
- Most P2P applications use virtual channels for communication which break the obstacles of corporate private networks, such as firewalls and Network Address Translation (NAT).
- P2P has better availability as each peer node can obtain content from multiple peer nodes. If one peer node is overloaded or experiences a hardware failure, other peer nodes in the network can still handle requests.

3 Evolution of Peer-to-Peer Networks

There are many interesting types of P2P applications, including file sharing, instant messaging, VoIP, Streaming media, High Performance Computing, search engine. Among them, file sharing, one of the most popular on-line activities [7], is the initial motivation behind many of successful P2P networks. P2P file sharing has become one of the most popular Internet activities. Today's popular P2P file-sharing applications, such as Kazaa and Gnutella, have more than one million users each at any point of time [5]. In this section, the history of the P2P file sharing networks is discussed along with the most popular file sharing applications.

Existing P2P file sharing networks can be divided into three categories [8] according to the degree of network centralisation: centralised P2P networks, decentralised P2P networks and hybrid P2P networks.

3.1 Centralised Peer-to-Peer Networks

Although P2P is often seen as an opposite model to the centralised client-server paradigm, the first generation P2P systems (e.g. Napster) started with the concept of centralisation. However, in contrast to traditional client-server systems, the server(s) in centralised P2P networks only keeps the meta-information about shared content (e.g. addresses or ID of peer nodes where the shared content is available) rather than storing content on its own.

- **Napster**

  Napster was the first widely-used P2P music sharing service. Before Napster came along, Internet users only passively operated their connected computers, such as browsing news or checking email. With the increased popularity of Napster, ordinary Internet users started opening their PCs to actively contribute resources and played more important roles for the Internet.

  Compared to follow-up P2P applications, Napster utilises a simple but highly efficient mechanism to share and search files in the network. To participate in the Napster network, new users need to register to the Napster server and publish a list of files they are willing to share. To search for a shared file in the network, users can request the Napster server and retrieve a list of providers hosting the files which match the query. File transfer takes place without the Napster server participating. The requested file is transferred directly between the requester and the provider as shown in Fig. 3.

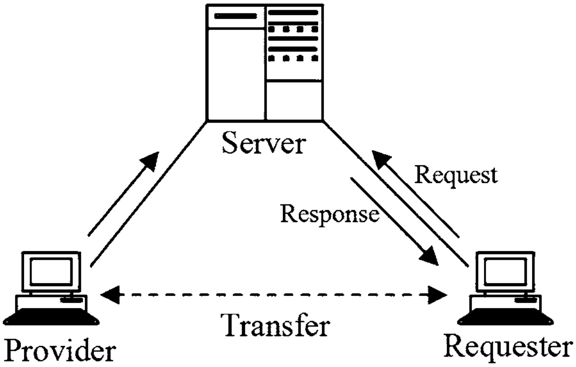


Fig. 3 Example of Napster network

- **BitTorrent**

  BitTorrent is designed to distribute large amounts of data without incurring the corresponding consumption in server and bandwidth resources. The original BitTorrent (before version 4.2.0) can be looked at as a Napster-like centralised P2P system. In order to share a file or a group of files, users need to create a small

.torrent file that contains the address of the tracker machine that launches the file distribution. The .torrent file is published on well-known web-sites, so that other users can find and download the .torrent file of interest using web search engines. The .torrent file is opened by the BitTorrent client software. The client software connects to the tracker machine and receives a list of peer nodes that are participating in transferring the file. In order to distribute a file efficiently, a file is broken into smaller fragments (typically 256 KB each) for transmission. The client, attempting to download the file, simultaneously connects to these peer nodes that are participating in file transfer, and downloads different pieces of the file from different peer nodes. In the meantime, the client can also upload downloaded pieces to other participants.

3.2 Decentralised Peer-to-Peer Networks

To address the problems of centralised P2P networks (such as scalability, single-point-of-failure and legal issues), decentralised peer-to-peer networks become widely used, which do not rely on any central server.

- **Gnutella**

  The Gnutella network is a decentralised file-sharing P2P network, which is built on an open protocol developed to enable peer node discovery, distributed search, and file transfer.

  Each Gnutella user needs a Gnutella client software to join the Gnutella network. The Gnutella client software on initial use can bootstrap and find a number of possible working peer nodes in the network and try to connect to them. If some attempts succeed, these working peer nodes will then become the new node's neighbours and give the new node their own lists of working nodes. The new node continues to connect to these working peer nodes, until it reaches a certain quota (usually user-specified). The new node keeps the peer nodes it has not yet tried as backup. When a peer node leaves the P2P network and then wants to re-connect to the network again, the peer node will try to connect to the nodes whose addresses it has stored. Once the peer node re-connects into the network, it will periodically ping the network connections and update its list of node addresses.

  In contrast to Napster, the Gnutella network is a decentralised P2P file-sharing network not only for file storage, but also for content lookup and query routing. Gnutella nodes take over routing functionalities initially performed by the Napster server. Figure gives an example of query propagation over the Gnutella network. In the Gnutella network, each peer node uses a Breadth-First Search (BFS) mechanism to search the network by broadcasting the query with a Time-to-Live (TTL) to all connected peer nodes. TTL represents the number of times a message can be forwarded before it is discarded. Each peer node receiving the query will process it, check the local file storage, and respond to the query if at least one matched file is found. Each peer node then decreases TTL by one and

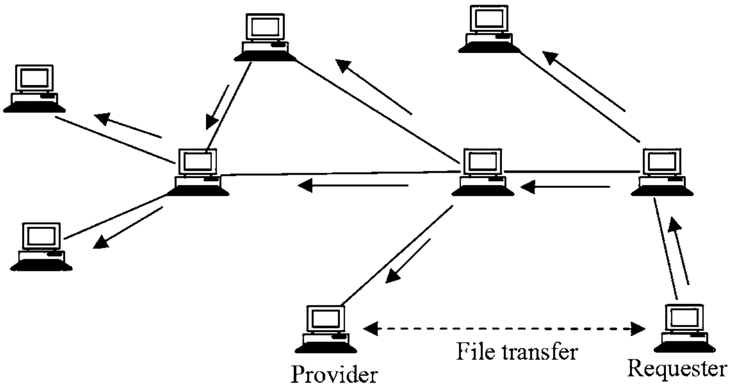


Fig. 4 Query propagation over the Gnutella network

forwards the query to all of its neighbours. This process continues until TTL decreases to zero.

The Gnutella network does not rely on a central server to index files, which avoids the single-point-of-failure issue and the performance bottleneck at the server side. Instead, many peer nodes are visited by flooding queries to see whether they have a requested file. The obvious drawback of Gnutella is that it generates potentially huge network traffic by flooding queries.

- **Freenet**

Freenet is a decentralised P2P data storage system designed to provide electronic document exchange through strong anonymity. In contrast to Gnutella, Freenet acts as a P2P storage system by enabling users to share unused local storage space for popular file replication and caching. The stored information is encrypted and replicated across the participating computers.

In Freenet, a file is shared with an ID generated from the hash value of the name and description of the file. Each peer node forms a dynamic routing table to avoid network flooding. A routing table includes a set of other peers associated with the keys they are expected to hold. To search a required file, the query is forwarded to the peer node holding the nearest key to the key requested. If the query is successful, the reply is passed back along the route the query comes in through. Each peer node that forwards the request will cache the reply and update the routing table by a new entry associating the data source with the requested key.

3.3 Hybrid Peer-to-Peer Networks

To avoid the observed problems of the centralised and decentralised P2P networks discussed above, hybrid P2P networks are emerging recently to provide trade-off solutions with a hierarchical architecture.

- **Kazaa**

Kazaa reorganises peer nodes into a two-level hierarchy with supernodes and leaves. Supernodes are capable and reliable peer nodes that take more responsibility for providing services in the network. A supernode is a temporary index server for other peer nodes. The peer nodes with high computing power and fast network connection automatically become supernodes.

Similarly to the bootstrapping method used in the Gnutella network, a newly joined node will attempt to contact an active supernode from a list of supernodes offered by Kazaa client software. The newly joined node will send a list of files it shares to the connected supernodes and further retrieve more active supernodes from the connected supernodes for future connection attempts.

In Kazaa, each leave node begins a lookup by sending a lookup request to its connected supernode as shown in Fig. 5. The supernode not only checks the local index for the file requested, but also communicates with other supernodes for a list of addresses of peer nodes sharing the files. When a supernode discovers the requested file from its local index, it will respond to the original supernode. The file is transferred directly between the query originator and the target peer node that shares the file as shown in Fig. 5.

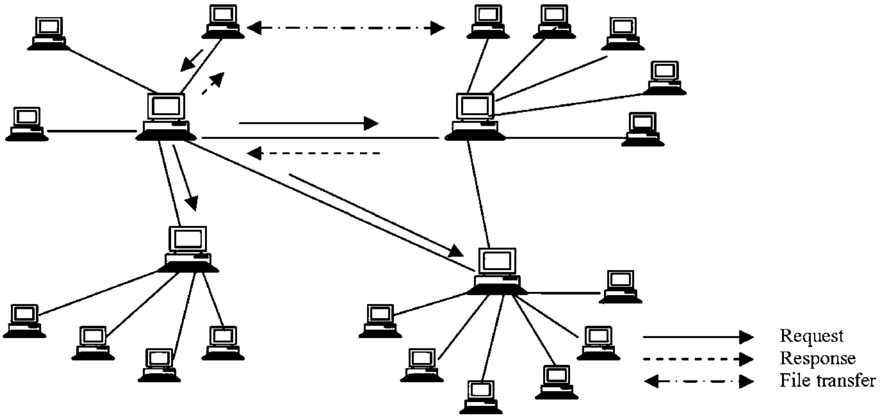


Fig. 5 Example of Kazaa network

- **Gnutella2**

Similarly to Kazaa, the peer nodes in the Gnutella2 network are classified into two categories: hubs and leaves. A leaf keeps only one or two connections to hubs. A hub acts as a proxy to the Gnutella2 network for the leaves connected to it. Queries are propagated among the hubs and only forwarded to a leaf if a hub believes it can answer the query.

- **JXTA**

  JXTA is an open source P2P platform developed by Sun Micro-systems. The JXTA Application Programming Interface (API) hides many programming details, which makes a JXTA application writing much easier than developing a P2P application from scratch.

  Similarly to Kazaa and Gnutella2, JXTA maintains a hierarchical network structure with rendezvous peers and edge peers. Different from Kazaa, the rendezvous peers in the JXTA network call the Shared Resource Distributed Index (SRDI) service to distribute indices to other rendezvous peers in the network. When a peer node searches for a file, it will send the query to the connected rendezvous peer and also multicast the query to other peers on the same subnet. If the rendezvous peer finds the information about the requested resources on its local cache, it will notify the peers that publish the resources and these peers will respond directly to the query originator. If the rendezvous peer cannot find the requested information locally, a default algorithm is used to go through a set of rendezvous peers for a rendezvous peer that caches the requested information [9].

As discussed above, hybrid P2P networks combine the techniques of both the centralised Napster and the decentralised Gnutella. However, since only a limited number of peer nodes are responsible for the query processing and routing, existing hybrid P2P networks still have the capability bottlenecks of the supernodes, which are also vulnerable to planned attacks [10].

4 Peer-to-Peer Search Systems

Since resource and service discovery in Grids involves a lot of elements in common with resource discovery in P2P networks, P2P search approaches are applicable for service discovery in large-scale Grid systems, which could help to ensure Grid scalability [11]. In this section, existing P2P search systems are investigated by classifying them into two broad categories: structured and unstructured P2P systems.

4.1 Structured P2P Systems

Structured P2P systems have a dedicated network structure on the overlay network which establishes a link between stored content and the IP address of a node. Distributed Hash Tables (DHTs) are widely used for resource discovery in the structured P2P systems like Chord [12], ROME [13], Pastry [14], CAN [15], and Kademlia [16].

In DHT-based P2P systems, each file is associated with a key generated by hashing the file name or content. Each peer node in these systems is responsible for storing a certain range of keys. The network structure is sorted by routing tables (or finger tables) stored on individual peer nodes. Each peer node only needs a small amount of "routing" information about other nodes (e.g. nodes' addresses and the range of keys the node is responsible for). With routing tables and uniform hash functions, peer nodes can conveniently put and get files to and from other peer nodes according to the keys of files.

- **Chord**

Chord [12] is a well-known DHT-based distributed protocol aimed to efficiently locate the peer node that stores a particular data item. Peer nodes are arranged in a ring that keeps the keys ranging from zero to $2^m - 1$. A consistent hashing is used to assign items to nodes, which provides load balancing and only requires a small number of keys to move when nodes join or leave the network [12]. The consistent hash function assigns each node and each key an ID using SHA-1.

In Chord, each peer node maintains a finger table pointing to $O(\log N)$ other nodes on the ring. Given a ring with 2^m peer nodes, a finger table has a maximum of m entries. The Chord routing algorithm utilizes the information stored in the finger table of each node to direct query propagation. For example, a node sends a query for a given key k to the closest predecessor of k on the Chord ring according to its finger table, and then asks the predecessor for the node it knows whose ID is the closest to k. By repeating this process, the algorithm can find the peer nodes with IDs closer and closer to k. A lookup only requires $O(logN)$ messages in a N-node Chord network and $\frac{1}{2}\log_2 N$ hops on average [12].

Unlike some other P2P models (e.g. Gnutella and JXTA) that provide a set of protocols to support P2P applications, Chord provides support for just one operation: given a key, it maps the key onto a node. In Chord, peer nodes are automatically allowed to participate in the network using the standard Chord protocol, no matter whether the nodes are useful and capable or not. Chord needs monitoring and selection functions in order to support and optimise its deployment over the real networks.

- **ROME**

ROME (Reactive Overlay Monitoring and Expansion) [13, 17] is an additional layer built upon the standard Chord protocol allowing control over the size of the network overlay via the selection and placement of peer nodes on the Chord ring.

Figure [18] shows the ROME architecture running on the top of Chord. ROME includes a set of processes (ellipses) and data structures (rectangles) [18]. Processes are comprised of a traffic analyser to monitor network traffic without changing the underlying Chord protocol, and operations to add, replace and remove nodes from the ring. Data structures of ROME store monitoring data, a copy of Chord data and ROME specific data (e.g. bootstrap server's address). ROME can provide an optimal size of Chord ring by monitoring the workload on each node to solve the problems of under-load or over-load nodes by adding, replacing and removing nodes. ROME provides more efficient and fault-tolerant resource discovery with message cost saving than the standard Chord [18].

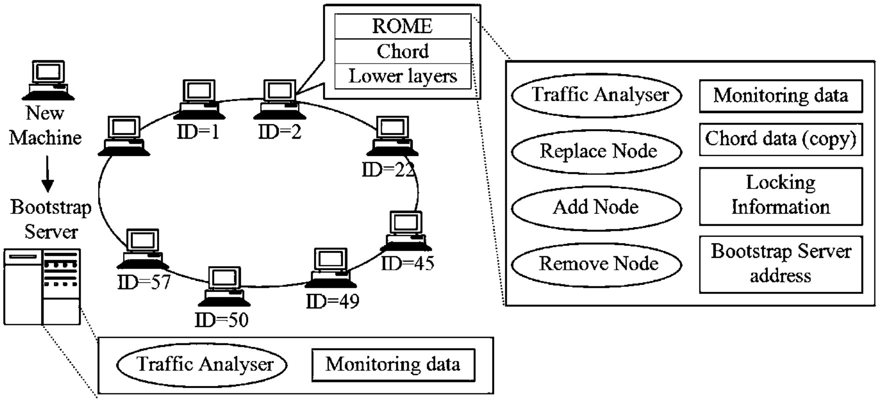


Fig. 6 ROME architecture

- **Accordion**

Accordion [19] is a DHT protocol extended from Chord, which bounds its communication overhead according to a user specified bandwidth budget. Accordion borrows Chord's protocols for maintaining a linked list in which the ID space is organized as a ring as in Chord. Different from Chord using a fixed routing table, Accordion can automatically adapt itself to achieve the best lookup latency across a wide range of network sizes and churn rates. Accordion maintains a large routing table when the system is small and relatively stable. When the system grows too large or suffers from high churns, Accordion shrinks its routing table on each peer node for lower communication overhead. By remaining flexible in the choice of routing table size, Accordion can operate efficiently in a wide range of operating environments.

- **Pastry**

Pastry [14] is a prefix-based routing system using a proximity metric. Similarly to Chord, Pastry organizes peer nodes in a 128-bit circular node ID space. At each step, a query message is forwarded to a numerically closer node to a given key. In a network consisting of N nodes, the message can be routed to the numerically closest node within $\log_{2^b} N$ hops, where b is a configurable parameter.

In contrast to Chord which uses one-dimensional tables, a node's routing table is organized into two dimensions with $\log_{2^b} N$ rows and $2^b - 1$ entries per row. Each entry in row n of the routing table points to a node whose node ID shares the present node's ID in the first n digits, but whose $(n+1)th$ digit is different from the $(n+1)th$ digit in the present node's ID. The routing procedure involves two main steps. Given a message, the node first checks whether the key is within the range of its leaf set. If so, the message is sent directly to the destination node. Otherwise, the message is forwarded to the node that shares a common prefix with the key by at least one more digit.

- **CAN**
  Content-Addressable Network (CAN) [15] generalizes the DHT methods used in Chord and Pastry. A CAN identifier space can be looked at as a d-dimensional version of Chord (which is one-dimensional) and Pastry (which is two-dimensional) identifier space. For a d-dimensional space partitioned into n equal zones, each node maintains $2d$ neighbours and the average routing path length is $\frac{1}{4}(n^{\frac{1}{d}})$. Higher dimensions in the identifier space reduce the number of routing hops and only slightly increase the size of routing table saved in each node. Meanwhile, fault tolerance of routing is improved by higher dimensionality of CAN, since each node has a larger set of neighbours to select as alternatives to a failed node.

- **Kademlia**
  Kademlia [16] not only uses the basic DHT methods (e.g. unique ID, routing table with $< key, value >$ pairs), but also provides a number of desirable features superseding other previous DHTs (e.g. Chord, CAN and Pastry).

  Kademlia is based on the calculation of the "distance" between two nodes on the overlay with an XOR metric. Each Kademlia node stores contact information about other peer nodes in the local routing tables. When a Kademlia node receives a message from another node, it will update the appropriate entry for the sender's node ID. Thus, the peer nodes which issue or reply to a large number of queries will become widely known, which enables more capable nodes to take on more workload in the network.

  To lookup a specific key, the query originator searches the local routing tables for α nodes with the closest distance to the query originator and then contacts them in parallel. Each recipient node replies with the information about the peer node which is closer to the key. The query originator resends the lookup to nodes it has learned about from previous RPCs.

4.2 Unstructured P2P Systems

In contrast to structured P2P systems, unstructured P2P systems do not maintain network structure, where address and content stored on a given peer node are unrelated. Although existing search methods in unstructured P2P systems are heterogeneous, most of them are dedicated to solving observed issues of flooding mechanisms which can be classified into two broad categories: blind search and informed search, according to whether the algorithm needs additional indices about the locations of resources.

4.2.1 Blind Search

In a network with a blind search method, peer nodes do not maintain additional information about resource locations. The advantages of blind search methods are that they do not need any communication overhead to maintain the additional indices about resource locations and are extremely resilient in the highly dynamic networks. Flooding is the fundamental approach of blind search where queries are forwarded to all connected peer node to see whether they have a requested file. In order to solve the massive traffic problem caused by flooding, several improved search methods have been presented recently [20–22].

Random walker [20, 21] is a well-known blind search method, which enables peer nodes to forward a query to a randomly chosen neighbour rather than broadcast the query to all of their neighbours. Each neighbour repeats this process until the required file is discovered. The random walkers can find targets more efficiently while significantly reducing the traffic compared to Gnutella's flooding method [20]. In order to increase the probability of resource discovery, pro-active replications are used to increase the density of copies of each object. The study in [20] shows that the square-root replication distribution is optimal in terms of minimizing overall search traffic, which replicates files in proportion to the square-root of their query probability.

Yang and Garcia-Molina [22] presented an iterative deepening technique for resource discovery in unstructured P2P networks. Iterative deepening enables query originators to use successive BFS (Breadth-First Search) queries with increasing depths, until the request is satisfied or the maximum depth is reached.

4.2.2 Informed Search

In contrast to blind search, informed search methods enable peer nodes to maintain additional information about other peer nodes in the network, e.g. network topology and resource locations. Existing informed search solutions can be classified in two groups: mechanisms with specialised index nodes and mechanisms with indices at each node.

The search mechanisms with specialised index nodes have been used in current P2P applications. For example, Napster employs a centralized server to maintain such additional information. Kazaa and JXTA utilize a set of semi-centralized supernodes to maintain the extra information about their leaves and other supernodes.

However, systems with specialised index nodes are vulnerable to attack by centralizing indices in a small subset of peer nodes. New methods have emerged in recent years by distributing indices to each individual peer node in the network. Such methods can be further classified into the following three approaches according to their design principles.

- **Topology optimisation**

  In many P2P applications, topology significantly affects the performance of resource discovery. The first approach intends to reduce search cost by adapting and optimising the overlay topology of network. Each peer node is expected to keep topology information about its neighbours or neighbours' neighbours.

  A topology optimisation method used in Gia [23] puts most of the peer nodes in the network within a short reach of high capacity nodes, which leads to the situation that the high capacity nodes are also the nodes with a high node degree of connectivity (a large number of links). This protocol ensures that the well-connected nodes which receive a large proportion of queries actually have the capability to handle these queries.

Gia enables each peer node to connect high capacity nodes as neighbours. Alternatively, the studies in [24, 25] present topology optimisation methods by preferentially selecting low-cost and low-latency connections. These methods address the topology mismatch problem between P2P logical overlay networks and physical under-lying networks (which incurs a large volume of redundant traffic on P2P networks) by deleting inefficient overlay links and adding efficient ones.

- **Statistical information about neighbours**

The second approach enables peer nodes to route queries to the neighbours that are likely to have the requested files in accordance with the maintained statistical information about neighbours.

Tsoumakos et al. [26] introduced an adaptive and bandwidth-efficient algorithm for searching in unstructured P2P networks, called Adaptive Probabilistic Search (APS). Different from the above topology optimisation methods, the search algorithm of APS is not allowed to alter the overlay topology. In APS, each peer node keeps an index describing which objects were requested by each neighbour with a success ratio of previous searches. The probability of choosing a neighbour to find a particular file is dependent on the success ratio. In APS, searching is based on the simultaneous deployment of k walkers and probabilistic forwarding. The query originator sends queries to k of its N neighbours. If each of these nodes can find a matched object in its local repository, the walker terminates, otherwise it gets forwarded to one of neighbours. The procedure continues until all k walkers have completed. The update takes a reverse path back to the query originator to adjust probability accordingly either with success or failure. According to the results shown in [26], APS can discover many more objects with much higher success rates than random walkers [20, 21] algorithm.

Adamic et al. [21] showed that many peer nodes can be reached by forwarding queries to peer nodes with the highest degree (number of links) in the neighbourhood and thus it can get many results back. Yang and Garcia-Molina [22] tested this model by forwarding queries to the peer node which had the largest number of neighbours. Yang and Garcia-Molina also designed and simulated several other heuristics to help in selecting the best peer nodes to send a query, for example, the peer node that returned the highest number of results from previous searches or the peer node that had the shortest latency.

The study in [27] evaluated a similar heuristic that the peer nodes with more files are more likely to be able to answer the query, called Most File Shared on Neighbourhood (MFSN) which forwards the query to the neighbouring node sharing the largest number of files. In accordance with these previous studies above, these heuristic-based methods can generally achieve better performance than random walkers.

In contrast to the above studies using only one heuristic parameter, DSearch method in [28] puts two heuristics in consideration. In order to achieve a high success rate and efficiency, queries are forwarded to the "good" nodes that are more likely to have the requested files or can find the requested files with a high probability in future hops. Two heuristics are considered to determine the "goodness" of a neighbour: the number of shared files on the neighbour and the expected number of accessible files in future hops via the neighbour. DSearch utilizes high-degree nodes to find a wider range of accessible nodes, while using the neighbouring nodes that shares a large number of files to discover the requested files. From the simulation results, DSearch achieves a better performance when compared to the methods with only one heuristic.

The principle behind these heuristic-based methods is to forward queries to more capable peer nodes, for example, the peer nodes with the highest success rate of searches, the peer nodes with the highest degree, or the peer nodes with most shared files. However, such capable nodes may be over-loaded and become the victims of their own success. Traffic unbalance is a significant limitation to these methods.

- **Cached semantic information**

In contrast to the second approach of maintaining simple statistical information, the third approach enables each peer node to keep a routing index which contains detailed semantic information about content of shared files. This information can be collected by exchanging indices regularly with other peer nodes or caching the historic record regarding the results of previous queries.

Routing Indices

Routing Indices (RI) [29] enable peer nodes to create query routing tables by hashing file keywords and regularly exchanging those with their neighbours. Peer nodes normally maintain additional indices of files offered by their overlay neighbours and neighbours' neighbours. In the RI system, shared documents are classified into different categories of topics. Each peer node maintains the RI indicating how many documents of which categories could be found through that neighbour. When a new connection is established in the RI system, two peer nodes will aggregate and exchange their own RI to each other. In order to keep data in RI up-to-date, peer nodes will notify neighbours about the changes in the local RI caused by addition/removal of shared documents or joining/leaving of other neighbours.

When a peer node receives a query, the peer node needs to compute the goodness of each neighbour for a query. The number of documents that may be found in a path is used as a measure of goodness. If more documents are found through a particular neighbour, this neighbour will be selected to forward a query.

Sripanidkulchai's model

The idea of constructing "friend lists" has also been used in the Sripanidkulchai's model [30], which presents a content location solution in which peer nodes loosely organize themselves into an interest-based structure. When a node joins the system, it first searches the network by flooding to locate content. The lookup returns a set of nodes that store the content. These nodes are potential candidates to be added to a "shortcut list ". One node is selected at random from the set and added. Subsequent queries will go through the nodes in the shortcut list. If a peer node cannot find content from the list, it will generate a lookup with Gnutella protocol. From their results, a significant amount of flooding can be avoided which makes Gnutella a more competitive solution.

NeuroGrid

NeuroGrid [31] is an adaptive decentralized search system. NeuroGrid utilizes the historic record of previous searches to help peer nodes make routing decisions. In NeuroGrid, peer nodes support distributed searches through semantic routing by maintaining routing tables at each node [31]. In the local routing tables, each peer node is associated with keywords regarding the content it stores. When a peer node forwards a query, it will search for the peer nodes that are associated with the query keywords. In each hop, NeuroGrid intends to find M matched peer nodes and then forwards the query to these peer nodes. If there are not M matches found, the algorithm will randomly select peer nodes in the routing table until M peer nodes are selected.

In NeuroGrid, users' responses to search results are stored and used to update the meta-data describing the content of remote peer nodes. NeuroGrid can learn the results from previous searches to make future searches more focused and semantically routes queries according to the cached knowledge. However, NeuroGrid is only effective for previously queried keywords and not suitable for networks where peer nodes come and go rapidly [32].

5 Future Trends

Some potential future trends are outlined in this section.

5.1 Self-organising Systems

P2P is beneficial when removing a centralised server. On the other hand, new mechanisms are required to compensate for the server, especially for resource discovery and network maintenance. However, owing to the lack of a centralised server, any attempts of additional control could be difficult to achieve in the distributed P2P architecture. In contrast, self-organisation could be a good way to solve the control issues in the decentralised P2P architecture. Self-organisation is a process where the organisation of a system spontaneously increases without being managed by an outside source.

Human society is a self-organising system. Social networks are formed naturally by daily social interactions. A social community is a group of people with common interests, goals or responsibilities which is formed spontaneously. By using social networks, people can find some acquaintances that potentially have knowledge about the resources they are looking for.

Similarly to social networks, where people are connected by their social relationships, two autonomous peer nodes can be connected if users in those nodes are interested in each other's data. If peer nodes can self-organise themselves like social networks, a large communication cost for peer group construction, maintenance and discovery can be saved, which will significantly improve overall performance of P2P networks.

5.2 Hybrid Systems

Since large-scale resource sharing is one goal of Grids, P2P networks and Grid systems are starting to converge to a common structure, leading to application of P2P techniques to Grid systems [33]. Some research has been done to try connecting Grid Services and P2P networks (e.g., [4, 31, 32]). Unfortunately, there is still not a good solution to seamlessly pull the two domains together. There are many boundaries to connect Grids and P2P networks because of different architectures, standards and protocols between them. Current Grid service standards need to be extended, especially for service descriptions and annotations, by including additional attributes for peer's specifications.

6 Conclusions

In this chapter, the evolution of the network architecture has been investigated from client-server to P2P networking. P2P is beneficial when removing a centralised server. On the other hand, new mechanisms are required to compensate for the server. The main advantage of P2P architecture lies in its good scalability, agility, resilience and availability. On the contrary, its major challenges lie in its efficiency, dependability and security.

To address these challenges, hybrid systems combining the techniques of both Grid and P2P computing could be potential solutions for the design of next generation distributed systems. By introducing P2P techniques into Grid computing, Grid systems could be more scalable and resilient, removing the sing-point-of-failure. With the cooperation of Grid computing environments, the usage of P2P networks could be also broadened from simple file provision to more advanced services, such as sharing redundant computing power for complicated scientific calculation and sharing extra bandwidth for real-time video transmission.

References

1. Vance, A.: Google goes gaga for Opteron. Available from: `http://www.theregister.co.uk/2006/03/04/goog_opteron_sun/`
2. Foster, I., Kesselman, C. (eds.): The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1999)

3. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications 15 (2001) 200–222
4. Qu, C., Nejdl, W.: Interacting the Edutella/JXTA Peer-to-Peer Network with Web Services. Second International Conference on Peer-to-Peer Computing (2002)
5. Zhao, S., Stutzbach, D., Rejaie, R.: Characterizing Files in the Modern Gnutella Network: a Measurement Study. SPIE/ACM. Multimedia Computing and Networking, San Jose, CA (2006)
6. Cachelogic.: Trends and Statistics in Peer-to-Peer. (2005) Available from: `http://creativecommons.nl/nieuws/wp-content/uploads/2006/04/CacheLogic_AmsterdamWorkshop_Presentation_v1.0.ppt.`
7. Oberholzer, F., Strumpf, K.: The Effect of File Sharing on Record Sales: An Empirical Analysis. Journal of Political Economy 115 (2006) 1–42
8. Risson, J., Moors, T.: Survey of Research towards Robust Peer-to-Peer Networks: Search Methods. Computer Networks 50 (2006) 3485–3521
9. Traversat, B., Abdelaziz, M., Pouyoul, E.: Project JXTA: A Loosely-Consistent DHT Rendezvous Walker. Sun Microsystems, Inc (2003)
10. Lo, V., Zhou, D., Liu, Y., GauthierDickey, C., Li, J.: Scalable Supernode Selection in Peer-to-Peer Overlay Networks. Second International Workshop on Hot Topics in Peer-to-Peer Systems, La Jolla, California (2005)
11. Mastroianni, C., Talia, D., Verta, O.: A P2P Approach for Membership Management and Resource Discovery in Grids. 2005 International Symposium on Information Technology: Coding and Computing, Las Vegas, NV (2005)
12. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. ACM SIGCOMM, San Diego, CA (2001)
13. Salter, J., Antonopoulos, N.: ROME: Optimising DHT-based Peer-to-Peer Networks. 5th International Network Conference, Samos, Greece (2005)
14. Rowstron, A., Druschel, P.: Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems. IFIP/ACM International Conference on Distributed Systems Platforms, Heidelberg, Germany (2001)
15. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. ACM SIGCOMM, San Diego, CA (2001)
16. Maymounkov, P., Mazi‘eres, D.: Kademlia: A Peer to Peer Information System Based on the XOR Metric. Internation Workshop on Peer-to-Peer Systems, Cambridge, MA (2002)
17. Antonopoulos, N., Exarchakos, G.: G-ROME: A Semantic Driven Model for Capacity Sharing Among P2P Networks. Journal of Internet Research 17 (2007) 7–20
18. Salter, J.: An Efficient Reactive Model for Resource Discovery in DHT-Based Peer-to-peer Networks. Department of Computing, Vol. PhD. University of Surrey, Guildford, Surrey (2006)
19. Li, J., Stribling, J., Morris, R., Kaashoek, M.F.: Bandwidth-Efficient Management of DHT Routing Tables. 2nd Symposium on Networked Systems Design and Implementation, Boston, MA (2005)
20. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and Replication in Unstructured Peer-to-Peer Networks. ACM SIGMETRICS, Marina Del Rey, CA (2002)
21. Adamic, L.A., Lukose, R.M., Puniyani, A.R., Huberman, B.A.: Search in Power Law Networks. Physical Review 64 (2001) 046135-046131–046135-046138
22. Yang, B., Garcia-Molina, H.: Efficient Search in Peer-to-Peer Networks. International Conference on Distributed Computing Systems, Vienna, Austria (2002)
23. Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S.: Making Gnutella-Like P2P System Scalable. ACM SIGCOMM, Karlsruhe, Germany (2003)
24. Xiao, L., Liu, Y., Ni, L.M.: Improving Unstructured Peer-to-Peer Systems by Adaptive Connection Establishment. IEEE Transactions on Computers 54 (2005) 176–184
25. Liu, Y., Xiao, L., Liu, X., Ni, L.M., Zhang, X.: Location Awareness in Unstructured Peer-to-Peer Systems. IEEE Transactions on Parallel and Distributed Systems 16 (2005) 163–174

26. Tsoumakos, D., Roussopoulos, N.: Adaptive Probabilistic Search for Peer-to-Peer Networks. Third International Conference on Peer-to-Peer Computing, Linkoping, Sweden (2003)
27. Li, X., Wu, J.: A Hybrid Searching Scheme in Unstructured P2P Networks. International Conference on Parallel Processing, Oslo, Norway (2005)
28. Liu, L., Antonopoulos, N., Mackin, S.: Directed Information Search and Retrieval over Unstructured Peer-to-Peer Networks. the International Computer Engineering Conference, Cairo, Egypt (2006)
29. Crespo, A., Garcia-Molina, H.: Routing Indices for Peer-to-Peer Systems. International Conference on Distributed Computing Systems, Vienna, Austria (2002)
30. Sripanidkulchai, K., Maggs, B., Zhang, H.: Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. IEEE Infocom, San Francisco (2003)
31. Joseph, S.: NeuroGrid: Semantically Routing Queries in Peer-to-Peer Networks. International Workshop on Peer-to-Peer Computing, Pisa, Italy (2002)
32. Joseph, S.: P2P MetaData Search Layers. International Workshop on Agents and Peer-to-Peer Computing, Melbourne, Australia (2003)
33. Marzolla, M., Mordacchini, M., Orlando, S.: Peer-to-peer Systems for Discovering Resources in a Dynamic Gridstar, Open. Parallel Computing 33 (2007) 339–358
34. Wang, M., Fox, G., Pallickara, S.: A Demonstration of Collaborative Web Services and Peer-to-Peer Grids. International Conference on Information Technology: Coding and Computing (2004)
35. Prasad, V., Lee, Y.: A Scalable Infrastructure for Peer-to-Peer Networks Using Web Service Registries and Intelligent Peer Locators. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (2003)

Examining the Use of Peer-to-Peer Networks from an Activity Perspective

Jorn De Boever and Dirk De Grooff

Abstract Since the introduction of Napster in 1999, millions of Internet users have exchanged massive amounts of files via P2P (Peer-to-Peer) file-sharing networks. Notwithstanding the widespread penetration of these systems among Internet consumers, little is known about the usage process. Therefore, the aim of this chapter is to examine the usage of "illegal" P2P networks by means of an exploratory, qualitative study. The main findings revealed significant differences between the uses of various systems. Bittorrent clients were mainly used to download large files such as video, movies, and complete albums, while Gnutella clients were particularly utilized for small files such as single songs. The results indicate that the type of content, the characteristics of the client, the omnipresence of fake files and malware, the users' motivations, the users' lifestyles and the presence of bandwidth caps had an impact on how the participants utilized P2P systems.

Key words: Behavior, Activity theory, Motivations, Trust, Quality, Context

1 Introduction

It is widely recognized that end users play an important role in the success or failure of P2P networks as users have to share their resources and files with other peers. Therefore, it is remarkable that user studies – such as [17, 27] – are still underrepresented in the P2P technology research domain.

Jorn De Boever
Centre for UX Research / IBBT, K.U.Leuven, Parkstraat 45 Bus 3605 - 3000 Leuven - Belgium, e-mail: jorn.deboever@soc.kuleuven.be

Dirk De Grooff
Centre for UX Research / IBBT, K.U.Leuven, Parkstraat 45 Bus 3605 - 3000 Leuven - Belgium, e-mail: dirk.degrooff@soc.kuleuven.be

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_4, © Springer Science+Business Media, LLC 2010

Up to now, most research in this area has focused on what types of files users download from P2P systems, leaving questions open so as to what motivates users to employ these systems or the contexts in which such actions take place. Motivated by this gap in the literature, we will endeavor to examine the usage process of P2P networks, which can be summarized in the following key questions: what do individuals use P2P systems for, what motivates their usage, how do they use these different systems in order to gratify their needs, and what is the relationship between the motifs/objects, behavior and context. This study investigates these issues, with specific attention to comparing the use of bittorrent and Gnutella clients. These two systems were compared as we found that both types of P2P clients were used in different ways and for different purposes. It should be noted that bittorrent is written in lower-case letters as we refer to all bittorrent clients – e.g. Azureus, BitComet – which is not limited to the BitTorrent client.

This chapter has been organized in the following way. The first section depicts the theoretical framework that was used throughout the analysis. In the second section, we elaborate on the methods that were adopted to carry out this research. Subsequent sections describe and compare the usage process of two cases, i.e. the use of bittorrent and Gnutella clients.

2 Theoretical Framework: Activity Theory

This section describes activity theory as the theoretical framework for this study. First, we will outline the introduction of activity theory in Human-Computer Interaction (HCI) research. Next, we will discuss the importance of context in activity theory. Finally, we will explain several concepts of the activity system.

2.1 Activity Theory and Human-Computer Interaction

Activity theory has its foundations in Russian cultural-historical psychology, since it was developed by Vygotsky and Leontiev [4, 23, 24, 28]. Bødker [5, 6] introduced activity theory as a theoretical framework in HCI research which inspired many academics, e.g. [25, 31, 36]. Since then, Information Systems researchers have adopted this theory as well, e.g. [14].

During the 80s/90s, activity theory had been introduced in HCI research in response to some flaws of cognitive psychology which considered users to be passive information processing subjects whose actions seemed to be isolated from contextual influences. In contrast with this former cognitive psychology perspective, activity theory stresses the importance of considering technologies as mediating artifacts that should be examined within the meaningful context of use. However, this does not imply that activity theory rejects cognitive psychology, but that it should be regarded as an expansion of existing knowledge.

Gay and Hembrooke [19] consider the following aspects to be central to activity theory: "The emphasis on meaning through action, the connection between the individual and the social, and the role of mediating tools provide the kernel around which activity theory has developed" (p. 2). Activity, which puts a meaning on our actions, has been defined as follows: "Activity (...) is understood as a purposeful interaction of the subject with the world. (...) The very concept of activity includes its orientation toward an object, an object that both motivates and directs the activity" [23]. These quotes stress that the meaning of users' actions result from a meaningful interaction between the motifs/goals/intentions of users, the context and behavior.

Mediation is considered to be one of the most important concepts within activity theory, as users' purposeful interaction with the world is mediated by artifacts. The concept of mediation is not limited to physical artifacts, as for instance a language is considered to be an artifact as well [2, 21]. Users are looked upon as active, intentional subjects that act within specific contexts and utilize artifacts that mediate their actions with the environment [23]. For instance, P2P file-sharing networks should be considered as mediating artifacts with which users try to achieve certain goals and gratify specific needs. Although we focus on the use of P2P systems, these P2P networks are not the only mediating artifacts in the activity, as e.g.: bandwidth is needed to be able to exchange files; users must have e.g. media players or an iPod to consume the content; etc. [10].

It should be noted that activity theory is not an explanatory theory, but it provides conceptual tools that allow researchers to study the complex context in which artifacts are used so as to be able to reflect, compare and discuss results [31].

Besides the strengths of activity theory, it has some flaws as well. One of the major drawbacks of this theory is that it has not been made fully operational in HCI research [21]. Although several authors have attempted to generate methods based on activity theory, there is still much work to be done. For a comparative analysis of these methods we refer to [33].

Finally, we have to mention that, whereas most papers about activity theory in HCI are related to designing new systems and interfaces, we will focus on evaluation of the use of P2P systems as we want to examine existing practices of the use of P2P systems.

2.2 Importance of Context

Because of the unpredictable, circumstantial "situatedness" of every action, a statement only regarding the motivations or intentionality of users is insufficient, as it reveals little information about how purposeful action eventually occurs: "every course of action depends in essential ways upon its material and social circumstances. (...) The significance of actions, and their intelligibility, resides (...) in a contingently constructed relationship between observable behavior, embedding circumstances and intent" [35]. For this reason, although the actions are guided by initial motivations, users' behavior should be considered in their specific contexts. Therefore, Kaptelinin and Nardi [23] argue that HCI research should take "the meaningful context of a subject's interaction with the world" (p. 34) as the central point of interest and that the activity system should be studied in real situations instead of in artificial environments with artificial tasks. In other words, the use of artifacts – in this case, P2P systems – should be studied in their real-life context.

This reasoning implies that the question – how people use P2P file-sharing systems – will depend in important ways on contextual factors such as the presence of fake or corrupted files, the fact that file-sharing is often illegal, the availability of bandwidth, the lifestyle of users, etc. As we will illustrate in the following paragraphs, reviewing the literature is important to gain a first insight in different context aspects that might be of significance.

A recent study points out that media usage patterns are to a large extent being influenced by users' lifestyle [10]. Younger people, aged 18 to 34, often have the following characteristics:

- They have irregular schedules.
- They possess different kinds of media: TV, DVD-players, computers, MP3 player/iPod, mobile phones, video game consoles, etc.
- They expect unrestricted accessibility and portability of content.

In addition, Condry [12] studied the culture of file-sharing in the US and Japan, and found that people retrieve files from P2P networks because: it is free of charge, they are dissatisfied with the medium CD (consumers have to pay the entire album whereas they mostly like only a few songs), it is a kind of protest against the powerful media industry, it allows them to control the files themselves.

LaRose et al. [26] reminds us that many students love to listen to music, have a lot of leisure time, but they often dispose of a limited budget. These authors examined several aspects that might determine the levels of file-sharing among college students. They found for instance that that the expectation of obtaining new and rare content motivated the usage of P2P systems in important ways. The chances of downloading content of poor quality and the moral unacceptability of illegal downloading discouraged the use of P2P networks. The fear of downloading corrupted content is a reasonable fear as different studies demonstrated the presence of malware [20, 34] and fake content [11].

Several studies have been conducted to analyze users' motivations to utilize P2P networks [9, 15]. These studies have found that people use P2P file-sharing systems for instance because they want to: have control over the content, have instant gratification, find rare or unreleased content, sample content before buying it, promote the content they like and save money.

2.3 Activity System

Applications, such as P2P systems, are cultural artifacts that mediate certain actions in order to obtain a certain object. The link between the subject and the object is bridged by one or several mediating components. The importance of mediating artifacts is especially crucial for HCI studies as computers, applications and their interfaces are tools that mediate the relation of people with the environment [19, 25].

The subject, who might consist of an individual or a group, utilizes mediating artifacts to reach a certain object. The transformation of an object into certain outcomes motivates the activities [25]. The following example illustrates this model: a user (subject) will adopt a P2P client (mediating artifact) that allows him to download music (object) which he wants to consume to make working on his computer a more pleasant experience (desired outcome). This model represents an activity system at an individual level.

The model, mentioned in the previous paragraph, was extended by Engeström [16] who had primarily work environments in mind when developing his model. In the previous model, which represents the activity system at an individual level, the relation between the subject and object was mediated by artifacts, whereas in the activity system of Engeström, the subject-object relationship was enlarged with a third component, i.e. community. This led to two additional relationships: (1) the relationship between the subject and the community which is mediated by rules and (2) the relationship between the community and the object which is mediated by the division of labor. We argue that Engeström's model has been build from the perspective of a work setting where people have to obtain a collective objective, which is the reason why we do not adopt this model as the use of P2P systems primarily turned out to be an individual practice where peers cannot interact in a direct way.

In Leontiev's activity theory approach, the structure of an activity consists of three layers, which includes activities, actions and operations [4, 23, 24]. This hierarchy is displayed in Table 1. Each *activity* has a certain, often unconscious, object, which users try to attain. Kaptelinin [22] suggests that an object might be considered as a collection or a set of motives. In addition, activities do not take place in isolation, but they are part of a web of activities that strive for the same or connected objects which put a meaning on users' behavior [7].

Table 1 Hierarchy of an activity system

Hierarchy	**Orientation**	**Characteristics**
Activity – Why?	Object/set of motives	Individual/collective
Actions – What?	Goals	Individual, conscious
Operations – How?	Conditions	Individual, unconscious

Next, an activity is performed by a number of *actions* which are guided by specific conscious goals. The execution of all the actions might result in the completion of the activity or the attainment of the motive. Finally, the actions in their turn are carried out by several unconscious, routine *operations* which are oriented toward the conditions of a specific context.

This hierarchy should not be considered as a static entity in that for instance actions can evolve into operations and vice versa. The concepts activity, actions and operations can be made operational by asking respectively the following questions: "why something takes place? what takes place? and how is it carried out" [7].

In practice, the distinction between actions and operations is difficult to retrieve from analysis, which results from the fact that during observations it is practically impossible to discover what is (not) automated as we ask the respondents to explain what they are doing. For this reason, we will make no distinction between actions and operations. By letting users think out loud, we turn their operations into actions as these users have to reflect on their behavior. This movement from operations to actions by reflection is called conceptualization [4–6]. The other way around, actions can turn into operations by a process of automation.

3 Methods

Activity theory does not suggest specific methods for examination as this should depend on the research object/questions [31]. In our research of the use of P2P systems, we have therefore combined three methods that served best to gather the right data in order to answer our research questions. In chronological order, these methods included: (1) incident diaries or logbooks, (2) observations and (3) semi-structured in-depth interviews.

We sampled 25 participants (aged 15–35 years) that utilized existing P2P file-sharing software such as BitComet, Azureus, Limewire, KaZaA, etc. We have chosen this age category for several reasons. First, several studies demonstrated that the penetration of P2P network usage is the largest in this age group [1, 18]. Second, as we are conducting qualitative research, we assumed that elder age categories would use P2P systems differently, with other motivations and needs in comparison with younger users. Because qualitative inquiries only allow gathering information from small samples, diverging the sample further would probably harm the validity of this research.

Before the commencement of the actual research, users that were willing to participate had to fill in an online registration form in which general context information was gathered, e.g.: which P2P system was used, frequency of use, personal information, etc. This information allowed us to recruit a representative sample of users for our research. After the recruitment phase, the selected users were asked to register their behavior on P2P networks by means of an online incident diary or logbook. The participants had to complete the online incident logbook each time they used the P2P system, without changing their natural behavior. In this sense, the participants were observers themselves as they had to register their own behavior. The diary method has proven to be especially useful in situations where users do not utilize an application frequently [13, 37]. For instance, most people utilize P2P systems only a couple of times a week or month, which makes it difficult for researchers to observe users' behavior. Another reason to use incident diaries is the problem that users have difficulties in recalling their actions which might cause inaccurate and unreliable results [29]. In other words, diaries as an observational technique decrease the danger of biased anticipatory self-reporting from the respondents. Furthermore, incident logbooks allow us to gather information without interrupting the natural behavior of the participants. The information from this diary method allowed us to create a profile of each person's use of his/her P2P system [30]. The information gathered via the incident logbook related for instance to: (1) how users came across certain content, (2) which files they downloaded/uploaded/injected, (3) which problems and experiences they had while using P2P networks and (4) what users were intending to do with the files after they had downloaded them.

Next, after having completed their incident logbooks several times, the participants were contacted for an observation/in-depth interview. The respondents were asked to contact us the next time they wanted to download a session in order to allow us to observe them while they were using their P2P software in their natural environment. We decided to observe the participants as observations have already proven to be "(...) excellent for collecting rich, detailed data and for obtaining a holistic view of the process or domain" [13]. During the observations, the participants were asked to tell us what they were doing and why they were using the application in such a way. By letting participants think aloud, we were able to turn unconscious operations into conscious actions, i.e. conceptualization, which allowed us to gain a more elaborate understanding of the users' behavior. These observations were videotaped, by focusing a video camera on the participant's screen, so as to be able to analyze the observations afterwards. These observations allowed us to gain insight in the process of using P2P file-sharing systems: the sequence of actions, the artifacts used to support the actions, the different focuses, etc.

Further, information, collected via incident logbooks and observations, were unraveled in semi-structured in-depth interviews. Additionally, during the interview, data was gathered as well which cannot be collected during observations or incident diaries, e.g. data regarding motivations. The value of interview data has been questioned in the HCI domain in terms of its reliability as it is often claimed that users cannot accurately tell what they have been doing. Although we agree with this view, we argue that interview data are quite reliable when it comes to gathering data about motives and goals. This view is supported by Nardi [32] who posed that although the use of interviews is problematic on the level of operations, it still has a reliable function on the level of actions and objects. For the previously mentioned reasons, we argue that the diary-observation-interview approach provided the most appropriate combination of methods to explore P2P network usage in its context.

Finally, the data analysis was performed using NVivo [3]. First, the information was coded in a detailed way so as to be able to distinguish different categories or themes. Next, a tree of codes was generated where some of the codes were merged and others split into new codes. Finally, we explored which of the variables were associated with each other. The results of this analysis are treated in the following sections. Based on this analysis, we were able to explore the process of using two different types of file-sharing systems, namely bittorrent and Gnutella clients.

4 Understanding the Use of P2P Networks

The aim of this chapter is to examine the following research questions: what do individuals use P2P networks for (object), what motivates the participants' usage of these systems (motivations), how do they use these file-sharing networks in their environment (behavior), and what is the relationship between motives/objects, behavior and context. In this section we will present the results of two cases, i.e. the use of bittorrent and the use of Gnutella clients. The findings will be illustrated by some quotes from the participants. In these quotes, we use for instance "R 22" to refer to the participant with that number. Each participant is named with a number to guarantee anonymity. In addition, "I" refers to the interviewer.

4.1 Process of Using Bittorrent Clients

Before continuing, some bittorrent terminology will be explained first to help readers, unfamiliar with bittorrent related terms, better understand this section.

Users that want to download content via bittorrent first have to download a *torrent file* from a website (i.e. a torrent site). A torrent file is a meta-data file that contains information concerning the location of the tracker of specific content. A *tracker* keeps track of which peers have or download which parts of the desired file. By loading a torrent file in their bittorrent client, a tracker is contacted which gives the bittorrent client all the information needed to start downloading the desired file.

The bittorrent case is organized as follows. The first section will present the different levels of motivations, and the subsequent section will deal with the usage process of bittorrent clients.

4.1.1 Different Levels of Motivations

In our analysis, we found different levels of motivations related to different aspects of the use of P2P file-sharing systems: the object, the desired outcome and the motivations for using certain artifacts, i.e. artifact motivations.

Users' Objects and Desired Outcomes

The object and the outcome are related as the transformation of the object in desired outcomes motivates and directs activity [25]. As described in the theoretical part of this chapter, the object of users can be captured by exploring what users actually want to download. Table 2 presents the types of content that our bittorrent participants attempted to download. The numbers in Table 2 have been gathered from 17 bittorrent users their incident logbooks. We distinguished the following types of content: audio, books, video/movie/TV series, photos, maps, games and software.

The largest share of content, downloaded via the bittorrent network, consisted of video content (57%), followed by music content (35%). In other words, the object is to download primarily video and audio files which users want to possess. The object is rather similar in all the activity systems of the participants, as they all want to download certain files. It should be noted that the share of video files would have been even larger if we would consider the file sizes instead of the amount.

Table 2 Types of content – bittorrent

Type	N	%	N failed	% Failed
Audio	57	35	15	26
Books	3	2	1	33
Video	91	57	34	37
Photos	0	0	-	-
Maps	4	2	0	0
Games	3	2	2	67
Software	3	2	0	0
Total	161	100	52	32

Interestingly, there are a lot of files that have failed to download. Of all the files in the incident logbook, 32% of the files were not found or downloaded. Users failed to find or download 26% of audio files and 37% of video content. It should be noted that, in the case of video content, the share of 37% has to be nuanced, as many participants indicated that some files did not succeed downloading during that particular session, but it did download in the course of several subsequent sessions. It is hard to predict, but 37% is definitely an overestimation of the amount of video files that did not download. In the case of the audio files, we have to notice that many of the audio files were entire albums, which were wrapped up in archive files such as RAR files. In sum, the bittorrent network is mainly used to download large files such as movies, TV series and entire music albums.

If a person attains an object, this person will hope that the object will result in the desired outcome. The desired outcomes of the participants turned out to be very diverging because it concerns different types of files and different personal preferences. For instance: the desired outcome of playing games is fun and amusement; the outcome of downloading games for someone else is to maintain good social relationships; the outcome of watching movies and series is entertainment and pastime; the outcome of downloading GPS maps is to enjoy mountain bike trips; the outcome of downloading software is to be able to study; university students download movies to drive away boredom during evenings; etc.

In other words, every subject downloads files with a fairly similar object – i.e. possessing files – whereas the outcomes of downloading these files are varying from case to case, depending on the particular circumstances. Most users have multiple outcomes in mind as they try to gratify different needs, e.g. on one moment they want entertainment by watching a movie, whereas on another moment, they will download software to practice for college. In other words, one artifact, in this case P2P networks, can be used in different activities to reach different outcomes. This stresses the role of P2P systems as mediating artifacts that are utilized to aid reaching desired outcomes, which implies that the use of P2P systems is not a motive in itself.

Artifact Motivations

In addition, a topic, which could not be discerned in activity theory, came to the fore in our data collection, namely "artifact motivations". This relates to the characteristics of the artifact, the service and the content which explain why end users have chosen a certain artifact, in this case P2P systems. We found that artifact motivations guide and direct behavior in important ways as will be illustrated in the following examination.

Some of the users download via bittorrent clients as they want to have the files physically on their own computer as this allows them to have control over how they consume content:

> R 11: But, if you want to watch a movie... I think it's easier to have movies on your computer. You can invite your friends. You can arrange a time when you want to watch it. You can stop it, you can watch it again.

Peers want to gain full control over the content they want to consume, which relates to the time and space of consumption. In other words, some of the users want to decide themselves when and where they are going to watch certain content. This attitude primarily emanates from dissatisfaction with how the traditional television industry has organized itself: linear television, commercials, different release moments in different countries, etc. Some of the series are not available in some countries whereas they are available via P2P systems. One of the participants expressed this view as follows:

> R 22: Actually, it's a lot more practical compared to TV. On TV, you have commercials and it's on fixed points in time. Whereas, with P2P, you can watch it anywhere, you can take it anywhere, you can save it on your hard drive. You don't have to wait long; you can do it whenever you want to. (...). For instance, other movies, such as Japanese movies, that are hard to get, that is something that I actually download quite a lot. That isn't broadcasted on TV here, and if it's on TV, then it mostly has voice-overs and that kind of stuff. (...). I like that it's really fast. For example, for a Japanese video, if they broadcast it on Wednesday, it's online on Thursday. "Lost" for example is already available one hour later on bittorrent. So, it's really fast. Sometimes, there are movies available that haven't been in the movie theatre yet.

In a nutshell, users want to gain control of the content. First, users do not want to be limited by what their TV channels offer them, as they downloaded series that were not available on local TV channels. Second, end users want to consume the content as soon as it has been released somewhere in the world. Most popular series and movies are first released in the USA, and then in other parts of the world. This entices many curious users to download these series via P2P networks as there are always peers that make this content available on torrent sites. Third, it is important to end users to have the content stored on their computer as this allows them to watch the content whenever and where they want to. In this sense, they escape the strict TV schedules and they are able to watch at their own pace, which is more tuned to their own time schedule. Fourth, a lot of peers hate commercials that interrupt their viewing experience, which motivates them to download it illegally via P2P systems. Some of the participants were real fans as: they wanted to watch their favorite TV programs as soon as possible, they watched it repeatedly; they often bought the content afterwards; and they wanted their friends to see these series as well. All these "fan" aspects stimulated these users to download content via P2P networks.

It should be mentioned that the choice of artifacts might be influenced by the lack of other artifacts as well, namely the lack of a TV or a TV-subscription. As many participants only had a computer and an Internet subscription, they turned to P2P systems to download and consume content.

Another (obvious) reason for downloading via P2P systems, which is very prevalent, is the fact that it is free of charge. Users save money as they do not have to pay a DVD (rental) shop, or a cinema. However, as already mentioned, some of the participants buy some of the content afterwards, which can be described as sampling: they consume the content first, and if it meets their needs in terms of quality, they purchase it. Finally, two other reasons for utilizing P2P systems relate to the fact that it is time-saving and there is a lot of content available that is difficult to retrieve elsewhere on the Internet. In addition, using P2P networks to download movies is time-saving as people do not need to leave their house e.g. to rent a DVD. Finally, the availability of the desired content, without P2P systems, is rather limited on the Internet and on TV.

4.1.2 Bittorrent Peers' Behavior

In this section, we will describe the usage process of bittorrent clients starting from how users search for content till how they will consume the content.

Search Process

The search process of bittorrent users proved to be very complex as users want to check whether (1) certain content will fit their taste, (2) they can trust the content, (3) the content has adequate QoC (Quality of Content), and (4) the content can be downloaded in a fast way. These aspects can be considered as goals that guide users' actions. QoC refers to the quality of sound and images of certain content.

First, the content users search for originates from different sources of inspiration. Most users do not search precipitately for content as, in most cases, users know what they are willing to download in advance. They encounter content for instance because of recommendations of friends or acquaintances. Other sources of inspiration are e.g. movie reviews in newspapers, commercials on TV, websites that recommend artists that might suit the users' music taste, posters in movie theatres, DVD covers in stores, information on the Internet in general (blogs, forums, specialized websites, chat boxes, etc.). For instance, Internet Movie Database proved to be a popular source to search for new content. However, sometimes, users do not utilize these artifacts, and they try to remember content – e.g. a song – when they hear it on the radio and they download it later on. Fans of series sometimes even remember on which days a new episode is going to be released, and on that day, they search for the new released episode. Finally, several users browse through titles and download content arbitrarily if it seems interesting to them. In these actions, users are primarily concerned with whether the content they search for will fit their taste. To achieve this goal (i.e. fit between the content and personal preferences) they use one or several of the above mentioned artifacts.

Further, trust emerged to be a major issue on bittorrent as every participant of this research had already encountered fake or malicious content. As we will demonstrate, some users spend a lot of time checking whether certain content is reliable or not. This fear has been expressed by one of the participants in the following way:

> R 1: There's a high level of uncertainty with P2P. And that's no good. There are so many spoofs, there are so many people with bad intentions that are seeding that kind of stuff. For me, that's to much insecurity.

Some users first surf to release websites – such as NFOrce and VCDquality – that publish which files have been released by which groups. As these are solely release websites, torrent files cannot be downloaded from these kinds of websites. People use these release websites for several reasons. First, they want to make sure that it is possible that an episode or a movie has been released. In this way, they know that when they encounter a certain file in search results whether this file might be real or not. In addition, users can verify the QoC on sites such as VCDquality, as it provides ratings of video/audio/movie, type of release, comments and jpeg-previews. Type of release has been valued to be important as this reveals much information about the QoC. For instance, in comparison with DVDRips, "CAM" versions are less appreciated as these are rips that have been generated by filming a movie with a video camera in movie theatres. Further, the trustworthiness of releases can be verified by controlling which groups have released it and the comments that other users have provided. Groups are persons that release content on for instance torrent sites. The above mentioned practices, mediated by release sites, can be regarded as actions to guarantee QoC and reliability. The information, users find on these release sites, is then utilized to generate search strings on torrent sites. Next, torrent sites, such as Mininova and The Pirate Bay, are websites where users can search and download torrent files. Or translated in bittorrent language, these websites contain trackers that allow users to download torrent files. Torrent files keep track of which peers make pieces of a specific file available so other users are able to download the requested file. To increase the range of their search, users often visit meta-search engines, such as Torrentz and TorrentPond, which search for torrent files on different torrent sites simultaneously to increase the chances for satisfactory search results. On torrent sites and meta-torrent-search engines, users have the following goals:

(1) to find the right torrent files in the search results, (2) to guarantee QoC, (3) to ensure the trustworthiness of content and (4) to increase the download speed.

It is interesting to note that some users have acquired specific search strategies in order to limit the amount of useless search results and to ensure that they have the right, most qualitative and most reliable file. To refine the search results, consumers of series often add the number of the season and the number of the episode, e.g. "s02e9", in the search string to assure that they have the right file. Furthermore, while searching, users often include the release type, e.g. "DVDRip", and the name of a trusted group, e.g. "aXXo", in their search strings. With regard to the release type, the QoC of DVDRips is perceived to be superior in comparison with e.g. "CAM" versions. Concerning the release groups, aXXo is a renowned and trusted group that regularly releases DVDRips. Peers enter these kinds of terms, such as "aXXo" and "DVDRip", with the aim of guaranteeing the trustworthiness and QoC.

> R 11: aXXo, I don't know what it is, but I've heard from a friend that, when you open that, that it always has good quality and that it's the right movie. Because it might happen that it's another title. (...).
>
> I: Are there other things that you pay attention to, to see whether it is reliable or not?
>
> R 11: Yeah, mostly aXXo. I never had trouble with that. So I trust that.

Next, users press the search button and a search result of torrent files appears. In the search results, users pay attention to several things. Some users take a look at the file name to make sure that it is the right version as the file name often contains information such as: the group, type of release, title, language of subtitles, etc. Moreover, several users pay attention to the comments, which accompany most of the search results, with the aim of verifying the trustworthiness of the file and sometimes the QoC. The following quote illustrates this point:

> R 13: Yeah, certainly in the case of movies. I almost always take a look at the comments. Not only to see whether there is a virus on it, or that it's real or fake, but also because it says whether the quality is good and whether it's a real DVDRip. Because, it often turns out that it was videotaped in a cinema with bad sound and bad images. And I don't want that, so I check it a lot.

In addition, some users take a look to the more detailed descriptions of the torrent so as to make sure that it is the right file in the right format and that it does not contain files that are redundant.

Further, the file size is also important in terms of trustworthiness, QoC and speed. We have to remark that the size is an issue as well as the participants in our research had a monthly restricted amount of bandwidth. Because of these bandwidth limitations, most users do not download High Definition (HD) versions of content. Smaller versions of content download faster as well in comparison with the HD versions.

> R 22: (...) And you can always choose between HDTV... It's actually always... Well, it's always HDTV, but you can choose between an HD or a Standard Definition (SD) version. An HD, that's about 1 GB or so, so that weighs. In most cases, I take SD versions because it's better to run your computer and it's also better for your (bandwidth) limits.

However, when a file is too small, several users said that it is a sign to them that the content is either fake or that it has poor quality. Other reasons for paying attention to the content size are: limited storage capacity and being able to burn it on a CD.

> R 24: I try to have content that's about 700 MB, then its quality is good enough. That's one CD. If it's, as here for instance (shows on computer), 1,37 GB, then it's actually way too much. You don't need that. It takes longer to download and it consumes more storage space.

Another aspect that turned out to be important is the number of seeders and leechers. A seeder refers to a user who offers an entire file, and a leecher is a user who offers one or several parts of a file. The primary goal of checking the number of seeders and leechers is the download speed. The more people sharing the content, the faster the file will be downloaded. Moreover, the number of seeders/leechers is important as some users reason that if there are a lot of people who possess the content, then there is a good chance that it is not a fake file. The other way around, if a file has only a limited amount of seeders, the file is considered to be suspicious.

> R 14: It mainly depends on the number of seeds. If there are 24.000 people who have that file, then it's like: "24.000 people, they won't all do it to fool me".

In summary, on the release sites and torrent sites, all the above mentioned tools – e.g. comments, search engines, file name, ratings, groups, number of seeds, file size, etc. – can be considered as artifacts that aid users in their actions to achieve goals such as: finding the right torrent files, ensuring QoC and reliability, and guaranteeing fast downloads. Some of the users utilize most of the mentioned artifacts, whereas others only use a few of them.

Download and Post-Download Actions

After finishing this rather lengthy part of the usage process, users decide which of the files meet their goals, in terms of e.g. QoC and reliability, and they download the torrent file. In most cases the torrent file opens automatically in the bittorrent client and starts downloading the requested file. What users do while the file is downloading is very diverging. Some users let their P2P client download the content, while doing other things in the meanwhile. Other users check once in a while how the download is progressing and whether the download is completed or not. The latter check the progress of the download by looking at for instance the speed, ETA (Estimated Time of Arrival), number of seeders/peers, the amount of connections, the amount of downloaded files, etc.

In some cases, users download archive files, such as RAR files, which contain several separate files. When users do not need all these files, they select the files they need so as to make sure that only these files are downloaded. This happens quite a lot in cases where for instance users download torrent files of entire music albums whereas they only need a few songs of it. In the case of TV series, where whole seasons are compressed in one RAR file, several users often indicate priorities so as to the first episodes are downloaded first (highest priority), and the last episodes are downloaded later on (less priority). Obviously, it is of little use to download the last episode of a season first, while you did not watch the previous episodes yet.

Further, several users pay attention to the amount of uploads as they only have a limited amount of bandwidth capacity. Most of these users limit uploading as much as possible to save bandwidth. Some of the participants utilize schedule tools of their P2P software as some Internet providers charge bandwidth consumption with 50% discount during nights:

> R 30: I mostly use it at night, because the Internet provider charges the hours only half during nights.
>
> I: You mostly use it just before going to sleep?
>
> R 30: Yeah, or you have a plug-in to schedule it. ... Here you can program when you want to download. So red means 'not downloading', and here 2h at night till 7h in the morning. And then it downloads automatically.

Finally, users consume the content, with the aim of gratifying their desires. Users do not utilize P2P networks because they like P2P networks, but they use it because this technology allows them to retrieve the content (object) with which they hope to achieve their desired outcome. In other words, P2P technology is neither the object, nor the outcome for the end users, but solely a mediating artifact that gets them closer to their objects. After having downloaded their content, users do very various things with their content as they: use it as samples for future purchases; store and listen to the music on their iPod; burn music on a CD to listen to it in the car; store music on computer so they can listen to it when they are using the computer; watch series/movies on the television or computer; watch series/movies together with friends; etc. It is remarkable as well that many users share this content with their friends afterwards. Either they watch the content together, or they exchange content for instance by swapping memory sticks or external hard drives.

4.2 Process of Using Gnutella Clients

Most of the participants in this section utilized Gnutella clients such as Limewire. There were a few exceptions of users that utilized Soulseek or Kazaa Lite. We assembled the Gnutella, Kazaa Lite and Soulseek users as these systems were utilized for the same purposes. Therefore, to avoid laborious phrasings, we will talk about Gnutella clients.

4.2.1 Different Levels of Motivations

In this section, we will dilate upon the different levels of motivations, i.e. the objects, the desired outcomes and the artifact motivations.

Users' Objects and Desired Outcomes

By achieving certain objects, users might obtain desired outcomes. In other words, people use Gnutella clients as these systems permit them to retrieve the content (object) with which they can gratify their needs (desired outcomes). Whereas the users' objects in the bittorrent case were very divergent, the object of using Gnutella clients is primarily to download audio content. As indicated in Table 3, 90% of the content consists of audio files, whereas video files only have a share of 8%. Whereas many users on bittorrent clients mainly download entire albums, users on Gnutella networks download many single songs which require less bandwidth capacity in comparison with the larger files that users retrieve on bittorrent networks.

Table 3 Types of content – Gnutella

Type	**N**	**%**	**N failed**	**% Failed**
Audio	231	90	17	7
Books	0	0	-	-
Video	21	8	15	71
Photos	0	0	-	-
Maps	0	0	-	-
Games	0	0	-	-
Software	6	2	6	33
Total	257	100	38	15

10 out of 25 participants utilized Gnutella clients as their primary source to download content. In addition, 4 bittorrent users made use of Gnutella clients as well to download primarily music. Two of the participants did not have to complete the incident logbook as they only utilized their P2P client a couple of times a year. For this reason, the numbers in Table 3 originate from 12 users. These numbers indicate that the availability of music files is excellent on these networks as only 7% of the files were not found or were fake files. On the other hand, Gnutella networks seem to be inappropriate for video content as 71% of the files were not found, did not complete their download or were corrupted.

The desired outcomes that can be achieved after attaining the object vary as people have different preferences. According to the participants, the outcome of listening to music might be: avoiding emptiness when you are alone; enjoying background music when friends come over; to make work/studying a more pleasant experience; for entertainment; to disrupt the silence; to kill time; to be able to be a DJ; etc. One of the participants sometimes even downloaded video files to possess illustrations when he had to teach courses.

Artifact Motivations

In comparison with the bittorrent case, the artifact motivations for using Gnutella networks are partly similar as we will demonstrate in the following paragraphs. A first artifact motivation consists of the fact that Gnutella networks allow users to store songs directly on their computer without having to rip a CD. It seems to be important for users to have control of the files so they are able to create a collection, to listen to the music the way they want to, to be able to listen offline, to exchange files with their friends. In comparison with listening to the radio, users of P2P systems can download content that fits their taste so they do not have to listen to music they dislike. In other words, users attach great importance to having music stored on their computer.

Another important reason for many participants to utilize Gnutella systems is the fact that it is free of charge. Many participants were students that dispose of limited means, as the following quote exemplifies:

> R 25: Also because it's for free... because I don't know how I should pay for CDs. Or iTunes, 99ct for one song. If I should download it with that, what I download today with P2P networks, then I should find a job just to listen to music. And I don't think that's worth it... And I also think that there's nothing wrong with it. (...). On average, I download about 100 songs in one month. I really can't spend 100EUR a month on music.

In addition, some participants disliked the idea of buying CDs as they reasoned that when you buy a CD, you are mostly only interested in a few songs of that particular album, whereas you have to pay for the songs that you are not interested in as well. On the other hand, some participants purchased some of the content after having downloaded them, which implies that they utilize P2P systems to sample music.

Moreover, the speed with which users are able to retrieve music via P2P systems appears to be important as well:

> R 19: You hear a song, you enter the title or a part of the title and a couple of minutes later, you have the song. (...). The possibility to hear a song on the radio and to have it a couple of minutes later with a minimum effort... that's the most important reason.

Additionally, speed refers also to the fact that users do not need to go to the music shop and search for a CD on the shelves. The content available on P2P networks does not need to be ripped as is the case with CDs. Furthermore, alternatives such as music streaming sites consume too much bandwidth as users have to use bandwidth each time they want to consume their music, whereas with P2P systems, they only have to download their desired songs once. It should be noted that bandwidth issues turned out to be especially relevant in the case of users that download video content with P2P systems because this type of content consumes more bandwidth.

4.2.2 Gnutella Peers' Behavior

In comparison with the bittorrent case, the usage process of Gnutella clients is far less complex as we will illustrate in this section.

Search Process

The content, that participants download, originates from different encounters, which inspired users to search for this content. Sometimes, users get recommendations from friends to listen to certain music. Other users browse through playlists and charts – e.g. websites of radio stations, billboard.com – to check whether there might be something interesting to download. Users have not always heard a song before downloading it. In such cases, a song was often recommended by friends, or a song seemed to be interesting because it fits their music style and it did well in charts. However most users do know the content which they want to download, they often forget the title or the name of the band, which forces them to surf several websites – e.g. Google, Wikipedia, etc. – to find this information so they can search it in Limewire. Sometimes, when users are not sure about the title or band, they search on Youtube and watch it to verify the information so they do not download the wrong files. Several users utilize their mobile phones in order not to forget the name of a song so as to be able to formulate accurate search strings in their P2P client. Other sources of inspiration are listening to the radio, watching music TV channels, browsing through Youtube videos, hearing music in movies, and so on. Several users do not aim to download specific songs, but they regularly check whether they can find new songs of their favorite artists on Gnutella networks. In such cases they only enter the name of the band, and they take a look whether there is new content from this specific band available on Gnutella networks. While browsing through search results, it often happens that users download other songs from an artist than originally intended.

After knowing what kind of content users want (object), they open their P2P client and start searching for the desired files. Most users first choose which type of content they want to download in order to filter the search results from inadequate content types. Another reason for choosing the content type is to limit the amount of advertisements. Next, they have to enter a search string – artist or title – which must allow them to achieve the most accurate search results. Sometimes, peers enter only the artist name because they want to check whether their favorite bands have released new songs.

Further, most of the participants focused on the search results in order to distinguish the most reliable files with good QoC from the fake or corrupted files. It is interesting to note that several users have developed certain search strategies in order to increase reliability, quality and speed of the search results. Several users indicated that search results that are exactly the same as the search string are less reliable compared to files that have additional characters in their title such as a number or a hyphen. In other words, some users perceive it to be a good sign if a file name does not exactly match the search string. Therefore, although some users know the exact entire name of the file they intend to download, they sometimes enter incomplete information in the search field. As the search results are shown, these users will only trust the file names that contain the missing information:

> R 19: This, for example, I would trust (points to a file in Limewire), because I only entered the title of the song… and here, the name of the performer appears which makes me feel that it's the right one. (…). For me, that's a reason to trust it.

In sum, irregularities in file names are perceived to be good signs in some cases, whereas in other cases this leads to distrust. Distrust might occur for instance when a file name has words that are not related to what one is willing to download.

Another strategy to decrease the chances for downloading fake files is to check whether it is the only file of a certain size. Several users do not trust files if they have exactly the same size as other files in search results. Research [20] has demonstrated that this strategy appears to be very efficient to filter out malware.

Some of the participants combined several of the above mentioned search strategies as illustrated by the following user:

> R 19: I'm looking for a file that's the only one of that size, or that's the only one with that kind of notation. Some of them use interspaces, others use an underscore, others use dots, others use normal hyphens between words, or with a number in the end or in the beginning. Then you know that there's a good chance that it has good quality, and second, that it's going to work as well.

Many users pay attention to the file size to make judgments about the QoC. The reasoning goes that if a file is too small, it would probably have poor quality:

> R 25: In Limewire, I always take a look at the size and how much there are with the same size. And I always pay attention that it's not a file of for example 59 kB for an MP3 file. In such a case, you know that it can't be right.

Only a few users paid attention to the bitrate of a file to figure out the quality. Another important way of controlling quality and reliability is by previewing the file in Limewire itself via its media player while the file is still downloading. In this way, users can listen for themselves whether the sound meets the users' requirements and whether it is the right file:

> R 12: And I do a preview so I can see whether it's what I'm looking for. If it turns out to be a version with a lot of impurities, then I delete that and try to look for another version.

A number of participants selected several files of the same song to download and to "postview" these. The primary reason to download several versions of the same file is to compare the QoC of these different files and delete the least satisfactory ones:

> R 7: I often download 3 or 4 versions of a song and then I listen to it… mostly in iTunes. I mostly don't work with this (points to the media player in Limewire). And then I choose one of them and I delete the rest. Sometimes I download quite a lot of the same version… about 5 versions. But I delete some of them. There are sometimes poor files in it and I delete those.

Another aspect that appeared to be important to users is the download speed. Many users focused on the amount of uploaders in order to be sure that the file can be downloaded in a fast way. Indeed, it is often the case that the more peers make certain content available the faster it might download. Some users even perceived this to be a criterion for the reliability and QoC as they reasoned that if a lot of people have it, it is probably reliable and has good quality:

> R 12: I also take a look at the number of hits in advance. If there are 20, then you know that it's going to be fast in comparison with when there are only two hits. In that case I take the one with 20 users. In such case, you have more chance to have the right file, because if there are 20 participants on one file, it must be a better file. (...). If it's a bad file, then everyone deletes that. So, in this sense, it's a good file because those people keep using it. And that's why... quality is already guaranteed in a certain way because you know that a lot of people are using it, so it probably won't be bad.

Speed is also being derived from the "Speed" column which indicates what type of connection the uploader has. This column displays whether a user has for instance a T1, T3, modem or cable/DSL connection. For instance, a T3 connection is faster in comparison with a modem or cable/DSL connection.

It should be mentioned that most participants did not utilize all of these attributes to verify the reliability or QoC. For instance, one participant even clicked around in a random way, not fearing to download corrupted content.

In sum, in contrast with the usage of bittorrent clients – where the QoC and reliability were verified before downloading files – users of Gnutella clients mainly checked the QoC and reliability afterwards by listening to the files in their media players.

Download and Post-Download Actions

In most cases, downloading files goes rather fast, leaving little time for users to do other things while their P2P application is downloading songs. For this reason, most users keep an eye on the progress of the download, whereas others start searching new files in the meanwhile or they organize the already downloaded files in their music collection.

After downloading the files, many users transfer these files – often automatically – in iTunes, where they organize their collection. Some users burn the songs on a CD to listen to it in their car; others listen to it on their MP3 player/iPod or on their computer. Some of the users, who utilize P2P networks to sample content, or who fear copyright infringement buy some of the songs afterwards:

> R 19: Now and then, I get this moral urge to compensate, and then I buy a whole bunch of albums to ease my conscience... to legalize my music.

Interestingly, most of the Gnutella users share their downloaded music with their friends via MSN, e-mail, by exchanging memory sticks or by burning it on a CD. The primary reasons for exchanging these files are to do friends a favor, or because they want to recommend certain music that they like. It is a bit paradoxical that most users do not upload or share their downloaded content on P2P networks, whereas they do share this content with their friends.

5 Discussion and Conclusion

The purpose of the current study was to explore the usage of P2P file-sharing systems by means of qualitative, explorative, contextual research. To understand the usage of these systems, thorough attention was paid to the relationship between the objects/goals/motivations, behavior and the relevant context. We found that different types of systems are utilized in different ways. Therefore, a distinction was made between the use of bittorrent (e.g. Azureus, KTorrent, BitComet, BitTorrent) and Gnutella clients (e.g. Limewire, Soulseek). The findings of the present study enhance our understanding of usage behavior on P2P networks.

The results of this study show that bittorrent clients are mainly used to download large files such as video, games, software and entire albums. People probably turn to bittorrent systems to download large files as this protocol has proven to be very efficient to exchange large content because of the swarming technology with which different parts of the same file can be retrieved from different peers. In the theoretical framework, we argued that the object and desired results direct the actions of users. For instance, college students that have their own apartment might download movies (object) via P2P networks to kill the time during the evenings (desired outcome). In addition, we found that the artifact motivations played an important role in the reasons for participants to use bittorrent clients. The artifact motivations comprise the characteristics of an artifact, service or content which explain to a large extent why users adopt a certain artifact. The current study found the following artifact motivations of bittorrent users: dissatisfaction with traditional TV, free of charge, vast availability of content and control over files. These artifact motivations should be considered as particular context aspects that often stem from users' lifestyle, e.g. college students with irregular schedules, that have their own apartment and that do not have cable subscriptions. Other context factors that appeared to influence users' actions related to the presence of malware, fake files and files with insufficient QoC. The existence of such files caused many users to verify in advance whether a file meets the expectations in terms of reliability and QoC by means of tools such as ratings, comments and jpeg-previews. These preventive inspections are presumably reinforced by the bandwidth caps in Belgium which makes that users want to avoid that the bandwidth they paid for, would be wasted on worthless or even dangerous files.

On the other hand, the study found that peers utilized Gnutella clients primarily to download small, unique audio files. Users downloaded music (object) with the aim of for instance making work more pleasant or passing time while commuting (desired outcome). In addition, analysis revealed that the artifact motivations of Gnutella systems correspond to a large extent with the artifact motivations if bittorrent clients. The Gnutella artifact motivations comprise among others: control over files, free of charge, dissatisfaction with the medium CD, extensive availability of files, possibility to sample and explore music. Furthermore, the omnipresence of fake files and malware influences users' behavior as these users want to make sure that they are downloading the right files with satisfactory QoC. Users of Gnutella clients have their own ways of assessing the QoC and reliability of files, e.g. by means of the file size, the number of uploaders, previews. Nevertheless, it should be mentioned that the availability of tools to verify QoC and trustworthiness in Gnutella clients are far more limited in comparison with the bittorrent clients.

Then why do music lovers adopt systems with which they only have limited possibilities to check the QoC and reliability and why do they not turn to bittorrent clients? This may be explained by a number of factors that are related to the technology and the file type. First, the availability of single songs is rather low on bittorrent probably because it would require a time-consuming procedure to make a torrent file for each separate song. Second, it would demand too much time and effort to verify the QoC and reliability of many small files in a thorough way as bittorrent users do with e.g. video files. Indeed, Gnutella users download more and smaller files during a download session in comparison with bittorrent users who retrieve less but larger files. In other words, Gnutella users download more files in number, whereas bittorrent users download more files in terms of volume. In our opinion, these findings might partially explain the differences in the use between both types of systems. Furthermore, we assume that the bandwidth caps play an important role as well. For example, if one downloads a single song, and the QoC turns out to be rather poor, then this person has only wasted approximately 3 MB. On the other hand, if one retrieves a fake movie or a movie with insufficient QoC, then this person has dissipated a lot of time and has lost easily 700 MB, which is a high price to pay when you only have a monthly bandwidth capacity of for instance 25 GB that you might have to share with other people.

The findings have important implications on several levels. To the research community, this study has demonstrated that user behavior should be studied as a complex interaction between motifs/goals/objects and the specific meaningful context. In this respect, activity theory provided an adequate framework to grasp the interrelations between the different variables. In addition, the results might be interesting to policymakers and the media industry as these interested parties can learn how and why people are using P2P file-sharing systems. For instance, the insertion of corrupted files in P2P networks by the media industry has only minor effects on experienced users as they adapted their search strategies in efficient ways which decreases the chances that users download fake files and malware. Another interesting result for the media industry is the fact that these users want to have control over their consumption themselves. P2P file-sharing users do not want to be restricted by what the media imposes on them, but the content needs to be flexible so as to it fits the lifestyle of the individual.

Finally, a number of important limitations need to be considered. First, the relatively small sample size of the study, in which we examined the use of P2P networks by youngsters, is a threat to the study's validity. Accordingly, caution must be applied, as the findings might not be transferable to the entire P2P file-sharing community. Second, the end users' behavior is influenced in important ways by bandwidth caps, whereas other users in many countries have "all-you-can-eat" subscriptions. Therefore, a promising line of study would be to enlarge the sample with elder users and with users that are not restricted by bandwidth caps.

In conclusion, our theoretical framework, combined with the results of this study indicate that P2P networks should be studied in its context of use as the usage of P2P systems is not an object nor an outcome for users. Users only utilize P2P networks as one of their mediating artifacts that bring them another step closer to their object and desired outcomes.

Acknowledgements FLEET (an interdisciplinary research project in FLEmish E-publishing Trends) is an IWT SBO project, with research partners IBBT SMIT, Cemeso, LSTS, MOFI, ICRI, CUO IBBT, MICT, ECDC and TNO.

References

1. Andersen B, Frenz M (2007) The Impact of Music Downloads and P2P File-Sharing on the Purchase of Music: A Study for Industry Canada. Available via Industry Canada. Cited 23 Oct 2008
2. Bannon LJ, Bødker S (1991) Beyond the Interface: Encountering Artifacts in Use. In: Carroll J (ed) Designing Interaction: Psychology at the Human-Computer Interface, Cambridge University Press, New York
3. Bazeley P (2007) Qualitative Data Analysis with NVivo. Sage Publications, London
4. Bertelsen OW, Bødker S (2003) Activity Theory. In: Carroll JM (ed) HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science, Morgan Kaufmann Publishers, San Francisco
5. Bødker S (1989) A Human Activity Approach to User Interfaces. Hum-Comput Interact 4(3): 171–195
6. Bødker S (1991) Through the Interface: A Human Activity Approach to User Interface Design. Lawrence Erlbaum Associates, Hillsdale, New Jersey
7. Bødker S (1993) Historical Analysis and Conflicting Perspectives – Contextualizing HCI. Paper presented at the East–West HCI conference, Moscow, Russia
8. Bødker S (1996) Applying Activity Theory to Video Analysis: How to Make Sense of Video Data in Human-Computer Interaction. In: Nardi BA (ed) Context and Consciousness: Activity Theory and Human-Computer Interaction, The MIT Press, Cambridge
9. British Music Rights (2008) Music Experience and Behaviour in Young People. Available via BMR. `http://www.ukmusic.org/cms/uploads/files/UoH%20Reseach%202008.pdf`. Cited 1 Oct 2008
10. Carey J (2008) Peer-to-Peer Video File Sharing: What Can We Learn From Consumer Behavior. In: Noam EM, Pupillo LM (Eds) Peer-to-Peer Video: The Economics, Policy, and Culture of Today's New Mass Medium, Springer Science + Business Media, LLC, New York
11. Christin N, Weigend AS, Chuang J (2005) Content Availability, Pollution and Poisoning in File Sharing Peer-to-Peer Networks. Paper presented at The Sixth ACM Conference on Electronic Commerce, Vancouver, Canada
12. Condry I (2004) Cultures of music piracy: An ethnographic comparison of the US and Japan. Int J Cult Stud 7(3): 343–363
13. Courage C, Baxter K (2005) Understanding Your Users. A Practical Guide to User Requirements: Methods, Tools, & Techniques. Elsevier, San Francisco
14. Crawford K, Hasan H (2006) Demonstrations of the Activity Theory Framework for Research in Information Systems. Australas J Inf Syst 13(2): 49–67
15. Einav G (2008) College Students: The Rationale for Peer-to-Peer Video File Sharing. In: Noam ME, Pupillo LM (Eds) Peer-to-Peer Video: The Economics, Policy, and Culture of Today's New Mass Medium, Springer Science + Business Media, LLC, New York

16. Engeström Y (1987) Learning by Expanding: An Activity-Theoretical Approach to Developmental Research. Orienta-Konsultit Oy, Helsinki
17. Fokker J, De Ridder H, Westendorp P, Pouwelse J (2007) Inducing Cooperationin Peer-to-Peer Television Systems. Paper presented at EuroITV'07, Amsterdam, The Netherlands
18. Gavosto A, Lamborghini B, Lamborghini S (2008) Peer-to-Peer Network and the Distribution in the EU. In: Noam ME, Pupillo LM (Eds) Peer-to-Peer Video: The Economics, Policy, and Culture of Today's New Mass Medium, Springer Science + Business Media, LLC, New York
19. Gay G, Hembrooke H (2004) Activity-Centered Design. An Ecological Approach to Designing Smart Tools and Usable Systems. The MIT Press, Cambridge
20. Kalafut A, Acharya A, Gupta M (2006) A Study of Malware in Peer-to-Peer Networks. Paper presented at the Internet Measurement Conference, Rio de Janeiro, Brazil
21. Kaptelinin V (1996) Activity Theory: Implications for Human-Computer Interaction. In: Nardi BA (ed) Context and Consciousness: Activity Theory and Human-Computer Interaction, The MIT Press, Cambridge
22. Kaptelinin V (2005) The Object of Activity: Making Sense of the Sense-Maker. Mind, Cult, Activity 12(1): 4–18
23. Kaptelinin V, Nardi B (2006) Acting with Technology: Activity Theory and Interaction Design. The MIT Press, Cambridge
24. Kozulin A (1986) The Concept of Activity in Soviet Psychology: Vygotsky, His Disciples and Critics. Am Psychologist 41(3): 264–274
25. Kuuti K (1996) Activity Theory as a Potential Framework for Human-Computer Interaction Research. In: Nardi BA (ed) Context and Consciousness: Activity Theory and Human-Computer Interaction, The MIT Press, Cambridge
26. LaRose R, Lai Y-J, Lange R, Love B, Wu Y. (2005) Sharing or Piracy? An Exploration of Downloading Behavior. J Comput-Mediat Commun 11(1)
27. Lee J (2003) An End-User Perspective on File-Sharing Systems. Commun ACM 46(2): 49–53
28. Leontiev N (1978) Activity, Consciousness, and Personality. Prentice-Hall, Englewood Cliffs, NJ
29. Lewis K, Massey C (2004) Help or Hindrance? Participant Diaries as a Qualitative Data Collection Tool. Paper presented at the 17th Annual Conference of the Small Enterprise Association of Australia and New Zealand, Brisbane, Australia
30. Mann C, Stewart F (2000) Internet Communication and Qualitative Research: A Handbook for Researching Online. SAGE Publications, London
31. Nardi BA (1996a) Activity Theory and Human-Computer Interaction. In: Nardi BA (ed) Context and Consciousness: Activity Theory and Human-Computer Interaction, The MIT Press, Cambridge
32. Nardi, B.A. (1996b) Studying Context: A Comparison of Activity Theory, Situated Action Models, and Distributed Cognition. In: Nardi BA (ed) Context and Consciousness: Activity Theory and Human-Computer Interaction, The MIT Press, Cambridge
33. Quek A, Shah H (2004) A Comparative Survey of Activity-based Methods for Information Systems Development. Paper presented at the 6th International Conference on Enterprise Information Systems, Porto, Portugal
34. Shin S, Jung J Balakrishnan B (2006) Malware Prevalence in the KaZaA File-Sharing Network. Paper presented at the International Measurement Conference, Rio de Janeiro, Brazil
35. Suchman LA (1987) Plans and Situated Actions: The Problem of Human Machine Communication. Cambridge University Press, Cambridge
36. Uden L, Willis N (2001) Designing User Interfaces using Activity Theory. Paper presented at the 34th Hawaii International Conference on System Sciences, Maui, Hawaii
37. Zimmerman DH, Wieder DL (1977) The Diary: Diary-Interview Method. J Contemp Ethnogr 5(4): 479–498

Part II
Unstructured P2P Overlay Architectures

Unstructured Peer-to-Peer Network Architectures

Xing Jin and S.-H. Gary Chan

Abstract With the rapid growth of the Internet, peer-to-peer (P2P) networks have been widely studied and deployed. According to CacheLogic Research, P2P traffic has dominated the Internet traffic in 2006, by accounting for over 72% Internet traffic. In this chapter, we focus on unstructured P2P networks, one key type of P2P networks. We first present several unstructured P2P networks for the file sharing application, and then investigate some advanced issues in the network design. We also study two other important applications, i.e., media streaming and voice over Internet Protocol (VoIP). Finally, we discuss unstructured P2P networks over wireless networks.

1 Introduction

In the recent years, P2P networks have seen enormous successes and rich developments over the Internet. When Napster first emerged in 1999 as a P2P file sharing system, the Internet traffic was dominated by web and ftp (accounting for 65 and 10% total traffic, respectively, according to CacheLogic Research). But in 2006, 50–65% of downstream traffic and 75–90% of upstream traffic was already P2P traffic (according to CacheLogic Research). In total, P2P traffic has accounted for 72% Internet traffic in the year, while that of web and ftp has decreased to 24 and 2%, respectively.

Xing Jin
Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, e-mail: `csvenus@cse.ust.hk`

S.-H. Gary Chan
Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, e-mail: `gchan@cse.ust.hk`

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_5, © Springer Science+Business Media, LLC 2010

In P2P systems, cooperative peers self-organize themselves into overlay networks and store or relay data for each other. The major challenge is how to achieve efficient resource search in a large scale distributed-storage network. Currently popular P2P search systems can be classified as *unstructured* and *structured*, depending on the overlay structures. Unstructured systems do not impose any structure on the overlay networks [1–3]. These systems are usually resilient to peer dynamics, and support complex queries with meta information. But they are not efficient for locating unpopular files. Structured systems impose particular structures on the overlay networks, which are commonly referred to as distributed hash tables (DHTs) [38, 42]. In a structured system, any file can be located in a small number of overlay hops, which significantly reduces the search cost as compared to unstructured systems. However, DHT only supports single-keyword exact-match lookups.

In this chapter, we focus on unstructured P2P networks. We first study the file sharing application, which is the original and one of the most important applications for P2P networks. Depending on system decentralization level, we classify unstructured P2P networks into centralized, distributed, hybrid and some other approaches. We select representative examples from each category and analyze their search mechanisms. Then we discuss some advanced issues in system design, i.e., content replication and system security.

We also study other important applications for unstructured P2P networks, including media streaming and VoIP. One of the pioneering P2P streaming softwares, CoolStreaming, has reported to attract more than $25,000$ concurrent peers for one streaming channel [50]. Another streaming software PPLive reported more than $400,000$ concurrent peers for its over 300 channels [25]. As for P2P VoIP, the leading software Skype has shown to attract over 8 million concurrent users in 2008 (from its client software interface), and has reported to accumulate 276 million user accounts at the end of 2007. Clearly, these applications have evolved into influential network applications over the Internet. We hence select CoolStreaming and Skype as two typical examples. We study the challenges in these applications and their corresponding approaches.

Finally, we study the implementation issue of unstructured P2P networks over wireless networks. Wireless networks are going through quick development nowadays. As they share many similar features with P2P networks (such as decentralization and dynamic topology), there has been an increasing interest in integrating them together. We compare P2P networks and wireless networks, highlight the challenges in integrating them, and discuss several state-of-the-art approaches.

The rest of the chapter is organized as follows. In Section 2, we present the general design considerations in unstructured P2P networks. In Section 3, we discuss representative examples of unstructured P2P networks for file sharing. In Section 4, we explore some advanced issues in P2P file sharing. In Section 5, we discuss the media streaming and VoIP applications for unstructured P2P networks. In Section 6, we investigate how to implement unstructured P2P networks over wireless networks. Finally, we conclude in Section 7.

2 Design Considerations

When designing an unstructured P2P network, several issues need to be carefully considered:

- *Search efficiency and replication cost*: In unstructured P2P networks, data are distributedly stored at peers and each peer only holds limited information about the system. Hence, it is important to design efficient search mechanisms. Usually, with more data replications in the network, a data file can be located more quickly. It is hence a tradeoff between storage cost and search time.
- *Scalability*: A P2P system may consist of hundreds of thousands of peers. This often requires a fully distributed system, where peers form a self-organized network and each peer communicates with only a few other peers.
- *Resilience to peer dynamics*: In P2P systems, a peer may arbitrarily join, leave or fail. A good P2P system should be resilient to such peer dynamics.
- *Load balancing*: Peers often have heterogeneous resource (e.g., bandwidth, computational capability, storage space). A good system should be able to achieve balanced loads among peers. This can avoid overloading hot peers, and hence improve system scalability.
- *Security*: In the open environment of the Internet, some participating peers may be selfish and unwilling to upload data to others, or some may launch attacks to disrupt the service. A practical P2P system should be well protected to targeted attacks or free-riders (i.e., peers only downloading data without uploading). Other considerations include peers' privacy and confidentiality.

3 Unstructured P2P Networks for File Sharing

In this chapter, we describe several unstructured P2P networks for the file sharing application. We classified the approaches into four categories: centralized, distributed, hybrid and others. We select one representative example from each category and discuss its system design.

3.1 A Centralized Approach: Napster

In 1999, Napster appeared as the first P2P file sharing system [4]. A Napster system consists of a central directory server and a set of registered users (or peers). The server maintains information of all files in the system, including an index with metadata (such as file name and size) of all files in the system, a list of all registered peers, and a list showing the files that each peer holds and shares.

When a new peer joins the system, it contacts the server and reports a list of files it maintains and shares. When a peer wants to search for a file, it sends a request to the server. The server will return a list of peers that hold the matching file. The searching peer then contacts the returned peers to download the file.

Figure 1(a) shows the search process in Napster. When peer *A* wants to search for some file, it contacts the central server. The server returns some peers that hold the file, say, peer *B*. Peer *A* then starts to download the file from peer *B*.

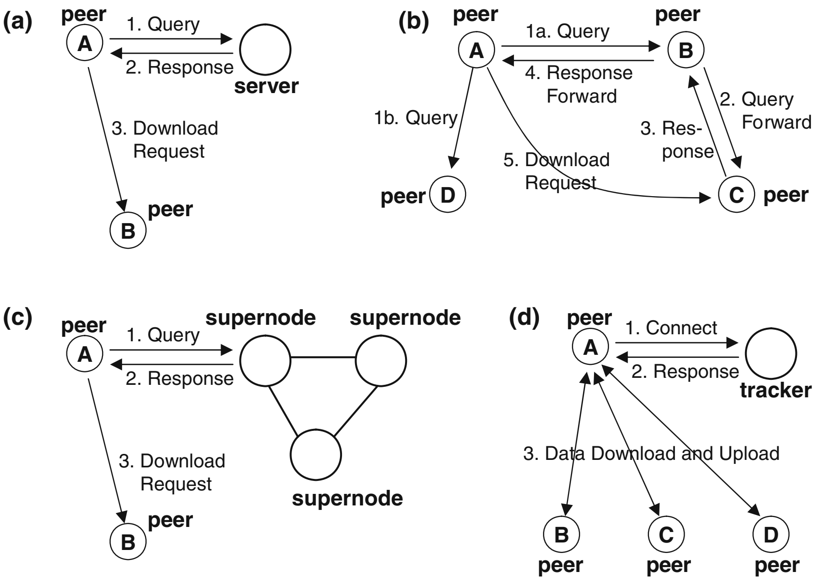


Fig. 1 Search process in unstructured P2P networks. (**a**) Napster. (**b**) Gnutella. (**c**) Kazaa. (**d**) BitTorrent.

The advantage of Napster is its ease of implementation and simplicity of deployment. The system administrator only needs to deploy and maintain a central server. Furthermore, the system is highly adaptive to peer joining and leaving. The major disadvantage is that such a centralized system is not scalable. The server needs to have much resource (such as computational capability and bandwidth) to support a large number of peers. In addition, the server forms a single point of failure. If the server is down, the whole system is broken. It is hence vulnerable to targeted attacks against the server (e.g., DDos).

3.2 A Distributed Approach: Gnutella

Gnutella is a fully distributed P2P system for file sharing [1]. It first appeared in 2000 and got quick development in a few years. According to Slyck news report, as of June 2005, Gnutella's population was 1.81 million computers. Gnutella's source code is publicly available in the Internet. This enables the development of different client softwares by different groups (e.g., LimeWire developed by the LimeWire group, or Gnucleus developed by the Morpheus group).

In the basic Gnutella protocol, when a new peer joins the system, it first connects to some public peers. For example, a list of public peers are available at *http://gnutellahosts.com*. A new peer then sends a PING message to any peer it is connected to. The message announces the existence of the new peer. Upon receiving a PING message, a Gnutella peer returns a PONG message and propagates the PING message to its neighbors. A PONG message contains IP address and port of the responding peer, and information of files being shared by the responding peer. In a dynamic network with frequent peer joining and leaving, a peer periodically sends PING messages its neighbors.

Search in Gnutella is based on flooding, which is broadcasting in the overlay. A search query is propagated to all neighbors from the original requesting peer. The query is replicated and forwarded by each intermediate peer to all its neighbors. Each intermediate peer also examines its local contents and responds to the query source on a match. The query responses are routed back along the opposite path towards the original requesting peer. To reduce the amount of query messages in the network, each query message contains a time-to-live (TTL) field. The TTL value will be decremented by one at each peer. When it reaches zero, the message is dropped.

Figure 1(b) shows the search process in Gnutella. Suppose peer A wants to search for some file. It floods its search query to its neighbors, i.e., peers B and D in the figure. When peer B receives the query, it checks whether itself holds the matching file. If not, it forwards the query to its neighbors. As in the example, peer B forwards the query to its neighbor C. Suppose C holds the file that A wants. C returns a response to the peer that sends it the query, which is B in the figure. B then continues forwarding the response to the query sender A. Finally, A contacts C to download the file.

Different from Napster, Gnutella is a dynamic, self-organized network. Each peer independently connects to and communicates with a few other peers in the system. The system is hence able to contain an unlimited number of peers, if no constraint on search efficiency. Meanwhile, the system is highly robust to peer dynamics. If a peer leaves the system, its neighbors can connect to other peers through the exchange of PING and PONG messages.

A limitation of Gnutella is its relatively low search efficiency. In flooding search, the number of query messages exponentially increases with the number of overlay hops. Then a query may generate many messages, especially for unpopular files, where a query has to go through many overlay hops and many peers before reaching a matching peer. Given the huge number of peers in the system, the traffic load for queries could be significantly high. The use of TTL can reduce the number of query messages. However, choosing an appropriate TTL is not easy. If the TTL is too high, peers unnecessarily burden the network. If the TTL is too low, a peer might not find the file even though a copy exists somewhere.

In addition, there are many duplicate messages in flooding search, particularly in heavily connected networks. This is because multiple copies of a query may be sent to a peer by its multiple neighbors. These duplicate queries consume extra processing power at peers and network bandwidth. Hence, it is necessary to develop some duplication detection mechanisms. However, even with duplicate suppression, the number of duplicate messages in flooding can be excessive, and the problem gets worse as the TTL increases.

Therefore, many new search methods have been proposed to improve or replace flooding search. Expanding ring is a method that addresses the TTL selection problem in flooding [34]. That is, a peer starts flooding with a small TTL. If the search does not succeed, the peer increases the TTL and starts another flooding. The process repeats until the search succeeds. This method is efficient for locating hot files. Hot files are widely replicated in the network, and a small TTL value is enough to locate them. But this method leads to higher search time due to repeated flooding.

Expanding ring does not address the message duplication problem. Another method uses multiple parallel random walks to address this issue [34]. In standard random walk, when receiving a search query, a peer randomly chooses one neighbor and propagates the query to it. To reduce the search time, the method uses multiple random walks. This method, together with proactive file replication (discussed in Section 4.1), can significantly improve the search performance in terms of search time, per-peer query load, and message traffic.

3.3 A Hybrid Approach: FastTrack/Kazaa

Given the limitations of purely centralized networks and purely distributed networks, there is a third approach which combines these two types of networks. FastTrack is a typical example as a partially centralized P2P protocol. In FastTrack, peers with the fastest Internet connections and the most powerful computers are automatically designated as *supernodes*. A supernode maintains information about some resource as well as connections with other supernodes. When a peer performs a search, it first searches for the closest supernode, which returns immediate results if any and refers the search to other supernodes if needed. Two practical softwares based on FastTrack are Kazaa [2] and Grokster [5]. But the latter closed its service in 2005 due to the copyright issue.

Figure 1(c) shows the search process in Kazaa. When peer *A* wants to search for some file, it sends the search query to the closest supernode. The supernode either returns some matching peers, or forwards the query to other supernodes. Finally, *A* will obtain some matching peers from the supernode (say, peer *B* in the figure) and download the file from these peers.

Therefore, an ordinary peer (e.g., peer *A* in the figure) communicates with a supernode as if communicating with the server in Napster. Then, Gnutella like search is performed in a highly pruned overlay network of supernodes.

As compared with purely distributed networks like Gnutella, Kazaa achieves much lower search time. Search among supernodes is much faster than search among all peers, because the number of supernodes is much smaller than the total number of peers. As supernodes have high bandwidth and large storage space, they can efficiently process a large amount of queries from ordinary peers. The system hence makes good use of peer heterogeneity. In addition, unlike Napster, it does not form a single point of failure. If some supernodes go down, the peers connecting to them can connect to other supernodes.

In view of the success of Kazaa, Gnutella also considers adopting a hybrid structure. Chawathe et al. propose the Gia system to improve Gnutella [12]. It uses a dynamic topology adaptation protocol to put most peers within a short distance of a few supernodes. Supernodes will receive a large proportion of the queries. Together with a few other improvements (e.g., active flow control and biased random walk), Gia increases the system capacity by three to five orders of magnitude. Later, Gnutella version 0.6 formally incorporates the idea of "ultrapeers". In detail, peers entering into the network are kept at the edge of the network as leaves, not responsible for any routing. Peers which are capable of routing messages are promoted to ultrapeers, which will accept leaf connections and route searches and network maintenance messages. Normally, a leaf peer is connected to 3 ultrapeers, and each ultrapeer is connected to more than 32 other ultrapeers. Within this network, the maximum number of hops a query can travel is reduced to around 4. The search efficiency and the whole system scalability are then greatly improved.

3.4 Other Approach: BitTorrent

BitTorrent is a P2P system that does not belong to any of the above categories [3]. As an important P2P file sharing application, it is estimated by CacheLogic to represent 35% of all Internet traffic in 2004.

BitTorrent uses a central location to coordinate data upload and download among peers. To share a file f, a peer first creates a small torrent file, which contains metadata about f, e.g., its length, name and hashing information. Usually, BitTorrent cuts a file into pieces of fixed size, typically between 64 KB and 4 MB each. Each piece has a checksum from the SHA1 hashing algorithm, which is also recorded in the torrent file. Most importantly, the torrent file contains the URL of a tracker, which keeps track of all the peers who have file f (either partially or completely) and the lookup peers.

A peer that wants to download the file first obtains the corresponding torrent file, and then connects to the specified tracker. The tracker responds with a random list of peers which are downloading the same file. The requesting peer then connects to these peers for downloading.

Figure 1(d) shows the search process in BitTorrent. When peer A wants to search for some file, it first needs to obtain the corresponding torrent for the file. From the torrent, A knows the address of the tracker and connects to the tracker. The tracker then returns a list of peers who are downloading or sharing the file. *A* then exchanges data with these peers.

In BitTorrent systems, torrent files are often published on large websites, which also serve as trackers. Clearly, the centralization of trackers brings some barriers in the system. If a tracker is down, peers will not be able to start their sharing (by uploading their torrents to the tracker), and new incoming peers cannot start their downloading. In order to remove the need of central trackers, the latest BitTorrent clients implement a decentralized tracking mechanism (e.g., μTorrent, BitComet, KTorrent). In the mechanism, every peer acts as a mini-tracker. Peers first join a DHT network, which is inherently implemented in the BitTorrent client. A torrent is then stored at a certain peer according to the DHT storage method. All peers in the DHT network can search for the torrent through DHT search. Therefore, this mechanism eliminates central trackers from the system.

3.5 Comparison and Discussion

The main characteristic of unstructured P2P networks is that the storage of files is completely unrelated to overlay topology. As a result, file search mechanism in such networks essentially amounts to random search. As compared to structured P2P networks, unstructured P2P networks have the following advantages:

- *Resilient to peer dynamics*: Because there is no specified requirement on the overlay structure, unstructured P2P networks are often formed in a random way. In case of peer joining and leaving, an unstructured P2P network can easily reconstruct the overlay. As a comparison, overlay reconstruction in DHT networks is much more complex and expensive, especially in a highly dynamic network with frequent peer joining and leaving.
- *Supporting complex search*: In unstructured P2P networks, peers eventually check their local resource to answer a query. Hence, unstructured P2P networks inherently support complex search based on file meta-data. On the contrary, in DHT networks, each file has a single keyword. File storage and search are all based on this keyword. DHT hence only supports single-keyword search.

Unstructured P2P networks have their own limitations. Its major problem is the low search efficiency, especially for unpopular files. Unpopular files have few copies in the system. Search for such a file may lead to large-scale flooding. Different from it, DHT networks guarantee a certain number of overlay hops for any search, which is very helpful for unpopular file search. In view of the advantages of structured and unstructured networks, some researchers have taken effort to integrate them together [11]. Clearly, there are many challenges in the integration due to their fundamental difference on network structures.

We now give a quick comparison of the above unstructured P2P networks. The centralized approach requires a central server for file management and search, and the supernode-based approach relies on elected or pre-deployed supernodes. Different from them, the fully distributed approach relies on peers for file storage and search, which does not require additional facilities. BitTorrent, a second generation P2P system, requires central trackers to initiate the file sharing process.

According to the system architecture, the approaches have different search methods. In the centralized approach, a query is directly sent to the server. In the supernode-based approach, a query is first sent to a supernode, which forwards the query to other supernodes if needed. In the fully distributed approach, blind or informed flooding is used to answer a query, which may consume much network bandwidth. In BitTorrent, a peer can directly obtain a list of peers sharing the file from the tracker. It hence eliminates the search process for matching peers as in other systems. This is a fundamental difference between BitTorrent and other unstructured P2P systems.

In practice, the supernode-based approach and BitTorrent are the most successful applications. Both the centralized and fully distributed approaches have serious limitations. The centralized approach has poor scalability. It cannot accommodate a large number of peers. But the huge number of peers, and hence huge resource in total, is exactly an attractive aspect of P2P systems. The fully distributed approach does not need any central component, however, it is not highly scalable due to its inefficiency and high bandwidth consumption in search.

As a comparison, the supernode-based approach has shown high scalability and high search efficiency given enough supernodes. It makes practical use of peer heterogeneity. BitTorrent requires central trackers. But the responsibility of a tracker is much lighter than the server in a centralized approach. A tracker only needs to track the peers sharing and downloading the specific file. And torrent files could be put at different trackers. Hence, the scalability of BitTorrent is not an issue in practice. In addition, the newly proposed decentralized tracking mechanism completely eliminates central trackers from the system.

4 Advanced Issues in File Sharing

In this chapter, we discuss two advanced issues in unstructured P2P file sharing systems, i.e,, content replication and system security.

4.1 Content Replication

Search in unstructured P2P networks is essentially random search. Content replication is hence a fundamental issue for search performance. Intuitively, with more replications in the network, it is easier (or faster) to locate a file. On the other side of the coin, more replications take up more storage space. We hence need to explore efficient replication mechanisms to achieve good tradeoff between search efficiency and storage cost.

In the original Gnutella, peers requesting a file make copies of the file. Other systems like FastTrack allow for more proactive replications of files, where a file may be replicated at a peer even though the peer has not requested the file.

Cohen et al. have proposed a network model for proactive replication in unstructured P2P networks [13]. Suppose there are in total m files and n peers in an unstructured P2P network. Each file i $(1 \leq i \leq m)$ is replicated at r_i $(1 \leq r_i \leq n)$ random distinct peers. Clearly, if there is not any limitation on storage space, a straightforward strategy would be to replicate everything everywhere, and search becomes trivial. Hence, we assume that the total amount of storage space over all peers is fixed. If we further assume that each file has a unit size, the total amount of storage space can be computed as $R = \sum_{i=1}^{m} r_i$.

Suppose that file i is requested with the query rate q_i. Here q_i is normalized so that $\sum_{i=1}^{m} q_i = 1$. We consider search of randomly probing peers until the specific file is found. Then, the probability that file i is found on the k'th probe is given by

$$Pr_i(k) = \frac{r_i}{n}\left(1 - \frac{r_i}{n}\right)^{k-1}.$$

Define search size of a query as the number of probes to locate the matching file for the query. For a certain file i, its average search size A_i is simply n/r_i. Hence, the average search size over all files is

$$A = \sum_{i=1}^{m} q_i A_i = n \sum_{i=1}^{m} \frac{q_i}{r_i}. \tag{1}$$

Given the above network model, we can analyze the performance of several different replication mechanisms.

- *Uniform replication*: This replication mechanism equally replicates all files regardless of their popularities. In other words, $r_i = R/m$, $\forall i \in [1,m]$. Hence, the average search size $A_{uniform}$ is given by

  $$A_{uniform} = n \sum_{i=1}^{m} \left(q_i \frac{m}{R}\right) = \frac{nm}{R}.$$

  Clearly, $A_{uniform}$ is independent of the query distribution.
- *Proportional replication*: This replication mechanism replicates more copies for more popular files. In other words, $r_i = Rq_i$, $\forall i \in [1,m]$. Hence, the average search size $A_{proportional}$ is given by

  $$A_{proportional} = n \sum_{i=1}^{m} \frac{q_i}{Rq_i} = \frac{nm}{R}.$$

  Therefore, the proportional and uniform replication mechanisms yield the same average search size, and that average search size is independent of the query distribution. On the other side, the distribution of average search size for a certain file is different for these two mechanisms. In uniform replication, all files have the same average search size (which is nm/R). In proportional replication, $A_i = n/(Rq_i)$. Hence, a popular file with a high query rate will have a small average search size, since it has many replicated copies in the network.

- *Square-root replication*: Given the formula of A as in Equation (1), Cohen et al. prove in [13] that A is minimized when

  $$r_i = \frac{R\sqrt{q_i}}{\sum_{j=1}^{m}\sqrt{q_j}}.$$

  The resulting average search size is

  $$A_{optimal} = \frac{n}{R}\left(\sum_{i=1}^{m}\sqrt{q_i}\right)^2.$$

  This result is later confirmed by Lv et al. through simulations [34].

Given the above replication mechanisms, an immediate question is how to achieve them via a distributed protocol in a decentralized unstructured P2P network. This is easy for uniform and proportional replications. For uniform replication, the system creates a fixed number of copies when the file first enters the system. For proportional replication, the system creates a fixed number of copies every time the file is queried. But the case for square-root replication is much more difficult. The major challenge is that no individual peer sees enough queries to estimate the query rate for a certain file.

Cohen et al. study several ways to achieve square-root replication [13]. Generally, when a query succeeds, the requesting peer creates some number of copies (denoted the number as C) at randomly selected peers. There is also some deletion mechanism to guarantee that in the steady state the creation rate equals the deletion rate. There are various ways to determine C. The first one is called path replication, which sets C to the search size (i.e., the number of peers probed). Another method further improves path replication by requiring a peer to record the value C with each copy. This is called replication with sibling-number memory. A third method is called replication with probe memory. Each peer records the number and the combined search size of probes it sees for each file. It then determines C by collecting this information from a certain number of peers. Clearly, this method needs extra inter-host communication. For more details of these methods and their comparison, please refer to [13].

4.2 Security and Reputation System

Most P2P systems work on the assumption of truthful cooperation among peers. However, in the open environment of the Internet, some participating peers may not cooperate as desired. They may be selfish and unwilling to upload data to others, or they may have abnormal actions such as frequent rebooting which adversely affect their neighbors. More seriously, some peers may launch attacks to disrupt the service or distribute viruses in the overlay network. We call these uncooperative, abnormal or attacking behavior *malicious actions* and the associated peers *malicious peers*.

To detect malicious peers or reward well-behaved ones, a reputation system is often used. In a typical reputation system, each peer is assigned a reputation value according to its history performance. Differentiated services are then provided to peers according to their reputation. We now study two key issues in P2P reputation systems, namely, reputation computing and storage.

4.2.1 Reputation Computing

There are mainly three reputation computing techniques in current P2P networks. We elaborate them as follows.

- *Social Networks* In this approach, all feedbacks available in the network are aggregated to compute peer reputation. It can be further classified into two categories: *separated reputation model* and *correlated reputation model*. In a separated reputation model, only the direct transaction partners (e.g., resource provider/downloader or streaming neighbor) of a peer can express their opinion on the reputation of the peer [14, 16, 21, 27, 28, 35, 41]. A practical example is eBay reputation system (although eBay is not a P2P network) [6]. After each transaction at eBay, the buyer and the seller rate each other with a positive, negative and neutral feedback. The reputation is calculated at a central server by assigning 1 point for each positive feedback, 0 point for each neutral feedback and -1 point for each negative feedback. The reputation of a participant is computed as the sum of its points over a certain period. Considering that peers may lie in their feedbacks, Mekouar et al. propose to monitor suspicious feedbacks [35]. The more suspicious feedbacks a peer generates, the smaller weight in reputation computing its feedback has. Xiong et al. develop a general reputation model, which considers, for example, feedbacks from other peers, credibility factor for the feedback sources, and transaction context factor for discriminating the importance of transactions [46]. In fact, almost all the separated reputation models can be expressed by this generalized model.

  In a correlated reputation model, the reputation of a peer is computed based on the opinion of its direct transaction partners as well as some third-party peers [29, 40]. In this model, a peer *A* who wishes to know the reputation of another peer *B*, can ask some peers (e.g., its neighbors) to provide their opinion on *B* (although some of the peers may not have conducted any transaction with *B*). *A* then combines the opinion from the peers to calculate *B*'s reputation. Clearly, this model is more like our real social networks, where third-party peers besides transaction partners can express their opinion on a peer. But it takes more cost to collect and aggregate third-party opinion.

- *Probabilistic Estimation* This approach uses sampling of the globally available feedbacks to compute peer reputation. It often has some assumptions on peer behavior. For instance, it may assume that a peer is trustworthy with a certain, but unknown probability. And when sharing its own experience with others, a peer may lie with some, again unknown, probability [17]. It then uses well known probabilistic estimation techniques to estimate all unknown parameters.

  Many estimation methods may be used. Despotovic et al. use maximum likelihood estimation [17]. However, it assumes that peers do not collude, which may not be practical in real networks. Mui et al. use Bayesian estimation to assess the future performance of peers based on their history performance, but it uses only direct interaction among peers and does not use third-party opinion [36]. Buchegger et al. take into account third-party opinion, but the approach is empirical rather than theoretically solid [10].

  By using a small portion of the globally available feedbacks, the probabilistic model spends a lower cost in feedback collection than the social network approach. On the other hand, the social network approach can use a complicated reputation model, and is robust to a wide range of malicious actions. But the probabilistic model can be applied to only simple reputation models (due to the difficulties in probabilistic estimation) and is effective to only a few malicious actions. The performance of the two models has been compared in [18]. It has been shown that the probabilistic model performs better for small malicious population, while the social network approach is better when most peers are malicious.
- *Game-Theoretic Model* Different from the above two approaches, the game-theoretic model assumes that peers have rational behavior and uses game theory to build a reputation system. Rational behavior implies that there is an underlying economic model in which utilities are associated with various choices of the peers and that peers act so as to maximize their utilities. Li et al. present a game-theoretic framework for analyzing reputation [30], and Fudenberg et al. offers certain characterizations of the equilibria payoffs in the presence of reputation effects [22].

4.3 Reputation Storage and Retrieval

A basic principle in reputation storage is that the reputation of a peer cannot be locally stored at the peer. Because this has no protection against dishonest peers. A dishonest peer may misreport its reputation value in order to gain rewards or avoid punishments. We summarize several techniques for reputation storage and retrieval in unstructured P2P networks as follows.

- *Centralized* This method uses a powerful server to keep the reputation of all peers. For example, eBay uses a central server to collect and keep all users' reputation [6]. Feedbacks from users are sent to and stored at the server. A query of a user's reputation is also sent to and replied by the server. Similar approaches have been used in [21, 27, 28].

This approach is easy to implement and deploy. Security of a central server is much easier to achieve than that of distributed components in a distributed approach. Furthermore, centralization makes reputation management independent of peer joining and leaving, which greatly simplifies reputation retrieval. However, as discussed in Section 3.1, a centralized approach is not scalable to large P2P networks. And the server forms a single point of failure, making the system vulnerable.

- *Supernode-Based* Mekouar et al. propose a malicious detector algorithm to detect malicious peers in Kazaa-like systems [35]. They assume that supernodes are all trustworthy and maintain reputation information for ordinary peers. Each peer is attached to a unique supernode. All the evaluation results about a peer are maintained at its attached supernode. Supernodes can then enforce differentiated services according to peers' reputation. Note that supernodes in Kazaa are elected according to peers' computational power and edge bandwidth. A supernode may not always be trustworthy. A possible improvement is to deploy some proxies (e.g., a content distribution network) to replace unauthenticated supernodes.

  The security and trustworthiness of pre-deployed proxies are much better than self-elected supernodes.

  The supernode-based approach is an extension of the centralized approach. In the approach, a set of supernodes instead of a single server serve peers. However, to serve a large P2P network, a large number of supernodes are needed, which leads to high implementation and maintenance costs. In addition, the search and load balancing mechanisms among supernodes need to carefully designed.

- *Unstructured Overlay* XREP uses a polling algorithm to help peers choose reliable resource in Gnutella-like networks [14, 16]. It consists of four operations: resource searching, vote polling, vote evaluation and resource downloading (as shown in Figure 2). The first operation is similar to searching in Gnutella. A peer broadcasts to all its neighbors a *Query* message. If a peer receiving a *Query* message has the matching file, it responds with a *QueryHit* message (as shown in Figure 2(a)). In the next operation, upon receiving *QueryHit* messages, the original searching peer selects the best matching resource among all possible choices. It then polls other peers using an encrypted *Poll* message to enquire their opinion on the resource or the resource provider. To achieve that, each XREP peer maintains information for its own experience on resource and other peers. Upon receiving a *Poll* message, each peer checks its experience data. If there is any information about the resource or the provider indicated by the *Poll* message, the peer sends its vote to the polling peer with an encrypted *PollReply* message (as shown in Figure 2(b)).

  In the third operation, the polling peer collects a set of votes and evaluates the votes. It first decrypts the votes and discards corrupt ones. Then it analyzes voters' IPs and detects cliques of dummy or controlled votes. After that, it randomly selects a set of votes and directly contacts them with a *TrustVote* message. Each contacted voter is required to send a *VoteReply* message for vote confirmation.

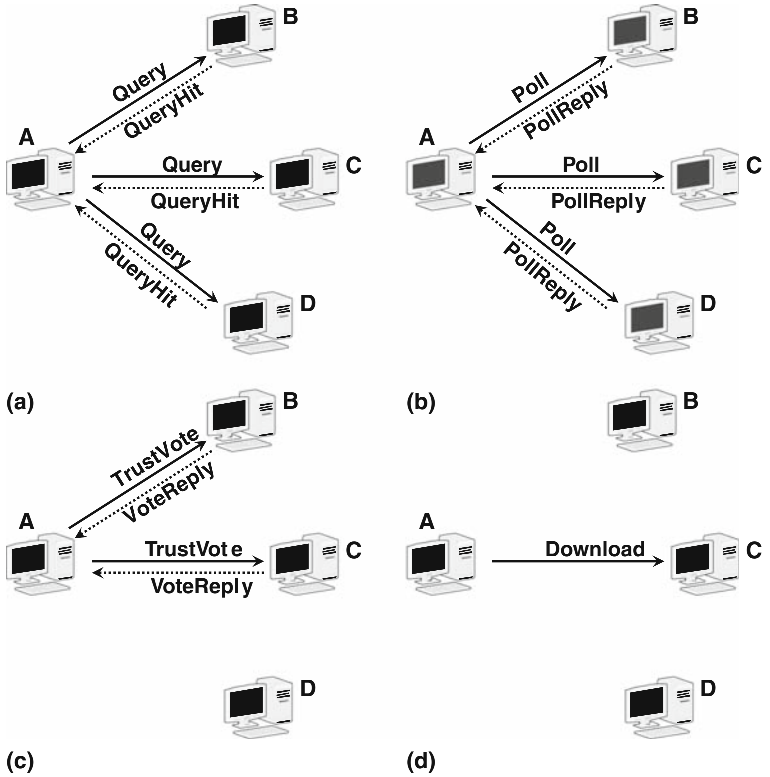


Fig. 2 Operations in XREP. (**a**) Resource searching. (**b**) Vote polling. (**c**) Vote evaluation. (**d**) Resource downloading.

This forces potential attackers to pay the cost of using real IPs as false witness (e.g., shilling attack). After this checking process, the polling peer can obtain the reputation of the resource or the provider, and finally decides to download it (as shown in Figure 2(d)). If the polling peer decides not to download from the current provider, it can repeat the voting process on another resource.

More examples of using unstructured overlays include NICE reputation model [40] and TrustMe [41]. All the approaches based on unstructured overlays have the security concern. Messages may be intercepted or blocked during transmission, and voting is vulnerable to collusion among peers. Therefore, no secure reputation computing or delivery can be guaranteed. Furthermore, searching or voting on an unstructured overlay is based on flooding, which incurs heavy traffic in the network. For example, in XREP, *Poll* messages are broadcast throughout the network each time a peer needs to find out the reputation of a resource or a provider. This in turn affects the scalability of the system because an increase in the number of peers can potentially lead to an exponential increase in the number of *Poll* messages and responses.

5 Other Applications of Unstructured P2P Networks

Besides file sharing, unstructured P2P networks have been widely used in other network applications. In this chapter, we discuss two example applications based on unstructured P2P networks: media streaming and VoIP.

5.1 Media Streaming: CoolStreaming

With the popularity of broadband Internet access and P2P technologies, media streaming has gone through rapid growth. Typical services include on-demand video streaming that allows users to choose and watch favorite movies anytime, Internet Protocol television (IPTV) and live streaming that provide live TV service.

In a P2P streaming system, one or multiple supplying peers who have all or part of the requested media can forward the data to the requesting peers. In turn, the requesting peers can become supplying peers for other requesting peers. Because each peer contributes its own resource (storage and network bandwidth) to the system, the whole system's capacity is vastly amplified compared to the traditional client-server architecture. The major challenges in P2P streaming include [47]:

- *Peer dynamics*: In P2P networks, peers do not always stay online in the system. Supplying peers might unexpectedly crash or leave. In this case, the requesting peers need to find new supplying peers to replace the failed ones. Therefore, the system should be highly robust to withstand such peer dynamics.
- *Limited and dynamic peer bandwidth*: Unlike powerful video servers, peers have limited bandwidth capacities. Each supplying peer might only be able to support a few requesting peers (or multiple supplying peers are required to support one requesting peer). Also, the available bandwidth of supplying peers might fluctuate. Hence, the system should be able to adaptively adjust each supplying peer's sending rate to keep the streaming quality at requesting peers unaffected.

CoolStreaming is one of the popular softwares that provide live streaming service through P2P networks [32, 51]. As the first P2P-based streaming system that attracts a remarkable amount of users, CoolStreaming has several notable features: (1) Intelligent scheduling algorithm that copes well with the bandwidth heterogeneity of peers; (2) Swarm-style architecture that builds a gossip overlay to distribute contents.

In CoolStreaming, a peer needs to search for some other peers called partners, with which the peer collaborates to download streaming contents. To construct the overlay network among partners, the system employs the Scalable Gossip Membership protocol (SCAM) [23]. The SCAM protocol is fully distributed and scalable, and can provide a uniform partial view of the whole system at each peer. Based on it, CoolStreaming forms an unstructured overlay among partners, which achieves excellent resilience against random failure and enables decentralized operation. Such an overlay is similar to the BitTorrent network [37].

In detail, a newly joined peer first contacts the boot-strap server, which responds with a randomized list of the currently active peers. The newly joined peer stores this list in its cache and randomly selects a few peers from the cache to establish TCP connections, i.e., partnership. Once the partnership is established between a pair of peers, they exchange and update their cache contents. The maximum number of peers in the cache is usually on the order of $\log N$, where N is the total number of peers in the system.

Content delivery in CoolStreaming is achieved as follows. The video stream is divided into segments of uniform length, and the availability of the segments in a peer's buffer is represented by a buffer map (BM). Note that the aforementioned cache is used to store system management data and the buffer here is used to store media content. Each peer periodically exchanges its BM with its partners. Upon receiving the BM from a peer x, a peer y chooses the data segments it does not possess and sends a request indicating the demanded data segments to x. Then, x delivers the requested data segments to y. Clearly, if a peer has multiple partners, the peer has to select one partner for each of its missing segments. This selection problem is called packet scheduling. Interested readers may refer to [48, 51] for more details.

Figure 3 shows an example of the CoolStreaming network. The system consists of a video source, a boot-strap server and four peers (labeled from A to D). The video source provides the complete video content for the system. In the figure, it delivers video content to peers B and D. The boot-strap server helps peers join the system. The real lines with double-ended arrows show the partnership between peers. For example, peer C has three partners A, B and D.

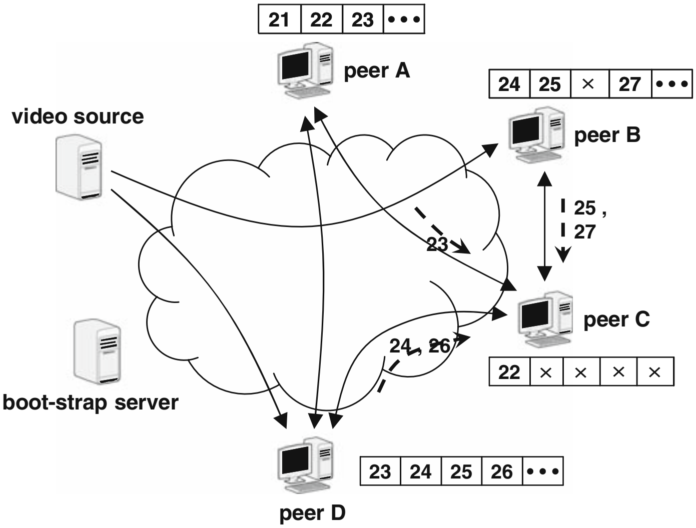


Fig. 3 An example of the CoolStreaming network

The square table along a peer shows part of the peer's BM. For example, peer *B* possesses segments 24, 25 and 27 in its buffer. But it does not possess segment 26. Missing of a segment is denoted as "×" in the BM. The subsequent segments after the 27th are not shown in *B*'s BM, which is denoted as "...". Note that peers have different starting segments in their BMs. This is because they have different play points (due to, for example, different end-to-end delay, or different downloading bandwidth).

The figure also shows an example of packet scheduling at peer *C*. Suppose *C* has known the BMs of its partners. It then uses the packet scheduling algorithm to decide to fetch which missing segment from which partner. In the figure, it requests segment 23 from peer *A*, segments 25 and 27 from peer *B*, and segments 24 and 26 from peer *D*, respectively. This scheduling imposes similar uploading load on its partners.

In summary, the advantages of CoolStreaming include:

- *Fully distributed and scalable*: Each peer distributedly joins the system and selects partners. During content delivery, a peer exchanges information with only a few partners. The whole system is hence highly scalable. This allows CoolStreaming to accommodate a large number of peers. As reported in [31, 45], it recorded over $80,000$ concurrent users with an average bit rate of 400 Kbps.
- *Highly resilient to peer dynamics*: The use of multiple partners and the corresponding multiple path delivery at peers provide high system resilience. If some partners of a peer unexpectedly leave, the peer can still retrieve data from other partners.

On the other side of the coin, CoolStreaming has some limitations due to its design.

- *High control overhead*: CoolStreaming has high control overhead for the gossip mesh maintenance and data distribution. A peer has to frequently communicate with its current and potential partners to keep a highly refreshed overlay. Otherwise, a peer cannot quickly find new partners in case of partner leaving. Furthermore, a peer has to periodically exchange BM with its partners. This further increases the control overhead.
- *High end-to-end delay*: Peers in CoolStreaming often encounter high end-to-end delay. This is sometimes fatal to the quality of service, for example, for people gambling during a live soccer play. The reason is two-fold. Firstly, the gossip mesh is randomly formed without considering peer locality. The mesh often contains long connections between faraway peers. Secondly, the procedure of BM exchange increases the delay. Before a peer can download a segment, it has to first obtain some valid BMs and then send a transmission request to the selected partner.

There has been much effort to improve CoolStreaming and study new streaming systems. For example, Ren et al. explore how to build a low-delay overlay mesh among peers by considering peer locality [39]. Zhang et al. propose new data delivery mechanism to reduce delay due to BM exchange [49]. Meanwhile, many other

P2P streaming systems are proposed and evaluated, for example, AnySee [33] and GridMedia [43] for live streaming, P2VoD [20] and oStream [15] for video-on-demand.

5.2 VoIP: Skype

VoIP service (also referred to as IP telephony, Internet telephony, or voice over broadband) allows for the transmission of voice through the Internet. Traditional telephone lines use the Public Switched Telephone Network (PSTN), which works on circuit switching and connects callers to receivers through electrical circuits. VoIP is based on packet switching, where data packets are carried across the Internet, from one computer to either another computer or a PSTN telephone. There have been many VoIP softwares in the market, for example, Skype [7], Google talk and AOL Instant Messenger. VoIP operates in different forms. Here are a few examples.

- *Computer to computer*: This is the most frequently used way of VoIP. Each computer should be equipped with a sound card, a headset consisting of earphones and microphone, and some VoIP software. Most VoIP softwares provide free service for one computer to connect to any other computer running the same software.
- *Computer to phone*: Some VoIP softwares allow users to call regular telephone landlines and mobile phones from a computer. This service is usually not free, but its cost is often lower than traditional telephone charges.
- *Phone to phone*: There are two ways to make a phone-to-phone connection: (1) Use a regular phone plugged into an Analog Terminal Adaptor (ATA), which in turn connects to the Internet. (2) Use a VoIP phone that connects to the Internet.

Figure 4 shows the above three forms of VoIP systems. Current VoIP softwares also provide many other features such as instant messaging, file transfer and video conferencing.

We now study Skype as an example VoIP system. Skype launched its service in 2003 and experienced rapid growth after that. In October 2005, it was purchased by eBay. According to eBay quarterly report, as of December 31, 2007, Skype had accumulated 276 million user accounts.

Skype uses a proprietary and closed-source protocol. Numerous attempts have been undertaken to study and reverse engineer the protocol [9, 24, 44]. It is believed that Skype uses a Kazaa-like P2P network. Both companies were founded by the same individuals and much of the technology in Skype was originally developed for Kazaa. This is further confirmed by comparing Skype and Kazaa traffic on the packet level [9].

The Skype system consists of three main entities: supernodes, ordinary peers and login servers. Supernodes are elected from ordinary peers which have high bandwidth, adequate processing power and no firewall. There are also a number of

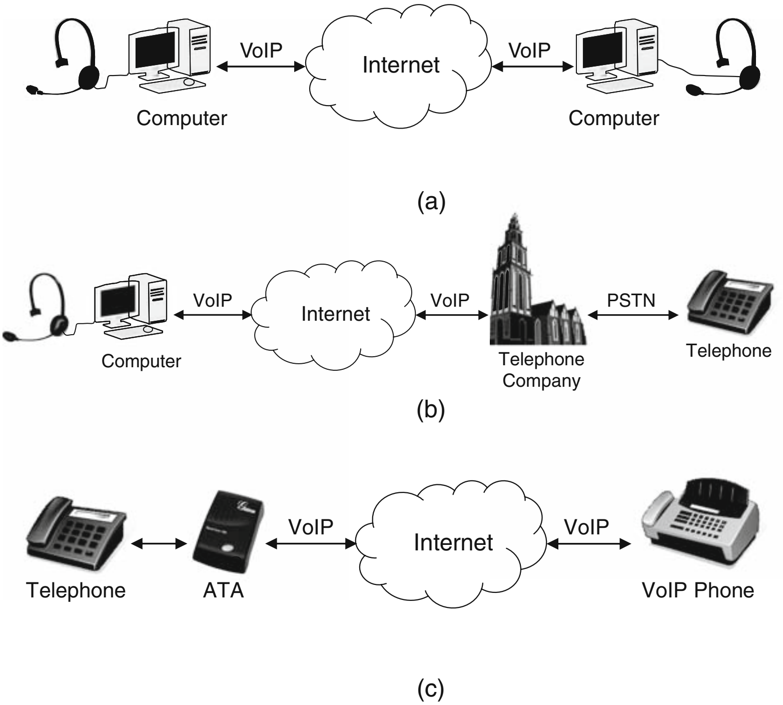


Fig. 4 Different forms of VoIP systems. (**a**) Computer-to-computer. (**b**) Computer-to-phone. (**c**) Phone-to-phone.

pre-deployed supernodes in the system, which keep staying online.Supernodes maintain an overlay network among themselves.

When login, a peer first connects to the Skype network. This is achieved by establishing a TCP connection and exchanging information with a supernode. To do that, the new peer contacts a default supernode to obtain a list of supernodes. The peer then caches the supernode list and regularly refreshes it. If the TCP connection fails, the peer tries to connect to some bootstrap IP addresses hardcoded in the client software (in version 1.2), or simply generates a login failure report (in version 0.97) [9]. After connecting to the Skype network, the peer authenticates the username and password with the Skype login server. An obfuscated list of servers has been hardcoded in the client software. Then the peer advertises its presence to other peers, determines the type of network address translator (NAT) and firewall it is behind and discovers peers that have public IP addresses.

When a peer *A* wants to call another peer *B*, *A* first queries some supernodes for *B*'s address. Through a search among supernodes, *A* can obtain *B*'s address. If both *A* and *B* are publicly reachable, *A* sets up a connection to *B* and directly exchanges voice traffic with *B*. If any participating peer is behind firewall or NAT, it sets up a connection to a supernode. The transmission of voice traffic is then relayed by the supernode. Sometimes Skype also routes calls through ordinary peers to ease the crossing of Symmetric NATs and firewalls.

The use of supernodes for communication has many advantages. Firstly, peers behind NATs or firewalls are not publicly reachable. Through public supernodes, such peers can be reached and called. Secondly, supernodes can manage multi-user sessions such as conferencing. They can store messages from different users and accordingly forward them. On the other hand, this network structure puts heavy burden on supernodes, leading to unfair work loads among peers.

6 Mobile Unstructured P2P Networks

With the advance of mobile devices and technologies, data distribution among handhelds has become a reality. Wireless networks share many similar features with P2P networks, e.g., distributed network structure and dynamic network topology. In this chapter, we discuss how to implement unstructured P2P networks over wireless networks.

6.1 Characteristics of Mobile Wireless Networks

There are different types of wireless networks. We list a few as examples.

- *Mobile* ad-hoc *networks (MANETs)*: MANETs do not have any infrastructure support (such as base stations, access points or remote servers). All network functions are performed by the nodes forming the network, which often have high mobility and low processing power. The resulting topology is then dynamic and unstable. An example of MANET is a vehicular ad-hoc network, where wireless devices in vehicles interact with each other while moving at high speed.
- *Wireless sensor networks (WSNs)*: A WSN uses spatially distributed autonomous sensors to monitor physical or environmental conditions (such as temperature, pressure and pollutants). The sensors are low-profile devices with very limited processing power, memory and battery. The primary concern of a sensor network is its lifetime, therefore protocols for sensor nodes often focus on power conservation.
- *Wireless mesh networks (WMNs)*: A WMN consists of gateways, mesh points and wireless end-users. Gateways provide access to the Internet. Mesh points are small devices with limited processing power and memory, which are often mounted on lamp posts or rooftops. Each mesh point is associated with a certain gateway, and forwards Internet-bound traffic from associated users to the gateway. Mesh points hence extend the network coverage of gateways. While end-users are mobile and may switch their associated mesh points at any time, gateways and mesh points are generally stationary. A WMN is reliable and offers redundancy. If some mesh points fail, the rest mesh points can self-form a new mesh to continue communications.

While these wireless networks have different forms and usage, they have some common characteristics, which differentiate them from the wired Internet.

- *Wireless transmission*: Packet transmission on wireless channels is based on broadcast. And the wireless communication medium is accessible to any entity with the appropriate equipment and adequate resource. As a comparison, Internet transmission is mainly based on unicast, and sometimes multicast. In addition, wireless channels have limited bandwidth and each hop has a certain transmission range.
- *Lack of routing infrastructure*: In the wired network, routing is readily available. But in wireless networks, routing is a non-trivial issue. Two important issues in wireless routing are high maintenance overhead of routes and inefficient bandwidth utility due to long routes.
- *Low processing capability, memory capacity and energy power*: Wireless devices are often small handheld devices. They are normally low in processing capability and limited in memory capacity and energy power.

6.2 Approaches for Mobile Unstructured P2P Networks

We select MANET as an example of wireless networks. A MANET is similar to a P2P network in the following aspects.

- Both systems are distributed. In MANETs, nodes usually have low local resource and cannot serve as servers. A scalable P2P network should also be fully distributed in order to accommodate a large number of peers.
- Their topologies frequently change because of peer on/off or mobility.
- Nodes or peers in the systems have similar functionalities. They cooperate to route queries and rely messages. In both systems, flooding or broadcasting is employed to some extent for data exchange or routing among peers/nodes.

There are also some differences between P2P and MANET. For example, P2P works on the application layer in the protocol stack, while MANET focuses on the network and lower layers. As mentioned above, peers in MANET are mobile and constrained by limited energy, bandwidth and computational power, which is not a big concern in P2P systems over the Internet. And MANET uses physical broadcast but P2P uses physical unicast.

When deploying P2P networks over MANETs, the major problem is how to quickly find the requested data in spite of the mobility and the scarcity of power and bandwidth in the underlying MANET. Ding et al. propose several ways for such purpose [19]. A straightforward approach is to simply implement the P2P flooding mechanism over MANET on-demand routing protocols. That is, a query message is flooded to every virtual neighboring peer in the P2P overlay. As two virtual neighboring peers may be multiple hops away in the MANET, we need to obtain the underlying route between them. Then the network routing request is also broadcast at the network layer.

We show an example in Figure 5(a). Circles in the figure represent mobile nodes in a MANET. Two mobile nodes connected by a real line are within the transmission range of each other in the MANET. Shadowed rectangles (labeled as A, E and F) represent peers in an unstructured P2P network. Two peers connected by a dashed line are neighboring peers in the P2P network. From the figure, A and E are neighboring peers in the P2P network. So are E and F. In this example, if E wants to search for some file, it floods its query to its neighboring peers in the P2P network, i.e., A and F. Different from the wired Internet, the route from E to A (or to F) is not readily available and needs to be discovered through the MANET routing protocol. A broadcast for routing on the network layer is then necessary.

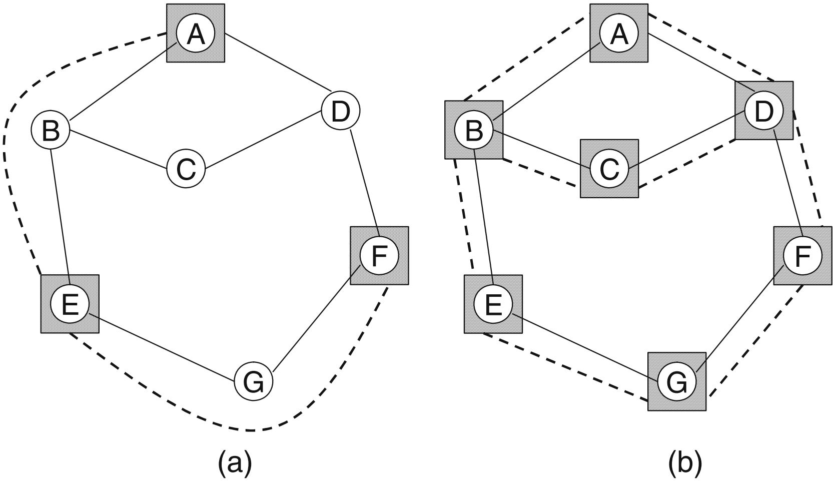


Fig. 5 Two examples of unstructured P2P network over MANET. Circles represent mobile nodes in a MANET. Two mobile nodes connected by a real line are within the transmission range of each other in the MANET. Shadowed rectangles represent peers in an unstructured P2P network. Two peers connected by a dashed line are neighboring peers in the P2P network

This approach is easy to implement. But it is not scalable due to the double-layer broadcasts. Two neighboring peers in the P2P network might be physically far away from each other. Hence, flooding in the P2P network might be expensive and inefficient. Therefore, this approach is only applicable to small MANETs.

Another approach is to map the MANET network to a P2P overlay network. Each MANET mobile node can be regarded as a peer in the P2P network. A pair of neighboring nodes in MANETs (within the transmission range of each other) correspond to a pair of neighboring peers in P2P. As wireless networks always employ broadcast to transmit data, the MANET routing protocol and the P2P flooding protocol can be implemented by one-pass broadcast.

For example, in Figure 5(b), if E wants to search for some file, it broadcasts a query to its neighboring mobile nodes B and G, which are also its neighboring peers in the P2P network. Upon receiving the query message, *B* and *G* check their local file resource and continue broadcasting the message if needed.

Clearly, this method is more efficient than the first one. It directly finds the shortest path between the file source and the original requester. But the whole network is still flooded by query messages, which imposes heavy burden on communication bandwidth and power supply for mobile nodes. So it still cannot work for large MANETs.

There are many other approaches for building P2P networks over wireless networks. For example, Hora et al. study how to reduce energy consumption and delay in mobile P2P networks [26]. Akon et al. propose a novel gossip protocol to build and maintain a P2P network over a mobile wireless network [8]. Interested readers can refer to these papers for more details.

7 Conclusion

We discuss in this chapter unstructured P2P networks, one type of widely used P2P networks. In an unstructured P2P network, peers form an overlay (often in a random way) to exchange or relay data. Different from structured P2P networks, unstructured P2P networks do not impose any structure on the overlays. As a result, data storage is unrelated to the overlay, and file search essentially amounts to random search.

We discuss several applications of unstructured P2P networks, including file sharing, media streaming and VoIP. For the first one, we classify existing approaches into four categories and explore representative examples from each category. We also discuss two advanced issues in file sharing, i.e., content replication and reputation system. For the other two applications, we study their key challenges. We select one state-of-the-art approach for each application and analyze its system design.

We also investigate how to implement unstructured P2P networks over wireless networks. The characteristics of wireless networks (such as limited bandwidth and transmission range) impose new challenges for building P2P networks. We study the state-of-the-art approaches and discuss their advantages and limitations.

References

1. Gnutella. URL `http://gnutella.wego.com`
2. Kazaa. URL `http://www.kazaa.com`
3. BitTorrent. URL `http://www.bittorrent.com`
4. Napster. URL `http://www.napster.com`
5. Grokster. URL `http://www.grokster.com`
6. eBay. URL `http://www.ebay.com`
7. Skype. URL `http://www.skype.com/`

8. Akon, M., Shen, X., Naik, S., Singh, A., Zhang, Q.: An inexpensive unstructured platform for wireless mobile peer-to-peer networks. Peer-to-Peer Networking and Applications **1**(1), 75–90 (2008)
9. Baset, S.A., Schulzrinne, H.: An analysis of the Skype peer-to-peer Internet telephony protocol. In: Proc. IEEE INFOCOM'06 (2006)
10. Buchegger, S., Boudec, J.Y.L.: A robust reputation system for P2P and mobile ad-hoc networks. In: Proc. P2PEcon'04 (2004)
11. Castro, M., Costa, M., Rowstron, A.: Should we build Gnutella on a structured overlay? SIGCOMM Computer Communication Review **34**(1), 131–136 (2004)
12. Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S.: Making Gnutella-like P2P systems scalable. In: Proc. ACM SIGCOMM'03, pp. 407–418 (2003)
13. Cohen, E., Shenker, S.: Replication strategies in unstructured peer-to-peer networks. In: Proc. ACM SIGCOMM'02, pp. 177–190 (2002)
14. Cornelli, F., Damiani, E., Vimercati, S., Paraboschi, S., Samarati, P.: Choosing reputable servents in a P2P network. In: Proc. ACM WWW'02, pp. 376–386 (2002)
15. Cui, Y., Li, B., Nahrstedt, K.: oStream: Asynchronous streaming multicast in application-layer overlay networks. IEEE Journal on Selected Areas in Communications **22**(1), 91–106 (2004)
16. Damiani, E., Vimercati, S., Paraboschi, S., Samarati, P., Violante, F.: A reputation-based approach for choosing reliable resources in peer-to-peer networks. In: Proc. ACM CCS'02, pp. 207–216 (2002)
17. Despotovic, Z., Aberer, K.: Maximum likelihood estimation of peers performance in P2P networks. In: Proc. P2PEcon'04 (2004)
18. Despotovic, Z., Aberer, K.: P2P reputation management: Probabilistic estimation vs. social networks. Computer Networks **50**(4), 485–500 (2006)
19. Ding, G., Bhargava, B.: Peer-to-peer file-sharing over mobile ad hoc networks. In: Proc. IEEE PercomW'04, pp. 104–108 (2004)
20. Do, T., Hua, K.A., Tantaoui, M.: P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment. In: Proc. IEEE ICC'04, pp. 1467–1472 (2004)
21. Dragovic, B., Kotsovinos, E., Hand, S., Pietzuch, P.: XenoTrust: Event-based distributed trust management. In: Proc. DEXA'03, pp. 410–414 (2003)
22. Fudenberg, D., Levine, D.: Reputation and equilibrium selection in games with a patient player. Econometrica **57**(4), 759–778 (1989)
23. Ganesh, A.J., Kermarrec, A.M., Massoulie, L.: SCAMP: Peer-to-peer lightweight membership service for large-scale group communication. In: Proc. NGC'01 (2001)
24. Guha, S., Daswani, N., Jain, R.: An experimental study of the Skype peer-to-peer VoIP system. In: Proc. IPTPS'06 (2006)
25. Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K.W.: A measurement study of a large-scale P2P IPTV system. IEEE Transactions on Multimedia **9**(8), 1672–1687 (2007)
26. da Hora, D.N., Macedo, D.F., Nogueira, J.M.S., Pujolle, G.: Optimizing peer-to-peer content discovery over wireless mobile ad hoc networks. In: Proc. IFIP/IEEE MWCN'07, pp. 86–90 (2007)
27. Jin, X., Chan, S.H.G., Yiu, W.P.K., Xiong, Y., Zhang, Q.: Detecting malicious hosts in the presence of lying hosts in peer-to-peer streaming. In: Proc. IEEE ICME'06, pp. 1537–1540 (2006)
28. Jun, S., Ahamad, M., Xu, J.: Robust information dissemination in uncooperative environments. In: Proc. IEEE ICDCS'05, pp. 293–302 (2005)
29. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The EigenTrust algorithm for reputation management in P2P networks. In: Proc. WWW'03, pp. 640–651 (2003)
30. Kreps, D., Wilson, R.: Reputation and imperfect information. Journal of Economic Theory **27**(2), 253–279 (1982)
31. Li, B., Xie, S., Keung, G., Liu, J., Stoica, I., Zhang, H., Zhang, X.: An empirical study of the Coolstreaming+ system. IEEE Journal on Selected Areas in Communications **25**(9), 1627–1639 (2007)

32. Li, B., Xie, S., Qu, Y., Keung, G., Lin, C., Liu, J., Zhang, X.: Inside the new Coolstreaming: Principles, measurements and performance implications. In: Proc. IEEE INFOCOM'08, pp. 1031–1039 (2008)
33. Liao, X., Jin, H., Liu, Y., Ni, L.M., Deng, D.: Anysee: Peer-to-peer live streaming. In: Proc. IEEE INFOCOM'06 (2006)
34. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: Proc. ICS '02, pp. 84–95 (2002)
35. Mekouar, L., Iraqi, Y., Boutaba, R.: Peer-to-peer's most wanted: Malicious peers. Computer Networks **50**(4), 545–562 (2006)
36. Mui, L., Mohtashemi, M., Halberstadt, A.: A computational model of trust and reputation. In: Proc. HICSS'02, pp. 2431–2439 (2002)
37. Qiu, D., Srikant, R.: Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In: Proc. ACM SIGCOMM'04, pp. 367–378 (2004)
38. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: Proc. ACM SIGCOMM'01, pp. 161–172 (2001)
39. Ren, D.N., Li, Y.T.H., Chan, S.H.G.: On reducing mesh delay for peer-to-peer live streaming. In: Proc. IEEE INFOCOM'08, pp. 1058–1066 (2008)
40. Sherwood, R., Lee, S., Bhattacharjee, B.: Cooperative peer groups in NICE. Computer Networks **50**(4), 523–544 (2006)
41. Singh, A., Liu, L.: TrustMe: Anonymous management of trust relationships in decentralized P2P systems. In: Proc. IEEE P2P'03, pp. 142–149 (2003)
42. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for Internet applications. In: Proc. ACM SIGCOMM'01, pp. 149–160 (2001)
43. Tang, Y., Luo, J.G., Zhang, Q., Zhang, M., Yang, S.Q.: Deploying P2P networks for large-scale live video-streaming service. IEEE Communications Magazine **45**(6), 100–106 (2007)
44. Xie, H., Yang, Y.R.: A measurement-based study of the Skype peer-to-peer VoIP performance. In: Proc. IPTPS'07 (2007)
45. Xie, S., Li, B., Keung, G., Zhang, X.: Coolstreaming: Design, theory, and practice. IEEE Transactions on Multimedia **9**(8), 1661–1671 (2007)
46. Xiong, L., Liu, L.: PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities. IEEE Transactions on Knowledge and Data Engineering **16**(7), 843–857 (2004)
47. Yiu, W.P.K., Jin, X., Chan, S.H.G.: Challenges and approaches in large-scale P2P media streaming. IEEE Multimedia **14**(2), 50–59 (2007)
48. Zhang, M., Xiong, Y., Zhang, Q., Yang, S.: On the optimal scheduling for media streaming in data-driven overlay networks. In: Proc. IEEE Globecom'06 (2006)
49. Zhang, M., Zhang, Q., Sun, L., Yang, S.: Understanding the power of pull-based streaming protocol: Can we do better? IEEE Journal on Selected Areas in Communications **25**(9), 1678–1694 (2007)
50. Zhang, X., Liu, J., Li, B.: On large scale peer-to-peer video streaming: Experiments and empirical studies. In: Proc. IEEE MMSP'05 (2005)
51. Zhang, X., Liu, J., Li, B., Yum, T.S.P.: CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming. In: Proc. IEEE INFOCOM'05, pp. 2102–2111 (2005)

Exchanging Peers to Establish P2P Networks

Mursalin Akon, Mohammad Towhidul Islam, Xuemin (Sherman) Shen,
and Ajit Singh

Abstract Structure-wise, P2P networks can be divided into two major categories: (1) structured and (2) unstructured. In this chapter, we survey a group of unstructured P2P networks. This group of networks employs a gossip or epidemic protocol to maintain the members of the network and during a gossip, peers exchange a subset of their neighbors with each other. It is reported that this kind of networks are scalable, robust and resilient to severe network failure, at the same time very inexpensive to operate.

1 Introduction

In the Internet world, Peer-to-peer (P2P) computing is an emerging model for service distribution. In contrast to the traditional client-server and push models, the P2P model is characterized by decentralization, self-organization, cooperation among peers and heterogeneity. In P2P model, participant peers work together to reach a common goal. According to this model, an overlay networks is created among peers, and peers bind each other in a logical neighbor relationship. Most often, such an overlay network remains as a pure virtual entity over the physical network.

M. Akon
Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, e-mail: mmakon@uwaterloo.ca

M. Islam
Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, e-mail: mtislam@uwaterloo.ca

X. Shen
Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, e-mail: xshen@bbcr.uwaterloo.ca

A. Singh
Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, e-mail: asingh@uwaterloo.ca

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_6, © Springer Science+Business Media, LLC 2010

Based on the nature of the binding or relationship among peers, P2P networks is often distinguished into two categories: (1) structured and (2) unstructured. The topology of the members in a structured network is ruled by explicit constraints. Contents are distributed among the members using either some hints [4] or the topology of the members [13, 15, 20, 22]. The unstructured networks do not have a predetermined scheme to bind peers in neighbor relationship.

In this chapter, we explore a specific kind of unstructured P2P networks that create the networks by interchanging a subset of neighbors. The heart of these networks is a gossip protocol where a peer talks to a neighboring peer about other neighbors. The key idea is to introduce randomness in the system and eliminate all sorts of global administration. This helps in sustaining dynamics of P2P networks, i.e., constant join and leave of peers, and shows the capacity of self-healing in severe network disasters.

We divide this chapter into several sections. We present four different P2P networks which are created and maintained using the concept of exchanging peers. These networks are PROOFS, CYCLON, IPPS, and Gradient Topology Network, and are elaborated in Sections 2, 3, 4 and 5, respectively. In Section 6, we end our discussion with concluding remarks and future research discussions.

2 The PROOFS Network

Stavrou et al. propose P2P Randomized Overlays to Obviate Flash-crowd Symptoms or PROOFS [19] to manage Internet flash crowd. Internet flash crowd takes place when an object reaches its peak popularity. During the pinnacle popularity of an object, the number of requests may become so tremendous that a significant number of the users are left out and thus the objects become unavailable to them.

2.1 Evolution

Previous solutions to the problem of flash crowd are either administration-wise impractical or expensive. Such solutions are provisioning accessibility based on peak demand, creating dynamic hosts with dynamic domain names, etc. The traditional solution of replicating servers increases availability but involves extensive amount of communication efforts to exchange information and synchronize data with each other, and thus is not scalable. In contrast, the PROOFS network provides a scalable solution that can reliably deliver objects which are extremely popular to be handled by standard delivery techniques.

There exist other structured P2P content distribution systems – such as CAN [13], Chord [20], Past [6, 15], Tapestry [22], Pastry [15], SCRIBE [3, 16], etc. These systems facilitate easy and inexpensive object searches. However, objects or contents with explosive popularity impose the same difficulties of traditional hosts. Moreover, such networks performs poorly in highly dynamic environments where participants or peers join or leave at a very high rate. In contrary, PROOFS networks are robust and possess self-healing property.

2.2 Components

The design of the PROOFS network consists of two components: (1) *client* and (2) *bootstrap server*. The clients[1] are participants of the P2P overlay and they interact with each other to search and retrieve (popular) objects. A bootstrap server caches a finite set of recently joined peers or clients and introduce them to a joining peer. Thus a joining peer becomes familiar with the network.

2.3 Protocols

A PROOFS network is created and maintained with help of two protocols – (1) *ConstructOverlay* and (2) *LocateObject*.

2.3.1 `ConstructOverlay`

The `ConstructOverlay` begins when a peer joins the PROOFS network and obtains initial neighbors from a bootstrap server. During lifetime, a peer maintains at most C number of neighbors. Here, if peer p includes q as its neighbor, p is allowed to be the initiator of a communication involving q. Peer q can communicate with p by only responding to p, unless p is also a neighbor of q. Each peer, in the network, performs a periodic operation called *exchange*.[2] The exchange operation at peer p is described in Algorithm 1.

Algorithm 1: The exchange operation of PROOFS

```
1 $p$ finds $\mathscr{E}_p \subseteq \mathscr{N}_p$, where each element of $\mathscr{E}_p$ is selected randomly
2 $p$ selects $q \in \mathscr{E}_p$ randomly
3 $p$ sends a request to $q$ with the set $(\mathscr{E}_p \cup \{p\}) \backslash \{q\}$ to be a participant
4 if $q$ agrees to the request then
5     $q$ finds $\mathscr{E}_q \subseteq \mathscr{N}_q$, where each element of $\mathscr{E}_q$ is selected randomly
6     $q$ sends a respond back to $p$ with the set $\mathscr{E}_q$
7     $p$ updates $\mathscr{N}_p \leftarrow (\mathscr{N}_p \backslash \mathscr{E}_p) \cup \mathscr{E}_q$
8     $q$ updates $\mathscr{N}_q \leftarrow (\mathscr{N}_q \backslash \mathscr{E}_q) \cup \mathscr{E}_p$
```

In the algorithm, $\mathscr{N}_p$ is used to designate the neighbor set of peer p. As evident, the sets of peers exchanged and the participant is chosen entirely randomly (line 1, 2,

[1] The term *client* does not refer to the clients in traditional client-server model. In P2P networks, such components are designated as peers. Though the authors used the term client solely, we use client and peer interchangeably to be in symphony with rest of the chapter.

[2] The authors [19] used the term *shuffle* to designate this operation. We use the term *exchange* to identify shuffle in PROOFS and other similar operations in other networks. The exchange operation of PROOFS is used as the basis of other exchange operations discussed in this chapter.

and 5). While updating the neighbor lists (lines 7 and 8) caution should be exercised so that – (1) each neighbor does not exist more than once in a list, (2) a peers is not included as its own neighbor, (3) the number of neighbors are always bounded by C and if not, new members are added until the bound C is reached. Figure 1 shows an example of an exchange operation in PROOFS network. In the figure p initiates the operation with q.

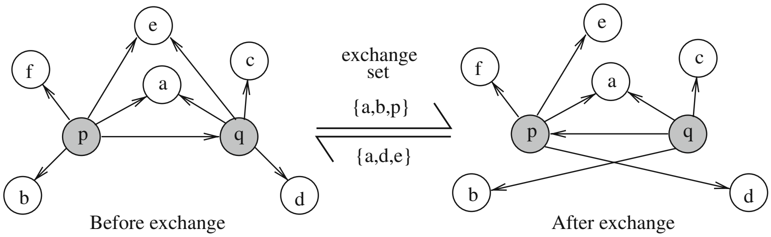


Fig. 1 An example of exchange operation in PROOFS

Note that, a request to participate in an exchange may be denied or even ignored. When the initiating peer does not receive a response back and times out, it assumes that the target peer is no longer maintaining membership with the network, i.e., left the network. A request is declined by an active peer, if and only if another exchange operation is pending. In case of an unsuccessful exchange operation, the initiator waits for a random time amount, picked from a uniform distribution and re-initiates another exchange.

Like other communication networks, in PROOFS, the neighborhood relationship is represented with a graph with directed edges. Peer q being the neighbor of p is indicated by the directed edge from p to q. An exchange operation introduces new edge and may eliminate existing edges, and always reverses the direction of the edge between participants.

2.3.2 `LocateObject`

When a peer wants to retrieve an object in a PROOFS network, it initiates a query using the `LocateObject` protocol. A query includes four vital piece of information – (1) identification of the requested object, (2) time to live (TTL), (3) a fanout value (f), and (4) the address of the query initiator as the return address. A query is allowed to traverse at most TTL number of overlay hops.

When a peer receives a `LocateObject` query, at first it checks the local object repository for the requested object. If available locally, the object is sent to the return address. Otherwise, the query is forwarded to f randomly selected neighbors after the TTL value is decremented by one. Note that, in case the TTL value is decremented to a negative value, instead of forwarding to the neighbors, the query is discarded. When the query initiating peer does not get a response back (due to timeout), it may re-initiate the same query again, possibly with a higher TTL value.

2.4 Properties of the PROOFS Networks

In graph representation, a PROOFS network is depicted with a directed graph. That is, if q is the neighbor of p, p does not have to be the neighbor of q. The biggest challenge of having an undirected graph appears when a peer leaves the network. Each neighbor of the leaving peer has to find a new neighbor who is also willing to accept an additional neighbor. The ease of not having a bi-directional neighborhood relationship is counteracted by network partitioning.

Let $G_d = (V_d, E_d)$ be the proper directed graph representation of a PROOFS network, where each vertex $p \in V_d$ represents a peer and each directed edge from vertex p to vertex q, i.e., $(\overrightarrow{p,q}) \in E_d$ indicates that q is a neighbor of p. Let $G_u = (V_u, E_u)$ be the undirected version of G_d, i.e., V_u and V_d are the same and for each $(\overrightarrow{p,q}) \in E_d$ there exists $(\overrightarrow{p,q}) \in E_u$ and $(\overrightarrow{q,p}) \in E_u$ or simply $(p,q) \in E_u$.

Property 1: Given that, G_u is the undirected representation of a PROOFS network and is connected. If an exchange operation drives the graph representation to G'_u from G_u, G'_u is also connected.[3] In other words, no exchange operation partitions a connected PROOFS network.

Property 2: Let G_d and G_u be the directed and undirected graph representation of a PROOFS network. Let there exists a path from peer p to q in G_u but not in G_d. There exists a series of exchange operations that introduces a path from p to q in G_d. To have the series of operations, consider a path, consisting of the sequence of vertices $< p, (p+1), (p+2), \ldots, (q-2), (q-1), q >$, from p to q in G_u. Considering the same sequence of vertices (and related edges) in the G_d graph, there will be some edges those point towards p and others towards q. Let $(\overrightarrow{u,v})$ be an edge on that sequence of edges pointing towards p. An exchange initiated by u with participant v would make the edge pointing towards q. To have a path from p to q, the direction of all those edges pointing towards p has to be reversed, and any combination of related exchange operations serves the purpose.

2.5 Results

Some of the important results about PROOFS are presented in this section.

[3] For details proof, readers may refer to the original paper.

2.5.1 Connectivity

The authors investigate the effect of dynamic join and leave on the connectivity of the network. In simulations, for each peer p in the PROOFS network, the fraction of other peers in the network reachable from p using the directed path is computed. It was found that at least 95% of the time the average reachability is one, i.e., each peer in the network can reach all other peers. The reported lowest reachability, considering all the peers, is 20%. However, such a poor connectivity occurs in extreme situations such as when the expected time a peer remains in the network is 50 times smaller than the expected time a peer exists the network. In practice, this kind of extreme situation is hardly found.

2.5.2 Noncooperative Peers

The peers in PROOFS are simply applications running on user computers. As a result, it is extremely difficult, if not impossible, to make all the peers fully cooperative. Besides cooperative peers, there may exist peers with different levels of cooperations – (1) a *query-only* peer simply forward queries irrespective of availability of the requested object in the local cache, (2) a *tunneling* peer is same as a query-only peer but considers fanout to be 1, and (3) a *mute* peer drops all the queries from any other peers in the network.

It has been found that if the number of query-only or tunneling peers grows up to 80%, almost 100% queries turn out to be successful in finding the target objects. The worst query success rate is observed with mute peers. When the population of mute peer reaches as high as 80% the query success rate drops but stays above 80%.

3 The CYCLON Network

Spyros et al. propose the CYCLON network [21] as a gossip-based network membership management protocol in unstructured P2P networks. The goal of the research is to design a management protocol that results in a network having low diameter, low clustering, highly symmetric node degree and at the same time is highly resilient to massive node failures.

3.1 An Enhanced Exchange Operation

To achieve the goal, the authors propose an enhanced peer exchange[4] operation. The enhanced operation uses the similar working steps of the basic exchange operation

[4] Spyros et al. use the term enhanced shuffle to designate their peer exchange mechanism.

discussed in the previous section. The critical difference is that unlike the basic one, in enhanced exchange, an initiating peer does not choose the participant randomly.

To facilitate the enhancement, the exchange operation is performed periodically with an interval of Δt. Each peer not only maintains a list of its neighbors but also *age* for each of the logical outgoing links (or edges to neighbors). The age of an edge gives an approximate estimation of time, in Δt unit, since the edge is created by the peer the edge points to. Algorithm 2 shows the steps of the enhanced exchange operation, initiated by peer p with participant q.

Algorithm 2: The enhanced exchange operation of CYCLON

```
1 $p$ increases the age of all outgoing edges pointing to the neighbors
2 Let $q \in \mathcal{N}_p$ be a peer, such that $t_{(\overrightarrow{p,q})} \geqslant t_{(\overrightarrow{p,r})}$, where $r \in \mathcal{N}_p \wedge q \neq r$
3 Let $\mathcal{E}_p \subseteq \mathcal{N}_p$, where each element of $\mathcal{E}_p$ is selected randomly and $|\mathcal{E}_p| = l - 1$
4 $p$ sends a request to $q$ with the set $\mathcal{E}_p \cup \{p\}$ to be a participant
5 if $q$ agrees to the request then
6     $q$ finds $\mathcal{E}_q \subseteq \mathcal{N}_q$, where each element of $\mathcal{E}_q$ is selected randomly and $|\mathcal{E}_q| = l$
7     $q$ sends a respond back to $p$ with the set $\mathcal{E}_q$
8     $p$ updates $\mathcal{N}_p \leftarrow (\mathcal{N}_p \backslash \mathcal{E}_p) \cup \mathcal{E}_q$
9     $q$ updates $\mathcal{N}_q \leftarrow (\mathcal{N}_q \backslash \mathcal{E}_q) \cup \mathcal{E}_p$
```

In the algorithm, p picks up the peer that is pointed by the oldest edge (line 2, where $t_{(\overrightarrow{p,q})}$ is the age of the edge $(\overrightarrow{p,q})$). The number of peers exchanged is called *exchange length* and determined by the system parameter $C \geqslant l > 0$. When q updates its neighbor list, a new edge pointing towards p is to be created with an age of 0. All other edges, including those, which are sent over during the exchange, continue to maintain their respective previous ages. So, while exchanging peers, not only a list of peers are sent out but also their respective ages. As the basic steps of both basic and enhanced exchange operation are the same, the properties described in Section 2.4 also hold for enhanced exchange operation.

3.2 Results

Spyros et al. reports some very interesting property of the CYCLON network. In this section, we discuss some of their findings.

3.2.1 Average Path Length

To compute the average path length, the undirected version of the graph representation of a network is considered. The undirected graph conveys the idea of peers being *informed* of the neighbors in the undirected sense, i.e., if q is a neighbor of p, at some point in the future p will become a neighbor of q. The concept is brought forward due to the second property of exchange networks described in Section 2.4. It is observed that CYCLON network can converge to the average path length of a random network within a hundred cycles, where a cycle is defined by the maximum time duration allowing all peers to engage in a single exchange operation or Δt. It is also observed that the average path length increases logarithmically with the number of peers in the network. These observations indicate robustness of CYCLON in applications where the entire network has to be reached out.

3.2.2 Average Clustering Coefficient

The clustering coefficient is defined as the ratio of the number of existing edges between neighbors of a peer and the total number of possible edges between them. An average over this coefficient for all peers gives an idea of how many peers are neighbors of their own neighbors. The authors demonstrate that average clustering coefficient of a CYCLON network converges to that's of a random network[5] within a few hundred cycles.

3.2.3 Degree Distribution

Degree of a peer is a very important performance metric in unstructured networks. *Degree* of a peer is defined as the number of edges from the peer in the graph representation. The degree related to the number of outgoing edges is defined as out-degree and the number of in coming edges is in-degree. The out-degree of each peer in the CYCLON network is fixed and is always C. In-degree is the factor we are most interested in.

The distribution of in-degree reveals some very significant characteristics of the network as described below:

- Existence of a peers with significantly low in-degree many result in partitioned or disconnected network, in case the peers referring to the concerned peer die or leave the network. Similarly, a peer with a very large in-degree also represents a weak point, as failure of the peer may result in a disconnected network.
- The distribution of in-degree represents how search or other epidemic protocols behave. For example, a massively connected peer may receive same query from many other peers several times which not only waste resources but also give poor performance for the protocols.
- The distribution of in-degree also bears the indication of how loads are distributed among peers. For example, a massively connected peer has to provide responses from many other peers where as a weakly connected peer may simply be idle.

Figure 2 shows an example distribution of in-degree in a basic exchange (such as PROOFS) and enhanced exchange (such as CYCLON) network. As can be seen

[5] Average clustering coefficient of a random network is effectively defined as $\frac{2\times C}{N-1}$, where N is the total number of peers in the network.

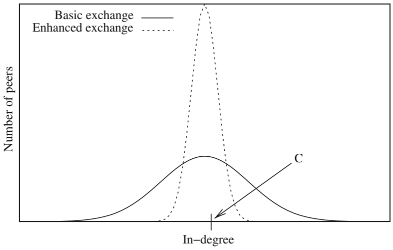


Fig. 2 In-degree distribution

in the figure, most of peers have in-degrees which are very close to the out-degree (i.e., C). This illustrates that the most of the peers have similar load. It also indicates that the load per peer is fair. The amount of services, a peer is expected to deliver is equivalent to the amount of service a peer is expected to receive. Figure 2 can be explained by investigating the introduction, deletion and lifetime of an edge. As shown in Fig. 1, at the end of the exchange, the edge from p to q is deleted and a new edge from q to p is created. The enhanced exchange mandates that the new edge is assigned an age 0 and the deleted edge is the oldest edge towards a neighbor.

At each cycle, a node engages in one exchange operation, and thus creates a new edge and abolishes an old one. Say, at cycle t, a new edge, with age 0, is created towards peer p. At cycle $(t+1)$, another new edge towards p will be created and the age of the last created edge will be incremented to 1. As a peer maintains C number of neighbors and at each cycle the ages of the edges towards neighbors are incremented by 1, a peer can host edges aged from 0 to $(C-1)$ only. So, a newly created edge will be deleted within C cycles and a peer can have an edge as old as C cycles pointing towards itself. In other words, a peer can have C edges pointing towards itself at a single point of time.

In CYCLON, exchange operations between all the peers are periodic but are not synchronized. Thus, the number of edges pointing towards each peer may vary a little around C. Departure of peers and non-responding peers may allow an edge to stay little longer or delete an edge prematurely. However, the system can recover within C cycles. Unlike the enhanced version, basic exchange does not impose any condition on the length of lifetime of an edge. As a result, at one extreme, an edge may stay in the system for indefinite time and at the other extreme, an edge may be deleted in the next cycle of its creation. That's why, the basic exchange operation results in a in-degree distribution with much higher variance as compared with that's of the enhanced version.

4 The IPPS Network

Inexpensive Peer-to-Peer Subsystem (IPPS) is an unstructured platform, proposed for wireless mobile peer-to-peer networks by Akon et al. [1, 2]. The platform addresses the constraints of expensive bandwidth of wireless medium, and limited memory and computing power of mobile devices. It uses a computationally-, memory requirement- and communication- wise inexpensive protocol as the main maintenance operation, and exploits location information of the wireless nodes to minimize the number of link-level messages for communication between peers.

4.1 The Problem and the Goal

A wireless mobile network is a cooperative network where each node requires to collaborate with each other to forward packets from a source to a destination. In such a network, the entire available channel capacity may not be available to an wireless application, and the actual throughput is also determined by the forwarding load generated by other wireless nodes. Besides, mobile devices are battery operated. Unlike electronics, advances in battery technology still lag behind. Minimizing the number of link-level wireless hops helps in increasing the capacity available to the applications. Reduced number of link-level hops also means less number of transmission and less power consumption for a mobile node. Along with being thrifty about bandwidth consumption, a suitable application for mobile devices is required be computationally inexpensive to ensure prolonged battery life and memory requirement-wise economical to confirm accommodation in the small system memory.

In spite of the limitations of wireless mobile networks, P2P over high capacity cellular networks and wireless LANs can provide a wide range of services such as sharing files [9]. In scenarios where accessing a commercial network is expensive, members of a P2P network can share downloaded objects with each other or even can collaborate to download a large popular object. This not only provides a cheaper way of sharing resources, but also enables low latency access to remote objects. Dissemination of rescue or strategic information in a disaster or war zone can be accomplished using mobile wireless P2P network. Short message broadcast, multimedia broadcast, text, audio and / or video based conference are some other examples.

There are some proposals to use existing or modified structured networks in wireless and mobile networks. For example, XScribe [11] is modified from SCRIBE [3] to suite in mobile networks. However, a structured P2P network faces a high network maintenance cost and the ability of this type networks to work in extremely unreliable environments has not yet been investigated. In contrary, an unstructured P2P network is a low cost network which can sustain any extreme environment [19]. Although such a benefit is achieved at the expense of higher search cost, the network assumptions and the overall gain have made this kind of P2P networks so attractive that several unstructured P2P networks have been deployed and are being used by a huge user communities.

In a wired network, due to the abundance of resources, performance metrics of many applications are abstract. However, P2P networks in wireless mobile environment should be very economic about the resources of the wireless medium and devices. The goal of IPPS is to provide an inexpensive and well performing P2P platform on which different P2P applications can be developed. To achieve the goal, an unstructured P2P network, exploiting location information, is examined. While designing the platform, careful choices are made to make the platform flexible, robust and fault tolerant.

4.2 System Model

IPPS system model consists of a set of collaborative computing nodes, each equipped with a wireless interface. Those nodes are assumed to have the capability to form a network on-the-fly using an ad-hoc networking technology, such as GeRaf [23, 24], an efficient location aware transmission (MAC) and forwarding (routing) scheme, to manage the network. In this model, for each node, participation in the P2P network is optional. However, irrespective of its membership in the P2P network, each node participates in routing messages from one node to another as a low level service. The network is equipped with low level (lower than application level) point-to-point unicast primitives, and each of the mobile devices has access to some form of location service [5, 12]. Through this location service, a node in the network can obtain the physical location of itself or other nodes. The information from the location service is used by the lower level network management (i.e., GeRaf) as well as by the P2P modules (i.e., IPPS library). Figure 3 shows an example of the considered network.

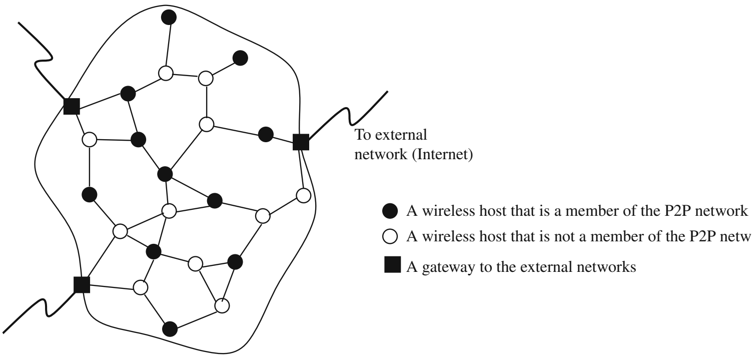

To external network (Internet)
● A wireless host that is a member of the P2P network
○ A wireless host that is not a member of the P2P netw
■ A gateway to the external networks

Fig. 3 An IPPS network

4.3 Topology Maintenance

In this section, we discuss some of the important components and properties of an IPPS network.

4.3.1 The Exchange Operation

IPPS borrows the concept of exchanging neighbors[6] from *PROOFS* [19]. However, the goals of the operation in these two networks are exclusive. In PROOFS network, the operation provides randomness, where as, in IPPS, the operation makes attempts to being neighboring peers closer to each other. The authors make the following claim about their operation.

Claim: It is expected that exchange operations reduce link level hop count between neighboring peers.

Similar to CYCLON, each peer in IPPS performs exchanges at a regular interval. During a exchange, l neighbors are interchanged between the initiator and the participant. Peer p chooses the participating peer q among its own neighbors with the intention of reducing the total distance between the peers. Distance between two peers convey the idea of physical distance between them. Considering the system model and the underlying network infrastructure, the hop count between two neighboring peers is proportional to the distance between them.

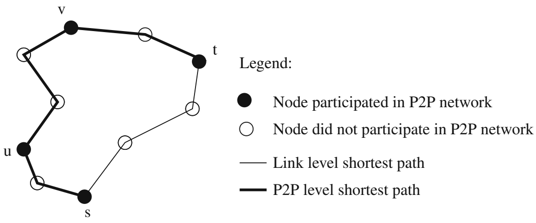

v
t
u
s
Legend:
● Node participated in P2P network
○ Node did not participate in P2P network
Link level shortest path
P2P level shortest path

Fig. 4 Shortest path in P2P and link level network

Claim: Exchange reduces the bandwidth requirement to forward P2P messages.

A peer usually forwards P2P messages, such as query messages, to its P2P neighbors only. As not all communication nodes participate in the P2P network, a P2P level hop may consist of several link level hops. Figure 4 shows the idea pictorially. There exists one non-P2P node between s and u (i.e., two hops), whereas there are two non-P2P nodes between u and v (i.e., three hops). In a random P2P network, on an average one P2P hop consists of average link level path length of the network. In the worst case, where two neighboring peers are located at the extreme ends, a single P2P hop has a link level hop count which is equivalent to the network diameter. Having a neighbor located at a nearby location results in reduction in number

[6] The authors designate their version of neighbor exchange protocol as *reformation*.

of hops between the peers. This helps in reducing of number of link level messages which helps in reducing the total bandwidth consumption to forward P2P messages. Moreover, fewer hops mean reduced message latency. Note that both of these properties are very much desirable for wireless mobile applications, as reduced number of link level messages slows down energy consumption and boosts battery life of mobile devices.

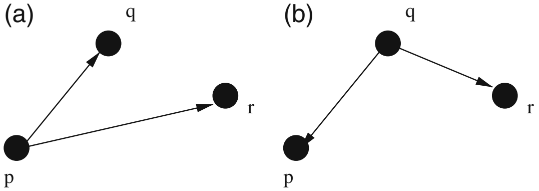


Fig. 5 An exchange operation in IPPS

To have peers located at a close geographic area, the concept of *distance gain* is introduced in IPPS. During an exchange between peers p and q, if the initiating peer p forwards another P2P neighbor r to q, the distance gain is the reduction of the distances between the pairs p and r and the second pair q and r. Figure 5 shows a exchange where a directed edge from any peer x to another peer y means that y is a neighbor of x. Now, the distance gain is formally given by:

$$d^p_{q,r} = |dist(p,r)| - |dist(q,r)| \tag{1}$$

where $dist(x,y)$ is the distance between x and y. When a peer p wants to engage in an exchange, it finds the peer which results in the maximum distance gain. To compute such a metric, for each $q \in \mathscr{N}_p$, p performs the following computations.

1. It computes a *preliminary reform-set* $\mathscr{N}\mathscr{R}^q_p$ such that $|\mathscr{N}\mathscr{R}^q_p| = l - 1$ and $\mathscr{N}\mathscr{R}^q_p \subset \mathscr{N}_p - \{q\}$. The preliminary reform-set must satisfy the following condition:

$$d^p_{q,u} \geqslant d^p_{q,v} \tag{2}$$

   where $u \in \mathscr{N}\mathscr{R}^q_p$ and $v \in \mathscr{N}_p - \mathscr{N}\mathscr{R}^q_p - \{q\}$. In other words, $\mathscr{N}\mathscr{R}^q_p$ includes $l - 1$ number of the most distance gain contributing neighbors of p, during a potential exchange with q;
2. it then computes the net gain for the preliminary reform-set as:

$$d^p_q = \sum_{r \in \mathscr{N}\mathscr{R}^q_p} d^p_{q,r} \tag{3}$$

Finally, p chooses $t \in \mathcal{N}_p$ as the participator of the operation where $d_t^p = \max_{q \in \mathcal{N}_p}\{d_q^p\}$. During the operation, p sends over a *REFORM_REQUEST* message to t accompanied with the reform-set $\mathcal{NR}_p^t \cup \{p\}$. When peer t receives the exchange request from p, it computes the reform-set for p and then sends the set back to p as a *REFORM_RESPONSE* message. Unlike the reform-set from p, the set, computed by t, consists of a list of l peers from $\mathcal{N}_t$ which maximizes the net distance gain for p. After a successful exchange operation, both p and t perform a merge operation as discussed in next. Detailed control flows of an initiator and a participator are given in Algorithm 3 and 4.

4.3.2 The Merge Operation

Peer p performs a merge operation after it gets back the reform-set from t. In contrary, t performs the operation after it decides about the reform-set to send out. Without lose of generality, let p be a peer performing a merge operation. $\mathcal{N}_{send}$ and $\mathcal{N}_{recv}$ be the reform-sets that are sent and received, respectively. During the merge operation peer p updates its neighbor set $\mathcal{N}_p'$ as follows:

$$\mathcal{N}_p' \leftarrow (\mathcal{N}_p \backslash \mathcal{N}_{send}) \cup \mathcal{N}_{recv} \tag{4}$$

where $\mathcal{N}_p'$ is the new P2P neighbor set of p. Note that it is certainly possible that $(\mathcal{N}_p \backslash \mathcal{N}_{send}) \cap \mathcal{N}_{recv} \neq \emptyset$. In such cases, $|\mathcal{N}_p'| < |\mathcal{N}_p|$. Measures should be taken to carefully handle such cases. This issue is further elaborated next.

4.3.3 Number of P2P Neighbors

In IPPS platform, an upper and a lower bounds is set on the size of the P2P neighbor set, a peer can have. Those bounds are defined as N_{max} and N_{min}, respectively and must satisfy the following condition.

$$N_{max} \geqslant N_{min} > l \tag{5}$$

There are some situations when the neighbor set size grows beyond the N_{max} threshold (for example, when a joining peer gathers peers from several known peers for its initial neighbor set). In those cases, the peer will keep N_{max} number of the nearest peers and discard the rest. Similarly, there are some scenarios where a neighbor list shrinks below the N_{min} threshold (for example, when a neighboring peer fails to respond to a P2P control message). Therefore, the peer requests for a neighbor list either from one of the available neighbors or from some widely known repository, following the same procedure of a joining peer.

The upper bound N_{max} puts a limit on the worst case computational and space complexity for a peer. The lower bound N_{min} provides robustness to IPPS. By tuning those parameters, the connectivity of the network can be controlled. The gap between N_{max} and N_{min}, i.e., $(N_{max} - N_{min})$, allows the platform different levels of fault tolerance. The larger the gap, the more a peer tolerates reduction of the size of the neighbor set, i.e., failure of neighbors. PROOFS can be mapped into a special scenario where N_{max} and N_{min} are equal. However, this makes PROOFS unfavorable for wireless mobile networks which suffer from temporal disconnections or for P2P networks which allow dynamic join and leave of participating peers. The reason is that to maintain a specific number of neighbors, PROOFS suffers from a huge number of initialization operation at detecting of each unavailable neighbor.

Algorithm 3: Control flow of an exchange initiating peer *p*

```
while true do
    Compute the participating peer
    Let, $t$ be the participating peer
    Let, $\mathcal{N}_{send}$ be the reform-set
    Send a REFORM_REQUEST to $t$ with the reform-set $\mathcal{N}_{send}$
    if $t$ responds before timeout then
        Let, $\mathcal{N}_{send}$ be the received reform-set in REFORM_RESPONSE
        $\mathcal{N}_p \leftarrow (\mathcal{N}_p \backslash \mathcal{N}_{send}) \cup \mathcal{N}_{recv}$
        if $|\mathcal{N}_p| < N_{min}$ then
            call AddNeighbor()
        else
            Shrink $\mathcal{N}_p$ to size $\min(|\mathcal{N}_p|, N_{max})$
        end
        break
    else
        $\mathcal{N}_p \leftarrow \mathcal{N}_p - \{t\}$
        if $|\mathcal{N}_p| < N_{min}$ then
            call AddNeighbor()
        end
    end
end
```

Algorithm 4: Control flow of an exchange participating peer *q*

```
Let, $\mathcal{N}_{recv}$ be the reform-set received from $p$
Compute the reform-set $\mathcal{N}_{send}$ to send to $p$
Send back a REFORM_RESPONSE to $p$ with reform-set $\mathcal{N}_{send}$
$\mathcal{N}_q \leftarrow (\mathcal{N}_q - \mathcal{N}_{send}) \cup \mathcal{N}_{recv}$
if $|\mathcal{N}_q| < N_{min}$ then
    call AddNeighbor()
end
```

Procedure `AddNeighbor`

```
Let $r$ be the executing peer
repeat
    Send a SHARE_REQUEST to a known repository
    Let, $\mathcal{N}_{recv}$ be the set received in SHARE_RESPONSE
    $\mathcal{N}_r \leftarrow \mathcal{N}_r \cup \mathcal{N}_{recv}$
until $|\mathcal{N}_r| < N_{min}$
Shrink $\mathcal{N}_r$ to size $\min(|\mathcal{N}_r|, N_{max})$
```

4.4 Results

An event driven simulation tool is developed to evaluate the performance of IPPS. In the simulations, a rectangular area of size 175×175 *square units*, where 5000 mobile nodes are randomly distributed according to a Poisson process, is considered. Different status from the network were collected at a fixed interval. The authors compare IPPS with PROOFS wherever possible.

4.4.1 Computational and Memory Complexity

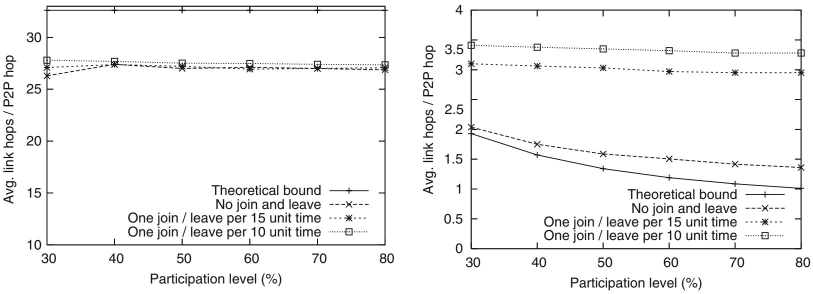
The computational complexity of exchange in IPPS is the complexity faced by the initiating peer. This is due to the fact that the initiating peer incurs more computational complexity than the responding or participating peer. The following is an analysis of the complexity with simple data structures and straight forward algorithms:

1. The complexity to find the net distance gain for a specific neighbor is $\Theta(|\mathcal{N}| + (l-1)) = \Theta(|\mathcal{N}| + l) = \Theta(|\mathcal{N}|)$;
2. For all neighbors, the complexity turns out to be $\Theta(|\mathcal{N}|^2)$;
3. By tracking properly during the previous computations, the neighbor with maximum net gain can be found in $\Theta(1)$ time.

Therefore, the total complexity becomes $\Theta(|\mathcal{N}|^2)$. The worst case scenario arises when $|\mathcal{N}| = N_{max}$ and then the computational complexity becomes $\Theta(N_{max}^2)$. A peer faces the worst case memory requirement when the neighbor list grows beyond N_{max} and this requirement can be formally expressed as $\Theta(N_{max} + l)$. For a given network, N_{max} and l are constants and small positive integers.

4.4.2 Number of Link Level Hops per P2P hop

Figure 6a, b show the average number of link level hops per one P2P hop using the PROOFS and IPPS, respectively. The figures also show the theoretical upper bound on the average number of link level hops, considering that each node has global view of the entire network and no existing node either leaves the P2P network nor a new node join in. In case of PROOFS, due to the randomness of the network, the theoretical upper bound is fairly followed. On the other hand, in case of IPPS, a node does not have the global view and it may not choose the optimal neighbors with the lowest distance between them. As can be seen in Fig. 6b, IPPS performs slightly poorer than the optimal upper bound. As the percentage of mobile nodes participated in the P2P network increases, the number of link level hops per one P2P hop decreases. In fact, as the participation level increases, the chance to find a P2P neighbor at a nearer location also increases. However, if a network uses the PROOFS system (which is random in nature), this metric remains approximately the same, irrespective of different levels of participation. In this case, as the neighbors of a peer are uniformly distributed all over the network, the average link level hop count is not affected at all by the participation level. Actually, the simulation results presented in [21] show that only in an ideal situation (which is a perfect random system with no network dynamics), PROOFS or similar systems can achieve the best performance where the average length of a single P2P hop is equivalent to the average path length of the whole network. Comparing Fig. 6a, b, link level hops per P2P level hop is significantly lower in our proposed platform. This indicates that IPPS reduces the bandwidth requirement and energy consumption to transmit P2P messages.



(a) Basic exchange (PROOFS)

(b) Exchange in IPPS

Fig. 6 Number of link level hops per P2P hop

4.4.3 Dual Cognizance

The exchange operation in IPPS establishes neighborhood relation among geographically close peers. At each exchange, a peer modifies the neighbor set with the peers that are closer than those of the previous set and the neighbors of that peer do the same. So, if p finds q to be at a closer location, it is likely that q also finds *p* the same and includes each other in their neighbor set. The property of a peer being the neighbor of its own peer is defined as *dual cognizance*, by the authors. Figure 7 shows the percentage of peers satisfying the dual cognizance property in PROOFS and IPPS. Note that, in a perfect random PROOFS network, the best case dual cognizance can be analytically defined as $(\frac{N_{\max}}{N-1})^2$, where N is the total number of peers. On the other hand, for an optimal IPPS (where each peer has a global view of the entire network), this metric is 1.

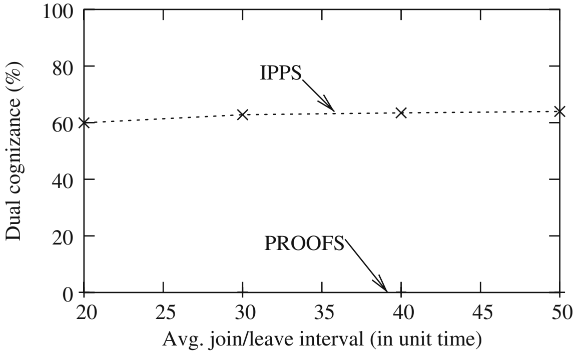


Fig. 7 Effects of join/leave interval on dual cognizance

4.4.4 Minimum Connectivity

An important property of an exchange network is – given a connected network, no exchange operation can make the network disconnected (see Section 2). However, it is possible that the P2P network becomes disconnected as peers join and leave the P2P network. Mobility may further deteriorate the scenario, when the underlying network becomes physically disconnected as mobile nodes are unreachable from one another using radio links. During the simulation, authors compute the connectivity of the P2P network. If p is a neighbor of q, q is considered to *know* p and vice versa, and are connected in both way. The simulation results fairly support the previous claims [19] that for almost all the cases more than 95% of the peers remain connected, given that they are also connected in their radio network. The worst case scenario, i.e., the minimum connectivity in the P2P network, was also investigated. Figure 8 shows the minimum connectivity of the network for different join/leave intervals. The numbers of peers in the largest connected peer graphs are computed and presented after normalizing in 1. As expected, the minimum connectivity decreases with decrement of participation level as well as with the frequency of joining/leaving the P2P network. It can be seen that the worst case connectivity is higher than 70% which provides an indication of robustness of IPPS platform.

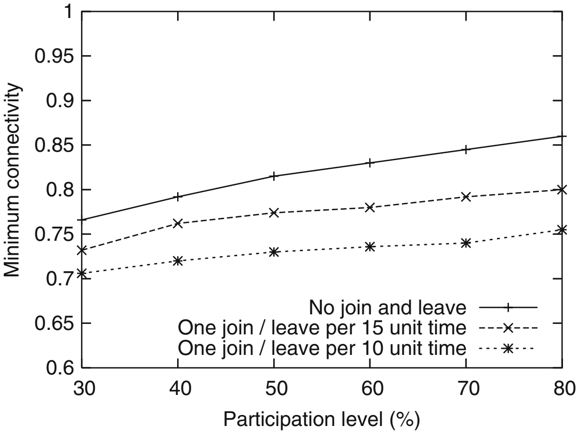


Fig. 8 Minimum connectivity among peers for different join / leave intervals

5 The Gradient Topology Network

In gradient topology, the highest priority entities are connected with each other. These connected entities are called the *core*. Lower priority entities are arranged gradually further away from the core. The position of an entity indicates its priority in the system. In this section, we discuss a gradient topology network, proposed by Jan et al., to facilitate ease of finding resourceful peers in a P2P network [17].

5.1 The Preliminaries

It is observed that distribution of peers in terms of resources is highly skewed [18] and peers with poor resource conditions can result in inferior network performance [14]. These observations lead to the concept of *super peers*. Compared to average peers in the network, a super peer is highly resourceful. To improve performance of the system, in many applications, critical and important services are assigned to these high capacity super peers.

OceanStore [7] architecture exploits a primary tier of super peers with high capacity (in terms of high bandwidth and connectivity) to preserve replicas of objects and employs them to manage updates. In Chord [20], multiple virtual servers are assigned to high performance hosts, i.e., super peers. Such peers are utilized to enhance the routing performance in distributed hash tables (DHT) [10].

The proposal of gradient topology network addresses two issues. Firstly, election of super peers – due to enormous size of P2P networks and their dynamics, it is very difficult for a single entity to maintain a global view of the entire network. So, a distributed solution is desirable. Secondly, finding super peers – it is important that super peers of interest are searchable so that other ordinary peers can easily obtain important services from these super peers. The former problem is out of the scope of this chapter and we concentrate on the later problem, which is solved using an exchange network.

5.2 Exchange Operation

In the gradient topology network, each peer maintains two sets of neighbors. At peer p, the first set, $\mathscr{S}_p$, contains a set of peers with similar capacity (or priority) and like PROOFS, the second set, $\mathscr{N}_p$, maintains a set of random peers. For each neighbor q in both $\mathscr{S}_p$ and $\mathscr{N}_p$, peer p also maintains the capacity (U). The random neighbor set is used to discover unexplored peers in the network for similar capacity. This way, the chance of having more than one cluster of similar capacity peers is reduced. The random network also provides robustness and makes the network resilient to network partitioning. Besides, the random neighbors facilitate the distributed computation of capacity of peers and election of super peers. In this network, peer p performs periodic exchange operation, as shown in Algorithm 6.

Algorithm 6: The exchange operation of Gradient Topology Network

```
Let $q$ be a randomly selected peer from $\mathscr{S}_p \cup \mathscr{N}_p$
$p$ sends a request to $q$ with the two sets $\mathscr{S}_p$ and $\mathscr{N}_p$ to be a participant
if $q$ agrees to the request then
    $q$ sends a respond back to $p$ with the two sets $\mathscr{S}_q$ and $\mathscr{N}_q$
    call GTNReplacePeer ($p$, $\mathscr{S}_q$, $\mathscr{N}_q$) from $p$
    call GTNReplacePeer ($q$, $\mathscr{S}_p$, $\mathscr{N}_p$) from $q$
```

Procedure GTNReplacePeer

```
  input: $x$, $\mathscr{S}_{recv}$, $\mathscr{N}_{recv}$
1 Let $p \in \mathscr{S}_{recv}$ such that $|U(p) - U(x)|$ is the minimum
2 Choose $r \in \mathscr{S}_x$ randomly
3 Update $\mathscr{S}_x \leftarrow (\mathscr{S}_x \cup \{p\}) \setminus \{r\}$
4 Choose $p \in \mathscr{N}_{recv}$ randomly
5 Choose $r \in \mathscr{N}_x$ randomly
6 Update $\mathscr{N}_x \leftarrow (\mathscr{N}_x \cup \{p\}) \setminus \{r\}$
```

In the GTNReplacePeer procedure, the calling peer replaces one entry in the similarity-based set with another peer, received during the exchange operation. The new peer is chosen such that the capacity is similar to the calling peer (lines 1–3).

Later on, one entry in the random neighbor set is replaced with another entry from received random neighbor set (lines 4–6).

5.3 Search

The organization of peers in a gradient topology enables an efficient heuristic search technique to find the high capacity or super peers in the network. The search algorithm uses the capacity information embedded in the topology to restrict the procedure within a small number of peers. When a peer initiates a search, a desired capacity threshold is included within the query message. The threshold is determined based on the resources requirement of the target operation. A peer p with capacity lower than the threshold greedily forwards the query to the neighbor q with highest capacity. Formally, $q \in \mathscr{S}_p \cup \mathscr{N}_p$ and $U(q) \geqslant U(r)$, where $r \in \mathscr{S}_p \cup \mathscr{N}_p \wedge r \neq q$. The forwarding process continues until a peer with required capacity is found or time-to-live (TTL) value of the query expires. Note that, due to peer churn, a search may result in looping in a local minima. To prevent such looping, all visited peers are added the query message and a message is never forwarded to a peer that the message has already visited.

5.4 Results

A P2P network, consisting of up to 100000 peers, is simulated to evaluate performance of the proposed scheme. The capacity of the peers are assigned such that only 1% of them are considered to be super peers. The network is put under constant churn. It is observed that the evolved gradient topology have very small diameter and the average hops to find super peers is bounded by the diameter. In a network as large as to include 100000 peers, the diameter is typically in the order of 5 or 6. As a result, it takes significantly fewer steps to find super peers in the gradient topology as compared to other techniques, such as random walk [8].

6 Concluding Remarks

In this chapter, we have investigated four unstructured P2P networks which are created and maintained using the concept of exchanging peers. These networks demonstrate that simple exchange operation can harness a handful extra ordinary features. They also signify the variety of usages of the exchange operation. The PROOFS network is a robust and scalable network to handle Internet flash crowd that traditional technologies fail to manage. The CYCLON network is introduced to enable a P2P network to be load balanced. IPPS is an unstructured platform for wireless mobile P2P networks. The platform addresses the limitations of wireless medium and mobile devices, such as, expensive bandwidth of wireless medium, and limited memory and computing power of mobile devices. IPPS is a computationally-, memory requirement- and communication- wise inexpensive protocol that is excellently suited for the target environment. Unstructured P2P networks are typically considered to sacrifice search performance for inexpensive maintenance operations. In contrast, the gradient topology network utilizes the exchanging peer mechanism to facilitate a superior search technique.

Little research on efficient searching in unstructured P2P network has been done. Working principles of networks like CYCLON and IPPS reveal interesting and fundamental properties which typical unstructured P2P network do not have. As a result, search techniques exploiting these features are yet to be explored.

References

1. Akon, M., Shen, X., Naik, S., Singh, A., Zhang, Q.: An inexpensive unstructured platform for wireless mobile peer-to-peer networks. Peer-to-Peer Networking and Applications **1**(1), 75–90 (2008)
2. Akon, M.M., Naik, S., Singh, A., Shen, X.: A cross-layered peer-to-peer architecture for wireless mobile networks. In: IEEE International Conference on Multimedia & Expo, pp. 813–816. Toronto, Canada (2006)
3. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.: SCRIBE: A large-scale and decentralised application-level multicast infrastructure. IEEE Journal on Selected Areas in Communications (JSAC) **20**(8), 100–110 (2002)
4. Clarke, I., Miller, S., Hong, T., Sandberg, O., Wiley, B.: Protecting free expression online with freenet. IEEE Internet Computing **6**(1), 40–49 (2002)
5. Cleary, D.C., Parke, D.C.: "Finding Yourself" building location services in a peer-to-peer wireless world. In: EUROCON 2005, pp. 60–63 (2005)
6. Druschel, P., Rowstron, A.: PAST: A large-scale, persistent peer-to-peer storage utility. In: HotOS VIII, pp. 75–80. Germany (2001)
7. Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: Oceanstore: An architecture for global-scale persistent storage. ACM SIGPLAN Notices **35**(11), 190–201 (2000)
8. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: ICS '02: Proceedings of the 16th international conference on Supercomputing, pp. 84–95. NY, USA (2002)
9. Mavromoustakis, C.X., Karatza, H.D.: Segmented file sharing with recursive epidemic placement policy for reliability in mobile peer-to-peer devices. In: IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 371–378 (2005)
10. Mizrak, A.T., Cheng, Y., Kumar, V., Savage, S.: Structured superpeers: Leveraging heterogeneity to provide constant-time lookup. In: IEEE Workshop on Internet Applications, pp. 104–111 (2003)
11. Passarella, A., Delmastro, F., Conti, M.: XScribe: a stateless, cross-layer approach to P2P multicast in multi-hop ad hoc networks. In: International Conference on Mobile Computing and Networking, pp. 6–11. Los Angeles, California (2006)
12. Pias, M., Crowcroft, J., Wilbur, S., Harris, T., Bhatti, S.: Lighthouses for scalable distributed location. In: Second International Workshop on Peer-to-Peer Systems, pp. 278–291 (2003)

13. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content addressable network. In: ACM SIGCOMM (2001)
14. Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J.: Handling churn in a dht. In: USENIX Annual Technical Conference, pp. 127–140 (2004)
15. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms, pp. 329–350. Germany (2001)
16. Rowstron, A., Kermarrec, A.M., Castro, M., Druschel, P.: SCRIBE: The design of a large-scale event notification infrastructure. In: Networked Group Communication, pp. 30–43 (2001)
17. Sacha, J., Dowling, J., Cunningham, R., Meier, R.: Using aggregation for adaptive super-peer discovery on the gradient topology. In: IEEE International Workshop on Self-Managed Networks, Systems and Services, pp. 73–86 (2006)
18. Sen, S., Wang, J.: Analyzing peer-to-peer traffic across large networks. In: ACM SIGCOMM Workshop on Internet measurement, pp. 137–150 (2004)
19. Stavrou, A., Rubenstein, D., Sahu, S.: A lightweight, robust P2P system to handle flash crowds. IEEE Journal on Selected Areas in Communications **22**(1), 6–17 (2004)
20. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: ACM SIGCOMM 2001, pp. 149–160. CA, USA (2001)
21. Voulgaris, S., Gavidia, D., Steen, M.: CYCLON: inexpensive membership management for unstructured p2p overlays. Journal of Network and Systems Management **13**(2), 197–217 (2005)
22. Zhao, B.Y., Kubiatowicz, J.D., Joseph, A.D.: Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. UCB/CSD-01-1141, U. C. Berkeley (2001)
23. Zorzi, M., Rao, R.R.: Geographic random forwarding (GeRaf) for ad hoc and sensor networks: Multihop performance. IEEE Transaction on Mobile Computing **2**(4), 337–348 (2003)
24. Zorzi, M., Rao, R.R.: Multihop performance of geographic random multihop performance of geographic random. IEEE Transaction on Mobile Computing **2**(4), 349–365 (2003)

Peer-to-Peer Topology Formation Using Random Walk[*]

Kin-Wah Kwong and Danny H.K. Tsang

Abstract Peer-to-Peer (P2P) systems such as live video streaming and content sharing are usually composed of a huge number of users with heterogeneous capacities. As a result, designing a distributed algorithm to form such a giant-scale topology in a heterogeneous environment is a challenging question because, on the one hand, the algorithm should exploit the heterogeneity of users' capacities to achieve load-balancing and, on the other hand, the overhead of the algorithm should be kept as low as possible. To meet such requirements, we introduce a very simple protocol for building heterogeneous unstructured P2P networks. The basic idea behind our protocol is to exploit a simple, distributed nature of random walk sampling to assist the peers in selecting their suitable neighbors in terms of capacity and connectivity to achieve load-balancing. To gain more insights into our proposed protocol, we also develop a detailed analysis to investigate our protocol under any heterogeneous P2P environment. The analytical results are validated by the simulations. The ultimate goal of this chapter is to stimulate further research to explore the fundamental issues in heterogeneous P2P networks.

1 Introduction

P2P networks have become an essential part of the Internet and many successful P2P applications have been developed and widely used such as live video streaming (e.g.,

Kin-Wah Kwong
Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA, e-mail: kkw@seas.upenn.edu

Danny H.K. Tsang
Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Kowloon, Hong Kong, e-mail: eetsang@ece.ust.hk

[*] This chapter is based on the publications in [22, 23]. This work was supported by the RGC grant 620306.

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_7, © Springer Science+Business Media, LLC 2010

PPLive[1], CoolStreaming[41]) and content sharing (e.g., BitTorrent[2], Gnutella[3] and KaZaA[4]). An important skeleton in the P2P applications is the formation of overlay topology. In general, overlay topologies can be classified into two types, namely unstructured topology and structured topology (a.k.a. DHT topology). Since unstructured topologies are simpler to construct as well as more robust to network dynamics (e.g., ungraceful peer departure) than structured topologies, as a result many of the P2P applications such as PPLive, BitTorrent, KaZaA and Gnutella use unstructured topologies to form their overlay networks. However, building a good unstructured P2P topology is not a trivial task because there are two main issues which are detailed as follows.

Network heterogeneity. P2P networks are very heterogeneous. For instance, users' access bandwidths are actually very diverse, ranged from 56Kbps connections to several Mbps connections. Moreover, there are many other factors determining a P2P system's performance. Users' local resources such as CPU power and available cache memory are also an important consideration for a P2P system design. Therefore, in this work, we define a generic term "node capacity distribution" to represent the heterogeneity of the users. A natural question to ask is how to build a P2P network based on the "capacity" of each user which is a critical step in load-balancing and providing a stable service to the users.

Scalability. P2P applications usually involve several ten thousands of users and are expected to grow for supporting millions of users. In such a large-scale P2P application, using central servers to coordinate a topology formation process suffers from scalability issues (e.g., server overloading, single point of failure). Furthermore, in bandwidth-demanding applications such as P2P live video streaming, the nodes should connect to a suitable neighbor which has a sufficient bandwidth. In this scenario, how to find such suitable neighbors in a large P2P network with low overhead becomes a challenging problem. It is necessary to have a scalable, lightweight, distributed topology formation algorithm for this purpose.

Therefore, our first objective in this chapter is to introduce a protocol for building a heterogeneous, unstructured P2P network and overcoming the difficulties as outlined above. Our protocol exploits the nature of random walk sampling which is completely decentralized and consumes a low overhead.

P2P networks are extremely complicated because of their unique characteristics which are totally different from the traditional server-client architecture. For example, P2P networks may involve millions of users simultaneously where they can continuously join and leave the networks without a predictable pattern. Moreover, the heterogeneity of the P2P network is changing from time to time. Due to the large complexity, it is difficult to evaluate our protocol solely based on large-scale simulations or Internet experiments which are time consuming and costly. Therefore, our second objective in this chapter is to introduce a mathematical model to analyze our proposed topology formation algorithm. The key point in the analysis is to model the heterogeneity of the users. Hence, based on our analytical model, we can easily examine a large-scale P2P topology structure built by our protocol under any heterogeneous environment. The results also help us to further engineer and optimize P2P protocols running atop the topology.

We structure this chapter as follows. Section 2 reviews literature on different topology formation algorithms and some other applications based on random walk sampling. Section 3 presents our protocol. We then provide a comprehensive analysis for the proposed protocol in Section 4. The simulation results are presented in Section 5. Finally, Section 6 concludes this chapter and discusses some possible extensions of this work.

2 Literature Review

Napster's appearance in 1999 created immense interest in the research community. Napster, based on centralized server architecture, suffers from a poor scaling problem because the file search process is carried out in a centralized server to which peers are required to periodically upload their file indexes. This causes a single point of failure problem and server overload. As a result, a distributed P2P architecture was advocated to replace a Napster-like centralized system. Distributed P2P networks can be generally classified into two main categories, namely unstructured networks and structured networks.

In the first category, unstructured P2P networks have been widely used such as Gnutella, KaZaA and BitTorrent. The benefits of using unstructured networks are in providing resiliency against peer's dynamics (e.g., peers may ungracefully leave the network anytime) and achieving a relatively low maintenance overhead for the overlay topology. In Gnutella, the nodes join the P2P network by connecting to m live nodes (a node is live if it is currently connected to the P2P network). m is typically 3$\sim$5. This method is very ad hoc without any mathematical support. KaZaA and Gnutella2 [5] employ a super-peer topology which is a kind of adaptation technique. In this case, powerful users (such as high bandwidth users) form the backbone of a P2P network and most of the query traffic is processed by them. Therefore, low capacity users can be shielded from massive query traffic, making the unstructured systems more scalable. The important question in constructing two-tier P2P networks is how to select "super" nodes. There are many ad hoc approaches such as selecting "super" nodes based on their lifetime, memory and bandwidth.

Recently, some algorithms have been proposed for building unstructured P2P networks. In [32], the authors suggest a protocol to build low-diameter P2P networks by using a bootstrap server to coordinate the connections between peers. In [33], an algorithm is proposed to produce a topology with $O(\log N/\log\log N)$ diameter where N is the number of peers in the network. Their idea is based on a random graph theory to construct the topology with a certain number of edges. As a result, the targeted diameter is achieved. Phenix [39] was recently proposed for building resilient unstructured P2P networks. First, it utilizes the idea of scale-free network to shorten the topology diameter. Second, it has a built-in mechanism to counter intentional attacks. However, all these algorithms assume the P2P network is homogeneous meaning that each node has the same capacity which is not true in reality. Therefore, it is necessary to develop protocols and analytical models for heterogeneous P2P networks.

In [9], the authors proposed a scalable Gnutella-like system, called Gia, by employing different algorithms such as query caching, adaptive overlay formation and traffic flow control. They carried out different simulations to investigate the system's performance. Since they focus on their specially-designed file-sharing system, it is a prior not clear if their solution can be applied for other P2P applications such as live video streaming and BitTorrent-like applications.

Our protocol, inspired by the Metropolis-Hastings algorithm [17, 31], is based on random walk sampling to assist the peers in selecting their suitable neighbors with high capacity per connectivity. Apart from this work, random walk sampling has also been used in different P2P applications such as construction of a random expander graph [24], realizing a distance-based random long-link connection [42] and searching objects in P2P networks (e.g., [15, 21, 29]). A general mathematical reference of random walk sampling may be found in [6, 26].

In the second category, structured P2P networks such as CAN [34] and Chord [37] have also been proposed. These systems emphasize a highly-organized overlay structure, based on Distributed Hash Tables (DHTs), to improve the search performance on a P2P network and achieve a low network diameter. To achieve an efficient search on a structured overlay network, each peer has to maintain a routing table for forwarding queries. However, under a highly dynamic P2P environment, the maintenance overhead for keeping the routing tables consistent is large, and unfortunately, inconsistent routing tables in DHT networks can significantly degrade the search performance and network diameter. Some hybrid approaches, e.g., [14, 25], suggest for building P2P networks in order to exploit DHT features (e.g., efficient search) while keeping the maintenance overhead small under a high network churn.

3 Our Proposed Protocol

In this section, we introduce a topology formation protocol based on random walk sampling. Then we provide a mathematical model to analyze the structure of the P2P network built by our protocol.

3.1 P2P Network Model

Heterogeneity among P2P users is the main concern in forming the topology. Many factors affect user's heterogeneity. First, each user has certain access link bandwidth, ranged from dial-up modem to several Mbps connections. Second, user's local scarce resources such as CPU power and available memory are also a dominant factor on a P2P system's performance. Therefore, to characterize the heterogeneity of the peers, we define the generic term "node capacity", $\eta_i \geqslant 1$, which is considered as a combination of the access link capacity and available scarce resources of node i. We assume that η_i is chosen from a probability density function (p.d.f.) $\rho(\eta)$ which is called "node capacity distribution" in this chapter. This p.d.f. can be either continuous or discrete. Therefore the node capacity distribution characterizes the global heterogeneity of the P2P network. Determining the value of node capacity depends on applications. For example, in a P2P streaming application, a node capacity can be mainly regarded as a node's upload bandwidth because it plays an important role on the performance of such system. For a network coding system like [16], the peers are required to encode and decode information received from their neighbors. This may be a CPU-consuming task. Therefore, in this case a node capacity should be defined as a combination of CPU power and access link bandwidth.

In this work, we assume that the network bottleneck in a P2P system happens only at the access links of the peers. In addition, we do not consider any correlation between the P2P logical links (i.e., shared congested links) in the Internet cloud since the physical routing path, congestion situation and traffic pattern are dynamically changing over time, and all these make the analytical model highly complicated and intractable. Unless a large-scale Internet experiment is being carried out, the actual environment is very difficult to realistically model and simulate due to the lack of publicly available information such as the Internet topology, routing information, link bandwidth and traffic pattern. This simplified but well-accepted network model has been previously employed by other researchers (e.g., [10, 16, 19, 30, 40]).

3.2 Joining Process

To describe our protocol, we use k_i to represent the degree[2] of node i. Generally, every node prefers to connect to a high capacity node in order to achieve a better service from its neighbors. However, it is not enough to purely consider the neighbor's capacity as a connection criterion. At the same time, each node should not connect to the neighbors with a high degree as well. This is because a high degree node would receive a large workload. Therefore, our joining process considers two important metrics, node's capacity and degree, to make the connection decision. Also, we would like to introduce some randomness for the joining process so that each node can connect to other different nodes. Mathematically, we can formulate our joining process as follows. The probability π_i that node i is connected by a new incoming node is proportional to its capacity and inversely proportional to its current degree in a nonlinear manner controlled by $\alpha > 0$ and $\beta > 0$, ı.e.,

$$\pi_i = \frac{\frac{\eta_i^\alpha}{k_i^\beta}}{\sum\limits_{j \in L(t)} \frac{\eta_j^\alpha}{k_j^\beta}}, \qquad i \in L(t) \tag{1}$$

[2] In this chapter, node degree means the number of P2P connections that a node has.

where $L(t)$ denotes the set of all live nodes in the network at time t. By tuning the values of α and β, they are analogy to transform the underlying node capacity distribution and control the growth of node degree respectively. As a result, our topology formation protocol is flexible in terms of load-balancing or achieving other special requirements (e.g., link-level homogeneity [27, 28]) for a spectrum of P2P applications running on top of the topology. Roughly speaking, if α is small and β is large, the node degree would increase slowly. If α is large and β is small, the node degree would increase fast. We discuss how to tune α and β for "matching" peer's workload for different P2P applications in Section 4.4.

It is impractical to use bootstrap servers to maintain global network information to achieve Eq. (1). Therefore, we employ the Metropolis-Hastings algorithm, developed in [17, 31], to achieve Eq. (1) distributively as follows.

We model a P2P network as a connected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ with node set $\mathscr{V} = \{1, \ldots, n\}$ and edge set $\mathscr{E} \subseteq \mathscr{V} \times \mathscr{V}$, with $(i, j) \in \mathscr{E} \Leftrightarrow (j, i) \in \mathscr{E}$. We assign a transition probability p_{ij} to each edge $(i, j) \in \mathscr{E}$ as follows

$$p_{ij} = \frac{1}{k_i + 1} \min \left\{ 1, \left(\frac{\eta_j}{\eta_i} \right)^{\alpha} \left(\frac{k_i}{k_j} \right)^{\beta} \frac{k_i + 1}{k_j + 1} \right\} \tag{2}$$

and create a self-loop at each node i with $p_{ii} = 1 - \sum_{(i,j) \in \mathscr{E}} p_{ij}$ such that the total transition probability is 1 (k_i and k_j do not count the self-loop). The edge's transition probability is used for random walk which is discussed in the following. By using this algorithm, each node i has to broadcast its capacity η_i and current degree k_i values to its neighbors such that the neighbors can use this information to assign a transition probability to their edges.

When a new node joins the network, it contacts m live nodes in the network. These m live nodes can be retrieved from a bootstrap server or a cached node list stored in the new node. Then, the new node issues m different walkers to these m nodes. Each walker is assigned a time-to-live (TTL) value, τ. This TTL value is equivalent to the number of iterations in the Metropolis-Hastings algorithm. The walker is forwarded from the current node to a neighboring node based on the edge transition probability defined by Eq. (2) and the walker's TTL is decremented by one after each forwarding. The new node connects to a node at which the walker stops (i.e., the value of TTL becomes zero). If a walker stops at the node which is already connected by that new node, then the walker moves additionally δ steps. In this chapter, we assume $\delta = 1$. This process repeats until the walker can find a node for the new node to join. However, this situation is very rare to happen if the network is large. In addition, each walker should send a keep-alive message to the new node, e.g., in each forwarding or a certain period of time. If the new node does not receive the keep-alive message for a period of time, it assumes that a walker is lost and issues another new walker to compensate for the lost one.

We can easily verify that the Markov chain defined by Eq. (2) is (1) *reversible*, i.e., $\pi_i p_{ij} = \pi_j p_{ji}$ for all $i, j \in \mathscr{V}$, (2) *aperiodic* and (3) *irreducible*. Therefore, the sampled probability by the random walk converges to a steady-state distribution that is exactly equal to Eq. (1) when $\tau \rightarrow \infty$. According to our previous experience, a small TTL value ($\tau \sim 10$) can obtain a good random mixing for some networks with 50000 nodes [20].

3.3 Rebuilding Process

When a node leaves the network, all of its neighbors i lose a link. To prevent network breakdown and node isolation, those nodes which lose a link rebuild r_i new link(s) to compensate for the lost one. This is called the rebuilding process. Note that, in general, if $r_i \leqslant 1$, it can also be interpreted as the probability of rebuilding a link. For simplicity, we assume $r_i \leqslant 1$ for the rest of the chapter.

When a node i tries to rebuild a link, it issues a walker with a TTL value, τ, to one of its neighbors randomly.[3] Then the walker traverses the network, the same as the node joining process. Finally, a new link is created by connecting node i and the node at which the walker stops. In particular, we introduce the following two rebuilding schemes (note that we use k_i^- to denote the degree of node i just after losing a link).

Probabilistic-rebuilding scheme: The nodes rebuild a link based on a probability r. Mathematically,

$$r_i = \begin{cases} 1 & k_i^- = 2 \\ r & k_i^- \geqslant 3 \end{cases} \tag{3}$$

for every node i. The threshold on $k_i^- = 2$ means that each node has to maintain at least three links to ensure network connectedness.

Adaptive-rebuilding scheme: The nodes should gradually not rebuild links when their degrees are getting large in order to prevent overloading. At the same time, each node should maintain at least m links such that an overlay service performance and reachability are not degraded. Therefore, this rebuilding process allows the nodes to make rebuilding decision adaptive by considering their current degrees. Mathematically,

$$r_i = \frac{m-1}{k_i^-} \tag{4}$$

for each node i.

In this chapter, we focus on analyzing the probabilistic-rebuilding scheme. The analysis of the adaptive-rebuilding scheme can also be done similarly.

4 Protocol Analysis

Network model can be classified into two main categories, namely static random graph and evolving networks. As for static random graph, the number of nodes

[3] The node can also send a walker to a random node in its cache node list.

and links are kept constant. This model was first suggested by Erdős and Rényi in 1959 [12]. This model has been used in analyzing P2P networks. However, due to its static behavior which is totally opposite to dynamic P2P networks, we believe that this model is not suitable for analyzing P2P networks. Another graph model is so-called evolving networks. The generation method, which is quite different from static random graph, is that the new nodes and links are added to the network over time. Thus the number of nodes in the network keeps increasing.

However, neither model can represent the actual properties of P2P networks. Basically, three phases happen in P2P networks – a growing phase, a stabilizing phase and a decaying phase. In a growing phase, the number of nodes keeps increasing (i.e., the node incoming rate is larger than the node leaving rate), which happens during the transition from non-busy hours to busy hours because many users go online. However, the size of the network cannot increase indefinitely. When the network reaches a certain size, the node incoming rate would be roughly the same as the node leaving rate. As a result, the number of nodes remaining in the network would be roughly constant. This is called a stabilizing phase. After the peak hours, the network enters a decaying phase in which the size decreases as many users go offline (i.e., the node incoming rate is smaller than the node leaving rate) and finally reaches another stabilizing phase. These three network evolution phases have been observed in PPLive [1] which is the most popular IPTV system today [18]. The measurement study in [18] shows that the duration of the stabilizing phase is clearly dominant over the other two phases. Similar results are also observed in the Gnutella network [38]. Therefore, it is worth studying and analyzing the P2P networks under the stabilizing phase. We describe our stabilizing network model in the following section.

4.1 Stabilizing Network Model

The model is inspired by the growing network model (e.g., [7, 11, 36]). However, there are two key differences. First, our model assumes the network size to be constant. We believe that our model is suitable to analyze the steady-state behavior of a topology meaning that the P2P network size remains roughly constant over time. This is a common observation in the P2P networks as discussed above. Second, our load-balancing protocol for constructing the topology is orthogonal to the preferential attachment model (e.g., [7]). Our analysis focuses on a large population regime.

Let $k(s,t,\eta_s)$ be the degree of node s at time t and $\eta_s \in [\eta_{\min}, \eta_{\max}]$ be its capacity. Assume the initial network is connected with N nodes. At each time step $t = 1,2,3,\ldots$, a new node is added into the network by connecting to m different live nodes based on our protocol where m is a system design parameter. Each node s is labeled by the time of its arrival $s = 1,2,3,\ldots \leqslant t$. For example, node 1 arrives in the P2P network earlier than node 2 and etc. Without loss of generality, we use step time throughout the analysis. At the same time, a randomly chosen node is removed from the network. It means that every node's lifetime is exponentially distributed as shown in Lemma 1. Then each neighbor s of the departing node should rebuild r_s new link(s) to compensate for the lost link. Thus, the node arrival rate is equal to the node leaving rate, and hence the network size remains N nodes.

We can establish the following differential equation to describe the evolution of a node degree $k(s,t,\eta_s)$ by using the ideas of a continuum approach (e.g., see [7, 11]):

$$\frac{\partial k(s,t,\eta_s)}{\partial t} = m\frac{\frac{\eta_s^\alpha}{k^\beta(s,t,\eta_s)}}{Z} + (r_s - 1)\frac{k(s,t,\eta_s)}{N} + \frac{\frac{\eta_s^\alpha}{k^\beta(s,t,\eta_s)}}{Z}\sum\nolimits_{i\in A(t)} r_i \tag{5}$$

where Z is the normalization factor

$$Z = \sum_{\varsigma\in L(t)} \frac{\eta_\varsigma^\alpha}{k^\beta(\varsigma,t,\eta_\varsigma)}\,. \tag{6}$$

The rationale behind Eq. (5) is explained as follows. At each time step, a new incoming node connects to node s with probability $\pi_s \propto \frac{\eta_s^\alpha}{k^\beta(s,t,\eta_s)}$ (i.e., Eq. (1)), the rate of change in the degree of node s is taken into account by the first term. Additionally, since a randomly chosen node departs, node s loses a link with probability $k(s,t,\eta_s)/N$. After losing a link, node s triggers a rebuilding process to rebuild r_s link(s). This effect is modeled by the second term. Moreover, for each departure event at time t, a set $A(t)$ of nodes lose a link. Since a randomly chosen node is removed from the network, thus $|A(t)| = \langle k(t)\rangle$ where $\langle k(t)\rangle$ is the average degree of the P2P network at time t.[4] Then the rebuilt links may connect to node s and hence this event is taken into account by the third term. Remark that Eq. (5) is a generalized formula for any rebuilding process, not only limited to the probabilistic-rebuilding scheme.

4.2 Analysis of Mean Degree of P2P Network

In this section, we investigate the relationship between the probabilistic-rebuilding scheme and the mean degree of the P2P network. Obviously, we cannot arbitrarily set the value of r because an improper value would generate excess links in the P2P network. Theorem 1 provides a foundation to choose the value of r for the probabilistic-rebuilding scheme and develops a relationship among r, m and the mean degree of the P2P network.

Theorem 1. *In a stabilizing network, the following condition*

$$r \leqslant 1 - \frac{m}{N-1} \tag{7}$$

is required for the probabilistic-rebuilding process. The resulting asymptotic mean degree of the P2P network is

[4] We use the symbol $\langle\cdot\rangle$ to denote expectation.

$$\langle k \rangle = \frac{m}{1-r} \tag{8}$$

Proof. Let $l(t)$ be the number of links in the network at time t. We treat $l(t)$ as a continuous variable. We can establish the differential equation of $l(t)$ for the stabilizing network as follows:

$$\frac{dl(t)}{dt} = m - |A(t)| + \sum_{i \in A(t)} r_i \tag{9}$$

where $A(t)$ is the set of nodes which lose a link in a departure event at time t and $|A(t)|$ denotes the number of nodes in set $A(t)$. On the right side of Eq. (9), the first term represents the increasing rate of the number of links in the network due to the new node arrival. The second and third terms denote the change of links in the network because of the node departure and the rebuilding process respectively. Since a random node is removed in each departure event, thus $|A(t)| = \langle k(t) \rangle$, where $\langle k(t) \rangle = \frac{2l(t)}{N}$ is the mean network degree at time t. Also $r_i = r, \forall i \in A(t)$. Therefore, Eq. (9) can be expressed as

$$\frac{dl(t)}{dt} = m - \frac{2l(t)}{N}(1-r) \ . \tag{10}$$

Suppose $0 \leq r < 1$, the steady state of $l(t)$ can be obtained by letting $\frac{dl(t)}{dt} = 0$ and hence

$$l(\infty) := \lim_{t \to \infty} l(t) = \frac{Nm}{2(1-r)} \ . \tag{11}$$

Since the size of the P2P network is fixed, the number of overlay links must be less than or equal to $\frac{1}{2}\left(N^2 - N\right)$ which represents a fully connected network. Therefore,

$$l(\infty) = \frac{Nm}{2(1-r)} \leqslant \frac{1}{2}\left(N^2 - N\right) \tag{12}$$

$$\Rightarrow r \leqslant 1 - \frac{m}{N-1} \ . \tag{13}$$

The asymptotic mean degree of the network is

$$\langle k \rangle = \frac{2l(\infty)}{N} = \frac{m}{1-r} \ . \tag{14}$$

□

Note that from Eq. (7) the upper bound value of r is very close to 1 because $m \ll N$. Moreover, the mean degree of the topology can be flexibly tuned by varying the values of r and m.

4.3 Analysis of Node Degree Evolution

In order to analyze the topology structure and understand how the node degree evolves over time under a heterogeneous environment, we have to solve the differential equation of the node degree, i.e., Eq. (5). We use a mean-field approach to seek the scaling form of the solutions to Eq. (5), i.e.,

$$\kappa(\zeta,\eta) := k(s,t,\eta) \tag{15}$$

where $\zeta = (s-t)/N$. For simplicity, we use η and η_s interchangeably in this subsection. Before proceeding, we need the following lemma.

Lemma 1. *In a stabilizing network, let $\theta(s,t)$ be the probability that node s is still in the network at time t. Then,*

$$\theta(s,t) = e^{\frac{1}{N}(s-t)} . \tag{16}$$

Proof. We can establish the recurrent equation for $\theta(s,t)$ as follows:

$$\theta(s,t+1) = \theta(s,t)\left(1-\frac{1}{N}\right) \tag{17}$$

Assume the incremental step from t to $t+1$ is very small, then $\theta(s,t)$ can be represented by the differential form:

$$\frac{d\theta(s,t)}{dt} = -\frac{\theta(s,t)}{N} . \tag{18}$$

By solving this differential equation with the initial condition $\theta(s,s) = 1$ (because node s connects to the network at $t = s$), the result follows. □

Now, the first step in solving Eq. (5) is to determine Z which is in terms of the random variable η. By applying the mean-field approach, we use the average value of Z over $\rho(\eta)$ to substitute Z. Thus,

$$\langle Z\rangle = \langle Z_0(t)\rangle + \left\langle \sum_{s\in L(t)} \frac{\eta_s^{\alpha}}{k^{\beta}(s,t,\eta_s)} \right\rangle \tag{19}$$

$$\approx \left\langle \int_0^t \theta(s,t)\frac{\eta_s^{\alpha}}{k^{\beta}(s,t,\eta_s)}ds \right\rangle \tag{20}$$

$$= \int_{\eta_{\min}}^{\eta_{\max}} \eta^{\alpha}\rho(\eta)\int_0^t \frac{\theta(s,t)}{k^{\beta}(s,t,\eta)}dsd\eta \tag{21}$$

where $\langle Z_o(t)\rangle$ denotes the normalization constant due to the original N nodes at $t = 0$. The second term on the right side of Eq. (19) describes the effect due to the arrival nodes after $t = 0$. As the stabilizing network evolves and $t \to \infty$, $\langle Z_o(t)\rangle \to 0$ because these original N nodes gradually leave the network, and hence $\langle Z_o(t)\rangle$ can be ignored. From Eqs. (19 to 20), we use integration to approximate the summation and set the lower limit of the integration to be 0 instead of 1 for easier calculation. Moreover, we multiply $\theta(s,t)$ inside the integration of Eq. (20) in order to take the effect of the node departure into account. Then by the change of variable $\zeta = (s-t)/N$, $\langle Z \rangle$ can be further expressed as follows

$$\begin{aligned} \langle Z \rangle \; &\approx \; N \int_{\eta_{\min}}^{\eta_{\max}} \eta^{\alpha} \rho(\eta) \int_{\frac{-t}{N}}^{0} \frac{e^{\zeta}}{\kappa^{\beta}(\zeta,\eta)} d\zeta d\eta && (22) \\ &\overset{t\to\infty}{=} N \int_{\eta_{\min}}^{\eta_{\max}} \eta^{\alpha} \rho(\eta) \int_{-\infty}^{0} \frac{e^{\zeta}}{\kappa^{\beta}(\zeta,\eta)} d\zeta d\eta \;. && (23) \end{aligned}$$

For convenience, we define G as

$$G := \int_{\eta_{\min}}^{\eta_{\max}} \eta^{\alpha} \rho(\eta) \int_{-\infty}^{0} \frac{e^{\zeta}}{\kappa^{\beta}(\zeta,\eta)} d\zeta d\eta \qquad (24)$$

and hence we replace Z by $\langle Z \rangle = NG$ in Eq. (5). To find the value of G, we have to obtain $k(s,t,\eta)$ and then put it back to Eq. (24) which becomes a self-consistency equation and is solvable by means of numerical methods. $k(s,t,\eta)$ is ready to solve in the following.

To find $k(s,t,\eta_s)$ for the probabilistic-rebuilding scheme, we assume $r_s = r_i = r$ and $|A(t)| = \langle k \rangle = \frac{m}{1-r}$ in Eq. (5). Note that in the above assumptions, we neglect the rebuilding threshold in Eq. (3) and set $|A(t)|$ to $\frac{m}{1-r}$. These assumptions greatly reduce the complexity of the solution but do not introduce a significant error as shown in the simulations. Then the rate equation of the node degree can be solved into the following solution

$$k(s,t,\eta_s) = \left\{ \frac{1}{G(1-r)} \left(\frac{m\eta_s^{\alpha}}{1-r} - \Delta e^{\frac{(1+\beta)(1-r)}{N}(s-t)} \right) \right\}^{\frac{1}{1+\beta}} \qquad (25)$$

where $\Delta := \frac{m\eta_s^{\alpha}}{1-r} - G(1-r)m^{(1+\beta)}$ and G is the non-zero positive constant satisfying the self-consistency equation in Eq. (24).

Equipped with Eq. (25), we can analyze how the node degree evolves over time. In particular, the node degree converges to a bounded equilibrium which depends on the node capacity. This result is summarized as follows.

Theorem 2. *Suppose node s always stays in the network, then the degree $k(s,t,\eta_s)$ of that node converges to the value $k^{\cdot}(\eta_s)$ as $t \to \infty$. Mathematically,*

$$k^{\cdot}(\eta_s) = \max \left\{ \left[\frac{m\eta_s^{\alpha}}{G(1-r)^2} \right]^{\frac{1}{1+\beta}}, 3 \right\} . \qquad (26)$$

Proof. The result can be proved by letting $(s-t) \to -\infty$ in Eq. (25). The lower limit is due to the threshold of Eq. (3). □

Remark that this theorem can also be easily proved by showing that the rate equation of the node degree has the attracting equilibrium which is identical to Eq. (26).

From this theorem, we show that the equilibrium value of a node degree changes accordingly as a node capacity changes. Thus our protocol can adapt to the fluctuation of a node capacity and prevent the nodes from overloading. Furthermore, the convergence speed of the node degree depends on the network size. The bigger the network size, the slower the convergence speed. For example, if the network size is large, the probability of a node being connected is small due to the sampling nature of our algorithm. This property can allow the node degrees growing slowly and prevent them from overloading. Note that the equilibrium degree of a node may be smaller than the initial number of connections (i.e., m) if its capacity is too small. This is because the rate of losing neighbors due to node departure is larger than "gaining" neighbors since the node capacity is too small. This is another property of our algorithm to prevent low-capacity nodes from overloading.

By using Eq. (25), we can also analyze the peer's workload and the P2P network diameter which are the focus of the next two sections.

4.4 Simple Analysis of Peer's Workload

The peer's workload mainly depends on two things, namely what kind of overlay application is running on top of the topology, and how many connections (i.e., node degree) does a peer establish. Suppose that a flooding search is deployed. When a node with degree k receives a flooding query from its neighbor, the node may require to duplicate the message and then forward it to $k-1$ neighbors.[5] As a result, if the node receives k different queries from its neighbors, then the total number of messages forwarded is $k(k-1)$ and hence the workload of the node is $O\left(k^2\right)$. This is analogy to a random walk search in which the workload of a peer with degree k is $O(k)$.

In this chapter, we particularly investigate the situation where the peer's workload is in the polynomial form of its node degree. It is also easy to examine other workload functions based on our model. Suppose peer s with degree $k(s,t,\eta_s)$ experiences a workload of $O\left((k(s,t,\eta_s))^{\gamma}\right)$ for some overlay application where $\gamma > 0$. According to our protocol, each node s has a degree $k(s,t,\eta_s) = O\left(\eta_s^{\frac{\alpha}{1+\beta}}\right)$ where we ignore other constant terms for simplicity. Therefore, each node s has a workload of $O\left((k(s,t,\eta_s))^{\gamma}\right) = O\left(\eta_s^{\frac{\gamma\alpha}{1+\beta}}\right)$.

Then our question is how to set the parameters, α and β, based on γ so that every peer prevents overloading. Intuitively, if $\gamma\alpha/(1+\beta) > 1$, it means that thepeer has

[5] If the node has seen the query before, it may drop the query to prevent looping in the network.

to "work" more than its capacity and hence it is overloading. On the other hand, if $\gamma\alpha/(1+\beta) < 1$, it implies that the peer is under-loading because the peer's workload is smaller than its capacity. Therefore, we can set α and β such that

$$\gamma = \frac{1+\beta}{\alpha} \; . \tag{27}$$

As a result, the workload of each peer s is $O(\eta_s)$ meaning that the peer only requires to do what it is "willing" to do. If a peer's workload is in other forms such as polylogarithmic and linearithmic functions, the peer's workload analysis can be done similarly.

As discussed above, flooding search and random walk search can be characterized by using $\gamma = 2$ and $\gamma = 1$ respectively. In general, for a given γ, there exists more than one pair of α and β satisfying Eq. (27). Note that the values of α and β control the rate of degree growth (i.e., Eq. (5)) and affect the level of degree equilibrium (because the value of G also depends on α and β). Thus, how to further select a pair of α and β from the candidates for constructing the topology depends on the requirements of the overlay applications (e.g., required degree equilibrium, the growth rate of node degree and network diameter). We plan to address this issue in a future work.

Therefore, based on the property of an overlay application running on top of our topology, we can adaptively tune α and β in order to control the workload of the peers and prevent them from overloading.

4.5 Analysis of P2P Network Diameter

Average node-to-node distance[6] is an important parameter in P2P networks. This information can be used to predict the network performance for the protocol design. We can use the following formula, derived in [13], to calculate the diameter D of the P2P network.

$$D = \frac{\ln\left(\langle k^2\rangle - \langle k\rangle\right) - 2\langle \ln k\rangle + \ln N - \varepsilon}{\ln\left[\frac{\langle k^2\rangle}{\langle k\rangle} - 1\right]} + \frac{1}{2} \tag{28}$$

where $\varepsilon \simeq 0.5772$ is the Euler's constant. Based on our analytical result, for a given node capacity distribution and network size, we can easily estimate a suitable value of m (i.e., number of initial connections) such that the diameter is bounded by some specified value or system requirement.

[6] In this chapter, we use the terms, "network diameter" and "average node-to-node distance", interchangeably.

Before using Eq. (28) to calculate the network diameter, we need to know $\langle k \rangle$, $\langle k^2 \rangle$ and $\langle \ln k \rangle$. The mean degree, $\langle k \rangle$, is already known. $\langle k^2 \rangle$ and $\langle \ln k \rangle$ can also be calculated easily. For example, $\langle f(k) \rangle$, where $f(\cdot)$ is some function, can be evaluated by using the following lemma.

Lemma 2. *The value of $\langle f(k) \rangle$ under a stabilizing network can be evaluated as follows.*

$$\langle f(k) \rangle := \frac{1}{N} \left\langle \sum_{s \in L(t)} f(k(s,t,\eta_s)) \right\rangle \tag{29}$$

$$\approx \int_{\eta_{\min}}^{\eta_{\max}} \rho(\eta) \int_{-\infty}^{0} e^{\zeta} f(\kappa(\zeta,\eta)) d\zeta d\eta \tag{30}$$

where $\kappa(\zeta,\eta)$ is the form of Eq. (15).

Proof. We use the same technique as finding $\langle Z \rangle$ presented in Section 4.3 for proving the above result. We do not show the proof again for brevity. □

Note that we take the expectation with respect to the random variable η on the summation sign in Eq. (29). In general, by using this lemma, we can easily calculate any degree moment for any given node capacity distribution.

5 Simulation

This section presents the simulation results to validate our analysis. We simulate the P2P networks as follows. Each node i is assigned a non-zero capacity η_i randomly chosen from a p.d.f. $\rho(\eta)$. The nodes join the network at a Poisson arrival rate λ nodes per unit time. The inter-arrival time of the node departure events is exponentially distributed with mean $1/\mu$ ($\mu = 0$ means no node departure). When a node departure event happens, a node is randomly removed from the network to model the node departure. Initially, the network grows from a small number of nodes $N_{\text{init}} = 5$ by using $\lambda = 1$ and $\mu = 0$ until the network size reaches the value of N. Then we set $\lambda = \mu = 1$ such that the network size maintains roughly at N nodes in the stabilizing phase. During this phase, the simulation runs for 120000 node arrivals, unless specified otherwise, to ensure the system reaches the steady state.

In all the simulations, when a new node joins the network, it contacts a bootstrap server to randomly get m different nodes from the network. Then the new node issues one walker to each of these m nodes. The walkers traverse the network, with TTL τ, based on our proposed algorithm. We let $m = 5$, $\tau = 10$. We run each simulation 30 times and report the average results. We have also tried the biased bootstrap server meaning that every new node randomly obtains m nodes from the last 100 incoming nodes only. The simulation results are roughly the same, so we do not repeat them here.

5.1 Node Capacity Distributions

In our simulations, we particularly explore two different node capacity distributions as follows.

Power-law (PL) capacity: This distribution is defined as follows

$$\rho(\eta) = d\eta^{-3} \tag{31}$$

for $1 \leqslant \eta \leqslant 1000$, where d is the normalization constant. The node capacity is a continuous random variable in this case.

Discrete Gia capacity: According to [35], the Internet access bandwidth of Napster's users is very heterogeneous. Measurement information has been used in [9] to model the heterogeneous P2P networks. The capacity distribution used in [9] can be modeled as

$$\begin{aligned}\rho(\eta) = {} & 0.2\delta(\eta-1) + 0.45\delta(\eta-10) + 0.3\delta(\eta-100) \\ & +0.049\delta(\eta-1000) + 0.001\delta(\eta-10000)\end{aligned} \tag{32}$$

where $\delta(\cdot)$ is the delta function.

In the following two sections, the power law capacity distribution (i.e., Eq. (31)) and the discrete Gia capacity distribution (i.e., Eq. (32)) are labeled by "PL" and "Discrete" respectively.

5.2 P2P Network Diameter

In this section, we validate our analysis by calculating the P2P network diameters. It is well-known that the diameter of random graphs scales as $O(\log N)$ [8] where N is the size of the graph. However, this result does not tell us how peer heterogeneity affects the network diameter and how to optimize it under a heterogeneous environment. Therefore, we are interested in how different node capacities influence the network diameter precisely. We can employ Eq. (28) to calculate the diameter of the P2P network.

In Fig. 1, we show that the analytical results for the diameters of the P2P networks, which are constructed by using $\alpha = 1$, $\beta = 1$ and $r = 0.5$ in the probabilistic-rebuilding scheme, match very well with the simulation results. Our analytical framework can be used to predict the network diameter under any given node capacity distribution. In our examples, the network diameter under the discrete Gia capacity distribution is much shorter than the one in the PL capacity distribution. The main reason is that our protocol makes the highest capacity nodes (e.g., $\eta = 10000$) to connect to more neighbors. As a result, these highest capacity nodes "self-organize" into the "hubs" of the network, and hence the diameter can be greatly reduced through those "hubs". However, in the PL capacity, it is extremely rare to produce high capacity nodes (e.g., $\eta > 500$), because most of the nodes have capacity

$\eta < 10$. It is noted that the variance of the measured diameters is very small, so we do not show the confidence interval for simplicity.

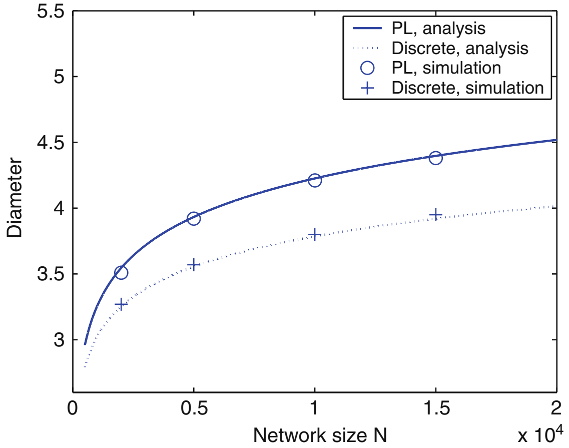


Fig. 1 Comparison of the P2P network diameters between simulations and analytical results. The probabilistic-rebuilding scheme with $r = 0.5$ is used. $\alpha = 1$ and $\beta = 1$

5.3 Node Degree Equilibrium

We have analytically shown that each node degree would eventually converge to a steady-state value, which depends on its capacity. In order to analyze the node evolution, we particularly monitor the 1000-th incoming node (we call it as node A) which is assumed to always stay in the network. In this section, the PL node capacity distribution is used for illustration. The initial capacity η_A of node A is 20. Afterwards, node A changes its capacity to 100, 20 and 50 for each period of 90000 node arrivals. This is used to model the capacity fluctuation of node A. In Figs. 2 and 3, we show the degree of node A under the probabilistic-rebuilding scheme with $r = 0.5$ and the analysis is predicted by Eq. (26) with $(\alpha = 1, \beta = 1)$ and $(\alpha = 1, \beta = 2)$ respectively. In these simulations, we let the value of N to be 15000.[7] From the results, we show that our analysis matches very well with the simulations. In the case of $\alpha = 1$ and $\beta = 1$, the node A's degree is proportional to $\sqrt{\eta_A}$ which implies that the workload of the node is proportional to η_A under aflooding search. We also show that the growth of the node A's degree becomes more

[7] In Figs. 2 and 3, there is a hump when the number of node arrivals equals 15000. This is because, before the network size reaches 15000, the network still behaves as a growing phase and hence the network behavior is totally different from a stabilizing phase when the network size reaches 15000.

"restrictive" as β increases to 2 as in Fig. 3. Therefore, the results match the idea of our load-balancing protocol. Moreover, the equilibrium degree changes accordingly as the node capacity varies. When node A's capacity is higher than other neighboring nodes, the random walkers have a larger chance to be "attracted" to node A. As a result, more links would be connected to it. On the contrary, if node A's capacity is low, the random walkers have a smaller chance to traverse node A, and hence the degree of node A is kept small. Therefore, from our analysis and simulations, we prove that our algorithm can adapt to the fluctuation of node capacity. Moreover, since a node can control its degree by changing the capacity, so the workload of a node can be calibrated accordingly.

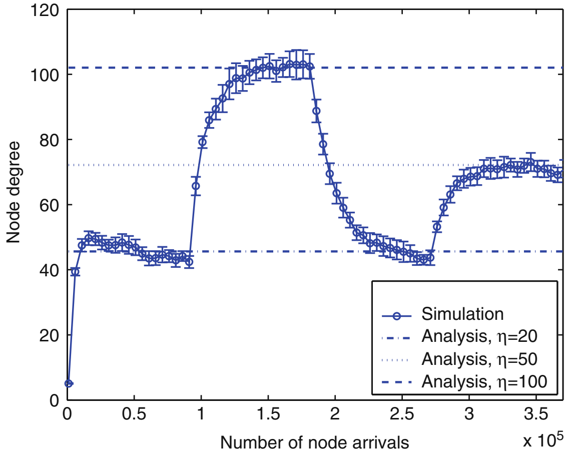


Fig. 2 Node A's degree evolution, $N = 15000$, $r = 0.5$, $\alpha = 1$ and $\beta = 1$. 95% confidence intervals are plotted in the figure as well

6 Conclusion

In this chapter, we first introduce a capacity-aware protocol to build the unstructured P2P networks. Our protocol exploits the idea of random walk sampling which provides a simple, low-overhead and distributed way to form the topology with a load-balancing feature in a heterogeneous environment. Moreover, no specially-designed bootstrap server is required to support our protocol because every new incoming node just needs some live nodes as a starting point of the random walk. Therefore, the workload, complexity and dependence of a bootstrap server can be greatly reduced which is a critical step to make the P2P networks scalable and reliable.

Furthermore, we provide a comprehensive analysis to study the performance of our proposed protocol such as network diameter and degree evolution under any heterogeneous environment. The analytical results are validated by the simulations. Our analytical model is mathematically tractable and easy to be extended to analyze other P2P topology formation algorithms.

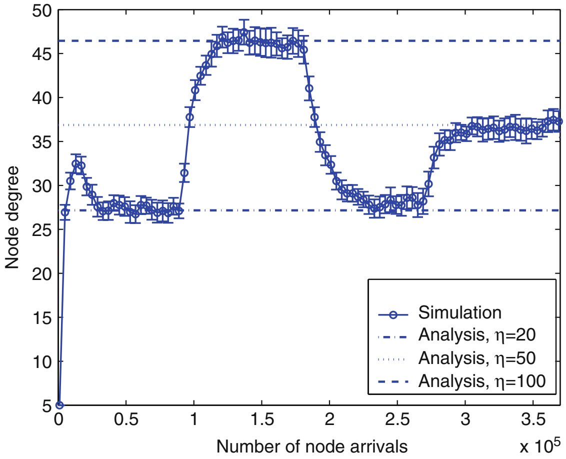


Fig. 3 Node A's degree evolution, $N = 15000$, $r = 0.5$, $\alpha = 1$ and $\beta = 2$. 95% confidence intervals are plotted in the figure as well

Starting from this work, there are still many interesting open problems to tackle in which some of them are listed in the following.

1. Based on our analytical model, it is possible to optimize the topology formation process with respect to a P2P application running on top of the overlay. For example, some simple ideas have been discussed in Section 4.4 for the P2P applications with different workload's behaviors. It would be a nice extension to consider a joint optimization of a P2P application (e.g., video streaming) and the topology formation process.
2. Since random walk sampling can incur delay in the joining and rebuilding processes which can be harmful for delay-sensitive applications such as live video streaming, it is necessary to develop methods to reduce such delay and minimize the impacts on the P2P applications.
3. It is important to investigate the mixing time of our random walk algorithm on a heterogeneous P2P environment which can provide insights on how to select a right time-to-live (TTL) value. Also, it is interesting to characterize the relationship between mixing time, network size and node capacity distribution.
4. Our analytical model assumes a simple peer arrival and departure process. A possible extension is to develop a more sophisticated model in order to capture different peer arrival and departure patterns.

References

1. PPLive, `http://www.pplive.com`
2. BitTorrent, `http://www.bittorrent.com`
3. Gnutella, `http://en.wikipedia.org/wiki/Gnutella`
4. KaZaA, `http://www.kazaa.com`
5. Gnutella2, `http://en.wikipedia.org/wiki/Gnutella2`
6. Aldous, D., Fill, J.: Reversible markov chains and random walks on graphs. `http://stat-www.berkeley.edu/users/aldous/RWG/book.html`
7. Barabási, A.L., Albert, R., Jeong, H.: Mean-field theory for scale-free random networks. Physica A **272**(1), 173–187 (1999)
8. Bollobás, B.: Random Graphs, second edn. Cambridge University Press (2001)
9. Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S.: Making gnutella-like P2P systems scalable. In: Proc. ACM SIGCOMM (2003)
10. Cui, Y., Xue, Y., Nahrstedt, K.: Max-min overlay multicast: Rate allocation and tree construction. In: Proc. IWQoS (2004)
11. Dorogovtsev, S.N., Mendes, J.F.F.: Scaling properties of scale-free evolving networks: Continuous approach. Physical Review E **63**(5), 056,125 (2001). DOI 10.1103/PhysRevE.63.056125
12. Erdős, P., Rényi, A.: On random graphs. Publicationes Mathematicae **6**, 290–297 (1959)
13. Fronczak, A., Fronczak, P., Holyst, J.A.: Average path length in random networks (2002). `http://arxiv.org/abs/cond-mat/0212230`
14. Ganesan, P., Sun, Q., Garcia-Molina, H.: Yappers: A peer-to-peer lookup service over arbitrary topology. In: Proc. IEEE INFOCOM (2003)
15. Gkantsidis, C., Mihail, M., Saberi, A.: Hybrid search schemes for unstructured peer-to-peer networks. In: Proc. IEEE INFOCOM (2005)
16. Gkantsidis, C., Rodriguez, P.R.: Network coding for large scale content distribution. In: Proc. IEEE INFOCOM (2005)
17. Hastings, W.: Monte carlo sampling methods using markov chains and their applications. Biometrika **57**(1), 97–109 (1970)
18. Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K.: A measurement study of a large-scale P2P IPTV system. IEEE Tranactions on Multimedia **9**, 1672–1687 (2007)
19. Kim, M.S., Lam, S.S., Lee, D.: Optimal distribution tree for internet streaming media. In: Proc. International Conference on Distributed Computing Systems (2003)
20. Kwong, K.W., Tsang, D.H.K.: On the relationship of node capacity distribution and P2P topology formation. In: Proc. IEEE Workshop on High Performance Switching and Routing (HPSR) (2005)
21. Kwong, K.W., Tsang, D.H.K.: A congestion-aware search protocol for heterogeneous peer-to-peer networks. Springer Journal of Supercomputing **36**(3), 265–282 (2006)
22. Kwong, K.W., Tsang, D.H.K.: Application-aware topology formation algorithm for peer-to-peer networks. In: Proc. IEEE International Conference on Communications (ICC) (2007)
23. Kwong, K.W., Tsang, D.H.K.: Building heterogeneous peer-to-peer networks: Protocol and analysis. IEEE Transcations on Networking **16**, 281–292 (2008)
24. Law, C., Siu, K.Y.: Distributed construction of random expander networks. In: Proc. IEEE INFOCOM (2003)
25. Loo, B.T., Huebsch, R., Stoica, I., Hellerstein, J.M.: The case for a hybrid p2p search infrastructure. In: Proc. International Workshop on Peer-to-Peer Systems (2004)
26. Lovász, L.: Random walks on graphs: A survey. In: Combinatorics, vol. 2, pp. 1–46. Bolyai Society Mathematical Studies (1993)
27. Luan, H., Kwong, K.W., Huang, Z., Tsang, D.H.K.: P2P live streaming towards best video quality. In: Proc. Special Session on P2P Media Streaming, IEEE Consumer Communications and Networking Conference (2008)
28. Luan, H., Tsang, D.H.K., Kwong, K.W.: Media overlay construction via a markov chain monte carlo method. In: ACM SIGMETRICS Performance Evaluation Review, vol. 34, pp. 9–11 (2006)

29. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: Proc. ACM International Conference on Supercomputing (2002)
30. Meo, M., Milan, F.: A rational model for service rate allocation in peer-to-peer networks. In: Proc. IEEE Global Internet Symposium (2005)
31. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E.: Equation of state calculations by fast computing machines. The Journal of Chemical Physics **21**(6), 1087–1092 (1953)
32. Pandurangan, G., Raghavan, P., Upfal, E.: Building low-diameter peer-to-peer networks. IEEE Journal on Selected Areas in Communications **21**, 995–1002 (2003)
33. Pyun, Y.J., Reeves, D.S.: Constructing a balanced, (log(n)/loglog(n))-diameter super-peer topology for scalable P2P systems. In: Proc. IEEE P2P Computing (2004)
34. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: Proc. ACM SIGCOMM (2001)
35. Saroui, S., Gummadi, P.K., Gribble, S.D.: Measurement study of peer-to-peer file sharing systems. In: Proc. Multimedia Computing and Networking (2002)
36. Sarshar, N., Roychowdhury, V.: Scale-free and stable structures in complex ad hoc networks. In: Physical Review E, vol. 69 (2004)
37. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proc. ACM SIGCOMM (2001)
38. Stutzbach, D., Zhao, S., Rejaie, R.: Characterizing files in the modern gnutella network. Multimedia Systems Journal (2007)
39. Wouhaybi, R.H., Campbell, A.T.: Phenix: Supporting resilient low-diameter peer-to-peer topologies. In: Proc. IEEE INFOCOM (2004)
40. Yang, X., Veciana, G.: Service capacity of peer to peer networks. In: Proc. IEEE INFOCOM (2004)
41. Zhang, X., Liu, J., Li, B., Yum, T.P.: Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming. In: Proc. IEEE INFOCOM (2005)
42. Zhong, M., Shen, K., Seiferas, J.: Non-uniform random membership management in peer-to-peer networks. In: Proc. IEEE INFOCOM (2005)

Semantic Social Overlay Networks

Alexander Löser, Steffen Staab, and Christoph Tempich

Abstract Knowledge sharing in a virtual organization requires a knowledge life cycle including knowledge provisioning, terminology alignment, determination of resource location, query routing, and query answering. We focus on determining a relevant resource in a completely decentralized setting such as necessitated by peer-to-peer knowledge management in virtual organizations. Requirements for this task include, e.g., full autonomy of peers as well as full control over own resources and therefore preclude prominent resource location and query routing schemes such as distributed hash tables. In order to tackle given requirements in this chapter we introduce use a resource location and query routing approach called INGA [23, 24, 31]. It exploits metaphors known from online social networks as well as the semantic similarity between queries and meta data of annotated documents. To adapt to the dynamics of the networks and to bound the local index we present an index update policy combining temporal, semantic and community locality. To further boost performance and enhance recall in a dynamic setting we introduce in INGA recommender and bootstrapping overlays. We have built a network simulator and conducted extensive experiments under realistic conditions. Results show that INGA outperforms other state-of-the-art approaches significantly while it displays small world characteristics.The approach has been fully tested in simulation runs and implemented in the system Bibster (http://bibster.semanticweb.org).

Alexander Löser
Technische Universität Berlin, Einsteinufer 17, 10587 Berlin, Germany,
e-mail: alexander.loeser@tu-berlin.de

Steffen Staab
University of Koblenz-Landau, Universitätsstrasse 1, 56070 Koblenz, Germany,
e-mail: staab@uni-koblenz.de

Christoph Tempich
DETECON international GmbH, Oberkasselerstrasse 2, 53227 Bonn, Germany,
e-mail: christoph@tempich.com

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_8, © Springer Science+Business Media, LLC 2010

1 Semantics and Communities in Peer-to-Peer Networks

Finding relevant information from a heterogeneous set of information resources is a longstanding problem in computing. In everyday life we observe that there are successful strategies for finding relevant information in a social network of people. Milgram's [25] and Kleinbergs [19] experiments illustrated that people with only local knowledge of the network (i.e., their immediate acquaintances) were quite successful at constructing acquaintance chains of short length, leading to "small world" networks. Studies of social networks, such as *Linkedin* or *XING* show that the challenge of finding relevant information may often be reduced to asking the "right" people. e.g., People who either have the desired piece of information and can directly provide the relevant content or the ones who can recommend "the right people". Another popular example are bootstrapping mechanisms of early distributed and unstructured music file sharing networks: To envision how Gnutella originally worked, imagine a large circle of users (called nodes), who each have Gnutella client software. On initial startup, the client software must bootstrap and find at least one other node. Before the concept of *Gnutella Web Caches (GWC)* where introduced one popular method was to ask people from a list of own contacts at the internet rely chat. Preferably those ones where contacted which had music files the would match future queries.

This section transfers observations from real-world social networks to the design of a peer-to-peer overlay structure. We start with identifying routing strategies in social networks and abstracting them to peer-to-peer overlay networks. Later we introduce strategies for knowledge sharing in virtual organizations for two applications in detail.

1.1 Query Routing Strategies in Social Networks

Real-world social networks are *highly dynamic* w.r.t. peer availability and people's expertise on topics. People querying the network still find the right information in that they use their *local knowledge* enriched by means of *semantic similarity* measures in order to determine knowledgeable person. We observe that such mechanisms in social networks work although

- people may not always be available to respond to requests,
- people may shift their interests and attention,
- people may not have exactly the "right" knowledge, but only knowledge which is *semantically close* to the request.

By local knowledge we refer to locally available information in a structured and unstructured form (e.g., meta tags of music files or extracted from documents) and expertise information about the own knowledge and knowledge from other persons. Inspired by these observations and focussed by the requirements of semantic search in the setting of distributed autonomous information sources, we have conceived

INGA [24] a novel peer-to-peer algorithm where each peer plays the role of a person in a social network. In INGA knowledge facts are stored and managed locally on each peer constituting the ‘topical knowledge’ of the peer. A peer responds to a query by providing an answer matching the query or by forwarding the query to what he deems to be the most appropriate peers. For the purpose of determining the most appropriate peers, each peer maintains a *personal semantic shortcut index*. The index is created and maintained in our highly dynamic setting in a lazy manner, i.e., by analyzing the queries that are initiated by users of the peer-to-peer network and that happen to pass through the peer.

The personal semantic shortcut index maintained at each peer reflects that a peer may play the following four different roles for the other peers in the network (in decreasing order of utility):

- The best peers to query are always those that have successfully answered the same or a semantically similar query in the past. We call such peers *content providers*.
- If no content providers are known, peers are queried that have *issued semantically similar queries* in the past. The assumption is that this peer has been successful in getting matching answers and now we can directly learn from him about suitable content providers. We call such peers *recommenders*.
- If we do not know either of the above we query peers that have established a good social network to other persons over a variety of general domains. Such peers form a *bootstrapping network*.
- If we fail to discover any of the above we fall back to the default layer of neighboring peers. To avoid overfitting to peers already known we occasionally select random peers for a query. We call this the *default network*.

Seen from a local perspective, each peer maintains in its index information about some peers, about what roles these peers play for which topic and how useful they were in the past. Seen from a global perspective, each of the four roles results in a network layer of peers that is independent from the other layers.

1.2 Knowledge Sharing Strategies in Virtual Organizations

We introduce two case studies, the IBIT case study in the area of tourism and the SWAP Bibliography case study. Both studies have been investigated, implemented and deployed in a large EU IST project called SWAP Semantic Web And Peer-to-Peer).

The IBIT case study [30] is about sharing of databases and documents between several autonomous, but cooperating tourism organizations on the Baleares, a group of Spanish islands in the Mediterranean. Some of the features of the case study include that the definition of a unique global schema or ontology is not possible, that the topics they are interested in keep changing and that the knowledge management support to be provided by the peer-to-peer knowledge management system must deal with this flexibility.

In the SWAP Bibliography case study, we have explored the sharing of Bibtex information between peers of researchers. Bibtex is locally harvested from files and stored on each peer in the SWAP *local node repository*. Each researcher may search on the own peer as well as in the P2P network in order to retrieve the appropriate bibliographic data. This scenario is particularly interesting for further investigation, because *(i)* Bibtex data have a stable interesting core, but also greatly varying additional fields as each user may define his own bibtex entries; *(ii)* Bibtex data can never be fully captured in a centralized repository, because one repository such as DBLP can only reflect a small set of topics (e.g., databases and AI, but not organizational issues of knowledge management).

2 System Architecture

In this section we start with surveying our application platform. Later we describe peer selection strategies on top of a an unstructured peer-to-peer network implementing social and semantic routing strategies. For evaluation purposes we use the SWAP infrastructure [15]. We recall that it provides all standard peer-to-peer functionality such as information sharing, searching and publishing of resources.

2.1 Building Blocks

Figure 1 shows the basic building blocks of our architecture. We assume that each peer provides a unique peer identifer (PID). Similar to file sharing networks each peer may publish all resources from its *local content database*, so other peers can discover them by their requests (this also applies to resources downloaded from other peers). All information is wrapped as RDF statements and stored in an RDF repository.[1] Additionally to local data (*Nick isExecutiveEditorOf JSAC2005*) each resource is assigned a topic *(JSAC isTypeOf IEEEJournal)* and hierarchical information about the topics is stored *(IEEEJournal subTopicOf Journal)*. The topics a peer stores resources for are subsequently referred to as the peer's own topics. Our algorithm supports exact match queries and queries that use a similarity function. For the last type we investigate in Section 6.2 two types of queries: single predicate queries queries using a common topic hierarchy or conjunctive queries without a shared topic hierarchy. For successful queries (own queries or those of other peers), which returned at least one match, the *shortcut management* extracts information about answering and forwarding peers to create, update or remove shortcuts in the *local shortcut index*. Contrary to related approaches, such as DHTs, INGA peers

[1] `http://www.openrdf.org/`

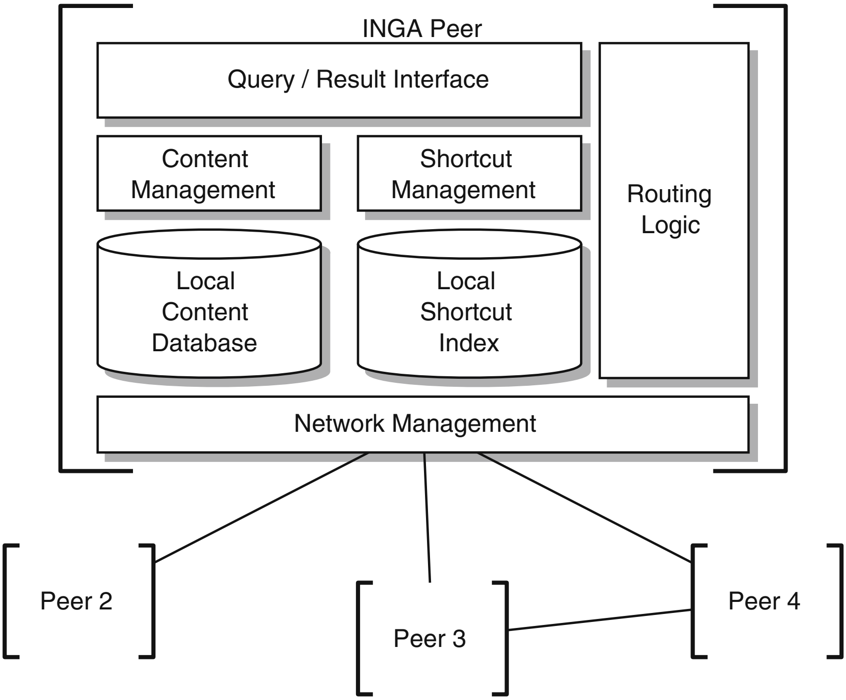


Fig. 1 Architecture model

only index "egoistically", i.e., shortcuts on topics they requested themselves. The *routing logic* selects "most suitable" peers to forward a query to, for all own queries or queries forwarded from remote peers. The selection depends on the knowledge a peer has already acquired for the specific query and the similarity between the query and locally stored shortcuts.

2.2 *Query and Result Messages*

We use a simple query message model which is similar to the structure of a Gnutella query message. Each query message is a quadruple: $QM(q,b,mp,qid)$ where q is a SERQL query.[2]). We support conjunctive SERQL queries, however for routing purposes only the topic information is used. From a query for all *IEEEJournal* with editor-in-chief *Nick*, only *JSAC* is utilized for routing. b is the bootstrapping capability of the querying peer to allow the creation of bootstrapping shortcuts, mp the message path for each query message containing the unique PIDs of all peers, which have already received the query, to avoid duplicated query messages, and qid a unique query ID to ensure that a peer does not respond to a query it has already

[2] SERQL is a SQL like query language for RDF.

answered. Unique query IDs in INGA are computed by using a random number generator that has sufficiently high probability of generating unique numbers. A result message is a tuple: $RM(r, mp, qid)$ where r represents the answer to the query. We just consider results which exactly match the query. Besides the message path mp is copied to the answer message to allow the creation of recommender and content provider shortcuts.

2.3 Similarity Function

INGA directs queries via shortcuts that exactly match a query. If such shortcuts do not exist, INGA selects shortcuts independently from the given query topic, e.g., shortcuts to peers that are well connected in the network. A similarity function helps to identify shortcuts to peers that provide content or have issued questions similar to the given query. Thus, shortcuts are selected based on the given query. Depending on query type, we investigated the following types of shortcut similarity functions:

- **Single predicate query using a common topic hierarchy.** In case the peers in the network share a common topic hierarchy our routing algorithm does not only use exact index hits, but it also exploits the semantic similarity between a query and an shortcut. In this case a query consists of a single predicate, which represents a topic in a common topic hierarchy, We define the similarity function $sim : qt \times sc \rightarrow [0;1]$ between the extracted topic qt from a SERQL query q and the extracted topic st from a shortcut, which are both given by query topics in the same topic hierarchy, as according to [20] as :

$$sim_{Topic}(qt, st) = \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & \text{if } q \neq sc \\ 1 & \text{otherwise} \end{cases} \qquad (1)$$

  where l is the length of the shortest path between qt and st in the graph spanned by the sub topic relation and h is the minimal level in the topic hierarchy of either qt or st. α and β are parameters scaling the contribution of shortest path length l and depth h, respectively. Based on the benchmark data set given in [20], we chose $\alpha = 0.2$ and $\beta = 0.6$ as optimal values.
- **Conjunctive queries.** Each query may include several predicates, e.g., *Select all resources that belong to the topic semantic web and to the topic p2p*. In the Semantic Web context we formalize this query using common topic hierarchies, *Find any resource with the topics /computer/web/semanticweb* $\wedge$ */computer/distributed/p2p* . A default approach would route a query only to a peer that matches *all* predicates of the query using a simple exact match paradigm. Too specific query predicates under the exact match paradigm often lead to empty result sets and do not appropriately consider negation. The notion of best matches and relative importance of predicates can be a good alternative to satisfy a user's information needs independently of the individual peer instances. In [31] we investigated metrics to determine the best peers to route a query

using multi predicate queries in shortcut networks. We observed satisfying results using the selection function described in [10] which uses an equation similar to equation (2) to combine query hits for distributed document retrieval. We refer to this strategy as *Multiply*.

$$R_p(q) = \prod_{i=1}^{\#t} q_i^p \tag{2}$$

We calculate the relevance *R* for a peer *p* for a query *q* using equation (2), where *#t* represents the number of topics in the query, q_i^p represents the query hits per topic *i* of each peer matching at least one of the topics of the query. We select the peers with the highest relevance.

3 Overlay Construction, Index Creation and Maintenance

Each peer is connected to a set of other peers in the network via uni-directional shortcuts. Following the social metaphors in Section 1, we introduce four different types of shortcuts a peer can use to route queries and discuss how these shortcuts are created.

3.1 Content Provider and Recommender Shortcuts

Content Provider Layer. The design of the content provider shortcut overlay departs from existing work as published in [29, 32] and exploits the simple, yet powerful principle of interest-based locality. When a peer joins the system, it may not have any information about the interest of other peers. It first attempts to receive answers for its queries by exploiting lower layers of the INGA peer network, e.g., by flooding. The lookup returns a set of peers that store documents for the topic of the query. These peers are potential candidates to be added to the content provider shortcut list. Each time the querying peer receives an answer from a remote peer, content provider shortcuts *sc* to new remote peers are added to the list in the form: *sc(topic, pid, query hits,"c", update)*, where *topic* is the query topics taken from the query message, *pid* is the unique identifier of the answering peer, *query hits* is the number of returned statements, *"c"* represents a type of shortcut, viz. content provider shortcut and *update* is the time, when the shortcut was created or the last time, when the shortcut was successfully used. Subsequent queries of the local peer or of a remote peer are matched against the topic column of the content provider shortcut list. If a peer cannot find suitable shortcuts in the list, it issues a lookup through lower layers, and repeats the process for adding new shortcuts. For an example consider Fig. 2. Peer 2 discovers shortcuts for the topic */Education/UML* by flooding the default network with a maximum number of three hops (TTL) and creates two content provider shortcuts to peer 3 and peer 5.

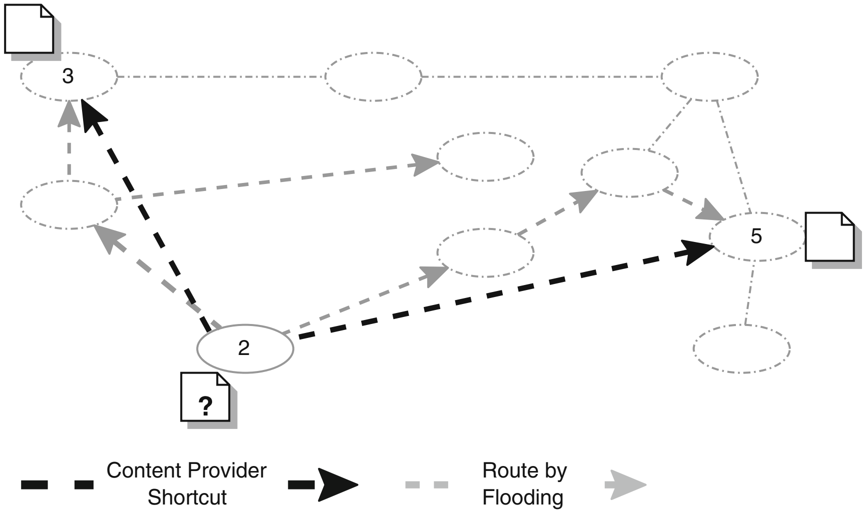


Fig. 2 Content provider shortcut creation

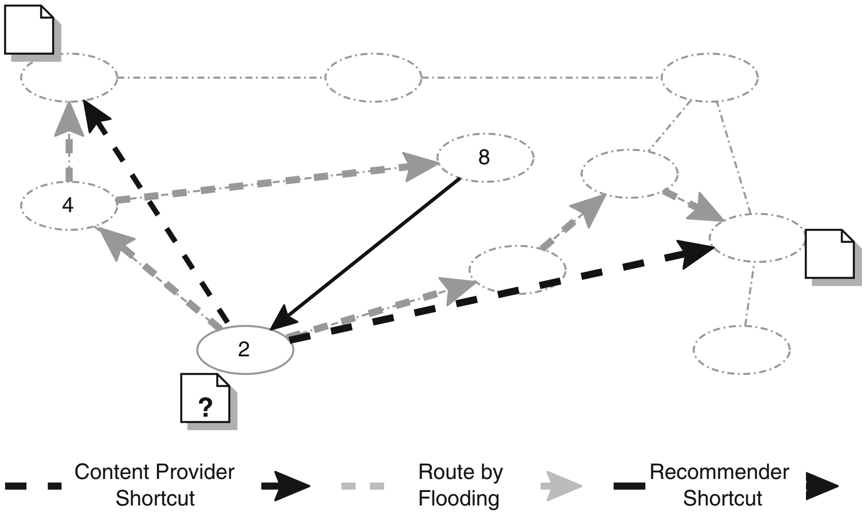


Fig. 3 Recommender shortcut creation

Recommender Layer. To foster the learning process of recommender shortcuts, especially for new peers in the network, we consider the incoming queries that are routed through one's peer, i.e., $\{QM(q,b,mp,qid)\}$. A recommender shortcut *sc(topic,pid,query hits, maxsim, "r", update)* is created, where *topic* is the set of query topics from the query *q*. The *pid* for a respective shortcut is extracted from the query message as the PID of the querying peer, i.e., *qid*. Since there is no information about the number of results retrieved for the query, we set the number of query hits to 1. Finally *"r"* indicates the type of the shortcut for passive recommender shortcut and *update* is the time, when the shortcut was created or the last time, when the shortcut was used successfully. For an example consider again Fig. 3. Peer 2 issues the query /Top/Education/UML. Peer 8 creates a shortcut to peer 2 since this query was routed through peer 8.

Content Provider and Recommender Index. The volatility of the peers in the network and their interest shifts require to update the local indices. We assume that each peer may only store a limited amount of shortcuts, hence only knows a limited set of topic specific neighbors it can route a query to. If the local index size is reached a peer has to decide, which shortcut should be deleted from the index. For each shortcut in the index we compute a rank based on the following types of localities:

- *Semantic locality.* We measure the maximum semantic similarity *maxsim* between the topic of a shortcut and the topics represented by the local content of a peer according to equation (1). Hence, we retain a shortcut about topic t to a remote peer, if t is close to our own interests.
- *LRU locality.* To adapt to changes in the content and interests we use a LRU replacement policy [3]. Shortcuts that have been used recently receive a higher rank. Each local shortcut is marked with a time stamp when it was created. The time stamp will be updated, if the shortcut is used successfully by the local peer. Thus, there is an "oldest" and a "most recent" shortcut. The value $update \in [0..1]$ is normalized with difference between the shortcuts time stamp and the "oldest" time stamp divided by the difference between the "most recent" and the "oldest".
- *Community locality.* We measure how close a shortcut leads us to a document. Content provider shortcuts, marked with a c, provide a one hop distance, we set $type = 1$. Recommender shortcuts, marked with a r require at least two hops to reach a peer with relevant documents, we set $type = 0.5$.

Using a weighted moving average we weigh the different locality values and compute the index relevance according to equation (3).

$$relevance = \frac{a \cdot maxsim + b \cdot type + c \cdot update}{a + b + c} \tag{3}$$

Shortcuts with the highest relevance are ranked at the top of the index, while peers with a lower relevance are deleted from the index.

3.2 Bootstrapping Shortcuts

Bootstrapping shortcuts link to peers that have established many shortcuts for different query topics to a lot of remote peers. We determine the bootstrapping capability by analyzing the in-degree and out-degree of a peer. We use the out-degree as a measure of how successful a peer discovers other peers by querying. To weigh the out-degree, we measure the amount of distinct sources a peer receives queries from. We use the in-degree as a measure, that such a peer may share prestigious shortcuts with a high availability. By routing a query along bootstrapping shortcuts, we foster the probability to find a matching shortcut for a query and avoid the drawbacks of having to select peers randomly, e.g., by flooding.

Each incoming query that is stored in our index includes the bootstrapping information of the querying peer. While a peer is online it continually updates its content/recommender index based on incoming queries and stores additional bootstrapping shortcuts in the form *sc(pid, bo)*, where *pid* is the PID of the querying peer and *bo* it's bootstrapping capability. Once an initial set of bootstrapping nodes is found, a peer may route its queries to the nodes with the highest *bo* value. One calculates it's *bo* value using equation (4)

$$Bo = (1 + |outdegree|) \times (1 + |indegree|) \tag{4}$$

where *out-degree* is the number of distinct remote peers one's knows. To compute the *in-degree* we count the number of distinct peers that send a query via one's peer. To do this we scrutinize the pen-ultimate peers from the message path of indexed recommender shortcuts. The number of distinct pen-ultimate peers denotes one's in degree. To avoid zero values we limited the minimum for both values to one.

3.3 Default Network Shortcuts

When a new peer enters the network, it has not yet stored any specific shortcuts in its index. Default network shortcuts connect each peer *p* to a set of other peers (*p*'s neighbors) chosen at random, as in typical Gnutella-like networks (e.g., using rendezvous techniques).

4 Routing in Semantic Social Overlay Networks

The basic principle laying behind the shortcut mechanism consists of dynamically adapting the topology of the P2P network so that the peers that share common interests spontaneously form well-connected semantic communities. Reference [12] shows that each user is only interested in a rather limited number of different topics. Therefore being part of a community that shares common interests is likely to increase search efficiency and success rate. To optimize the overall message traffic we propose a dynamic shortcut selection strategy, where each peer selects only a certain number *k* of most promising shortcuts for query forwarding. Then we evaluate our approach against related approaches.

4.1 Overview

INGA consists of several steps executed locally and across the network when recommending peers for a query and retrieving or returning results. Consider a query posed to the P2P network. Necessary steps are:

- *Across the network: Recommending.* Whenever a peer receives a query message, it first extracts meta-information about the querying peer and updates its bootstrapping and recommender index if needed. Then the INGA forwarding strategy is invoked to select a set of *k* peers that appear most promising to answer the query successfully. Finally the original query message is forwarded to these *k* peers.
- *Across the network: Answering Queries.* When a peer receives a query, it will try to answer the query with local content. We only return non-empty, exact results and route them directly to the querying peer. If the maximum number of hops is not yet reached, the query is forwarded to a set of peers selected as above.
- *Locally: Receiving Results.* On the arrival of result items a querying peer analyzes the message path and the respective number of results to create or update local content provider and recommender shortcuts.

4.2 Selecting Best Matching Shortcuts

The INGA shortcut selection algorithm determines the candidate peers that are most promising to forward the given query to. The INGA strategy is based on the available local knowledge about the query topic as it is stored in the index of the peer:

- INGA only forwards a query via its *k best matching* shortcuts.
- INGA prefers content and recommender shortcuts over bootstrapping and default network shortcuts for forwarding queries.
- The INGA strategy constitutes a greedy *k best*-search heuristics. As such it might be led astray into a subnetwork of peers that appear to be the optimal choice from a local point of view, but that do not yield all the appropriate answers. To let the search escape such local optima, some queries are forwarded to a random set of peers.

  This randomness will later on show two major beneficial effects: First, it allows the individual peer to have a larger overview of the whole network and, hence, to establish the appropriate short distance *and* long distance shortcuts.[3] Second, it facilitates accommodation to volatility (especially in the form of new joining peers).

Algorithm 8 defines the basic peer selection procedure for choosing *k* peers: In step 1 it selects at most *k* peers from content or recommender shortcuts that match the

[3] “short” and “long distance” as seen from the default underlying network.

topic of the query with the highest similarity. To avoid forwarding queries along shortcuts with only low topic similarity a minimum similarity threshold t_{greedy} is required to hold between the topic(s) of the query and the shortcut. If less then k shortcuts have been found, the algorithm selects the top bootstrapping shortcuts (step 3). Finally, remaining slots for query forwarding are filled by a random selection from the default network. The algorithm terminates if the query has reached its maximum number of hops. Furthermore, the algorithm is constrained such that a query is not forwarded to a peer if this peer has already occurred in the message path of the query (step 6). We will now show all subroutines for shortcut selection in more details. Algorithm *TopGreedy* allows for selecting the top peers above a similarity threshold. The algorithm browses trough the index of all topic dependent shortcuts (step 3) and identifies the most similar shortcuts for a query (step 4) above t_{greedy} (step 5). If two shortcuts have the same similarity, it selects the shortcut with the higher value of query hits (not shown in the algorithm below). The algorithm selects the top-k peers for a query while avoiding different shortcuts with overlapping peers (step 7–8). The *TopBoot* Algorithm works similarly to the *TopGreedy* Algorithm, but selects the peers with highest known bootstrapping capability (line 4). It also avoids overlapping peers within the set of selected shortcuts (line 6–7).

Algorithm 8: Dynamic

```
Require: Query $q$, MsgPath $mp$, int $k$, float $t_{Greedy}$, float $f$, Set $topicDependentShortcuts$, Set
    $bootstrappingShortcuts$, Set $defaultNetworkShortcuts$
Ensure: $TTL_q < maxTTL$
1: Set $s \leftarrow$ TopGreedy($q$,$topicDependentShortcuts$,$k$,$t_{Greedy}$)
2: if ($|s| < k$) then
3:     $s \leftarrow s \cup$ TopBoot($bootstrappingShortcuts$, ($k - |s|$))
4: end if
5: $s \leftarrow$ RandomFill($s$,$defaultNetworkShortcuts$,$f$,$k$)
6: $s \leftarrow$ removeAlreadyVisitedPeers($s$,$mp$)
7: Return $s$.
```

The task of algorithm *RandomFill* is twofold: if the other subroutines fail to discover k peers for a query, it fills up remaining peers until k is reached (step 12–14). The second task of the algorithm is to contribute some randomly chosen peers to the selected set of k peers to avoid overfitting of the selection process as known from simulated annealing techniques. Depending on the probability f (step 5) the algorithm exchanges already selected peers with randomly chosen ones. Finally the algorithm fills up the remaining peers with randomly chosen peers (step 10–13).

5 Experimental Setup

To validate INGA and Remind'in, we have explored several possibilities for evaluation. Semantic routing has been implemented in the Bibster system, which was evaluated in a real world case study [15]. However, during the case study a maximum of 50 researchers were simultaneously online. This number was too small to evaluate the scalability of a peer-to-peer routing algorithm. Furthermore, though we have collected information about the content and the queries the peers have shared and submitted, this amount of available information was too small to test our algorithm for a larger number of peers. Therefore, we have decided to evaluate INGA by simulating a peer-to-peer network with more than 1.000 peers based on generated data sets with different ontologies and different statistical distributions of content.

Before we discuss evaluation results in Section 6, we here elaborate the experimental setup.

Algorithm 9: TopGreedy

```
Require: Query $q$, Set $topicDependentShortcuts$, int $k$, float $t_{greedy}$
Set $topShortcuts \leftarrow \{\}$
Set $s\_tmp \leftarrow topicDependentShortcuts$
Case ($s\_tmp$ is not empty) $\wedge$ ($k > 0$)
    $Next \leftarrow \text{argmax}_{p \in s\_tmp}\, \text{sim}_{\text{Topic}}(q, p)$
    if $\text{sim}_{\text{Topic}}(q, Next) > t_{greedy}$ then
        $s\_tmp \leftarrow s\_tmp - \{Next\}$
        if ($Next$ routes not to a peer in $topShortcuts$) then
            $topShortcuts \leftarrow topShortcuts \cup \{Next\}$
            $k \leftarrow k - 1$
        end if
    else
        break
    end if
end Case
Return $topShortcuts$
```

Algorithm 10: TopBoot

```
Require: Set $bootstrappingShortcuts$, int $k$
Set $topShortcuts \leftarrow \{\}$
Set $s\_tmp \leftarrow bootstrappingShortcuts$
Case ($s\_tmp$ is not empty) $\wedge$ ($k > 0$)
    $Next \leftarrow \text{argmax}_{p \in s\_tmp}\, \text{topBoot}(p)$
    $s\_tmp \leftarrow s\_tmp - \{Next\}$
    if ($Next$ routes not to a peer in $topShortcuts$) then
        $topShortcuts \leftarrow topShortcuts \cup \{Next\}$
        $k \leftarrow k - 1$
    end if
end Case
Return $topShortcuts$
```

Algorithm 11: RandomFill

```
Require: Set $preSelected$, Set $defaultNetworkShortcuts$, float $f$, int $k$
Set $postSelected \leftarrow \{\}$
Case ($preSelected$ is not empty)
    $Next \leftarrow$ next($preSelected$)
    $preSelected \leftarrow preSelected - \{Next\}$
    if ($rand(0,1) > f$) then
        $postSelected \leftarrow postSelected \cup \{Next\}$
    end if
end Case
$k \leftarrow k - |postSelected|$
Case $k > 0$
    $postSelected \leftarrow postSelected \cup$ next$\{(defaultNetworkShortcuts)\}$
    $k \leftarrow k - 1$
end Case
Return $postSelected$
```

5.1 Content Distribution

We used three different ways of assigning data to peers in our simulation runs. The assignment was done based on synthetic data sets and on real-world data sets. The data sets exhibited different characteristics with regard to dimensions like size, relational structure (i.e., being able to be used for simulation runs with conjunctive queries) and "being natural" for the task at hand. All three data sets had a low level of replication, i.e., few data items were assigned to multiple peers, and all data sets exhibited a hyperbolic (Zipf-like) distribution of topics.

1. *Open directory project.* The first data set is based on the open directory *DMOZ.org*. *DMOZ.org* constitutes a *large* data set of content distributed among a substantial community of content editors. The data set was so large that for the purpose of our simulations we have selected a subset consisting of the first three levels of the DMOZ hierarchy.
   Each editor is responsible for one or several content topics and maintains the corresponding topic pages. At the first three levels we found 1657 topics. There is a Zipf-like skew in the distribution of editors to topics: 991 editors only maintain content about one topic, 295 about two, 128 about three, ... one editor about 20, and one editor maintains content about 22 topics. Vice versa: 755 topics are dealt with by 1 editor, 333 by 2, 204 by 3, . . . , 44 by 6, . . . , 14 by 10, and 1 topic has 32 editors. Thus, mapping an individual editor onto an individual peer and the topic content of the editor onto the local node repository of the corresponding peer appears to be a rather natural, realistic choice for a basic data set.
   This data set does not contain relational structures, but only information about instantiation of a topic, e.g., "$UMLcompedium.pdf$" is an instance of topic $topic$ "$/education/UML/docs$" . Hence, it is not possible to test conjunctive queries on this data set.

2. *Synthetic data.* The number of classes, the number of properties and the number of sub-class relationships together with their respective distributions determine the schema of the ontology. The number of instances and the number of relations between instances determine the distribution of the instance data. The distributions are modeled as a Zipf distribution with parameter settings according to observations from real world data sets. The parameter settings for the schema generation are based on [33] while the parameter settings for instance generation are based on [9]. Data distribution on the peers follow the model presented in [12]. For the schema we choose to generate an ontology with 1.000 classes and each class was assigned a popularity based on a Zipf distribution with skew factor 1. The popularity of a class influences its number of instances, its replication in the network and its connectivity with other classes through properties. We selected the number of sub-classes and the number of properties of a class (Zipf with skew factor 1.1). This resulted in 357 properties.
   200,000 instances were generated and assigned to one class based on the popularity of the different classes. $TotalNoPropertyInstances = 100 \cdot TotalNo\,Properties$ many properties between instances were generated. Likewise to classes, a property schema(one could also call it a binary relation), had a popularity based on a Zipf distribution with skew factor 1 that was considered when generating properties between instances (i.e., when populating the binary relations). Assignment of data to peers is done based on the conjecture that users are generally interested in a small subset of the entire content available in a peer-to-peer network. We have modeled that the interests are more likely in only a limited number of content classes and thus users would be more interested in some classes while less in others. The maximum number of classes that a peer is interested in and its content is computed by $ClassesOfInterest = ln(NoOfClasses) * 2$. The actual number of classes is chosen randomly from a uniform distribution. Observing the studies in [9] all peers do not share the same amount of data and also do not exhibit the same 'social behaviour'. For instance, a large number of users are so-called free riders or freeloaders who do not contribute anything to the network but essentially behave like clients. On the other hand, a small number of users (less than 5%) provide more than two thirds of the totally available amount of data and thus behave like servers. Considering the study in [2] the following storage capacity was assigned to the peers in the network: 70% of the peers do not share any instances (free riders); 20% share 100 instances or less; 7% share 101 up to 1000 instances and finally, only 3% of the peers share between 1001 and 2000 instances.
3. *Bibster data set.* This data set bases on real captured query data from the peer-to-peer bibliography network "Bibster" [15]. There, we observed the behavioral characteristics of different peers. In order to run a simulation at a larger scale we re-used the observed characteristics, but on data from a large bibliography database.

In total 27.037 distinct instances represent the bibliographic entries in the network in this data set. The different topics available from an ACM-index file.[4]

5.2 Query Distribution

For the different data sets and their assignments to peers, we generated queries in the simulation runs as follows:

1. *Open directory project.* Queries were generated in the experiments by instantiating the blueprint query $(*; rdf: type; topic)$ with topics arbitrarily chosen from the set of topics that had at least one document. An example of a SeRQL query is:
   with topics arbitrarily chosen from the set of topics that had at least one document. We generated 30000 queries, uniformly distributed over the 1657 different topics. We choose a uniform query distribution instead of a ZIPF-distribution, which is typically observed in file sharing networks [28]. This simulates the worst case scenario, where we do not take advantage of often repeated queries for popular topics.
2. *Synthetic data.* The query set for the synthetic ontology is based on a special type of queries that request instances satisfying a varying number of constraints. The basic concept for the queries is built on the following schema: $(instance; rdf: type; class) \wedge (instance; owl: hasProperty; instance2) \wedge$ $(instance2; rdf: type; class2) \wedge (instance; owl: hasProperty; instance3) \wedge$ $(instance3; rdf: type; class3)$. Informally, this concept requests all *instances* of a certain *class* with the constraint that the *instances* have two particular *properties* pointing to other *instances*.
   The built query set only contains queries that can be answered by the network and is distributed uniform.
3. *Bibster data set.* The extracted queryset for the Bibster-based ontology contains several types of queries that request instances satisfying different constraints. It is worth mentioning that the queryset was adopted the way it was originally created and therefore, it also consists of non-conjunctive queries.

5.3 Peer-to-Peer Network Setup

Gnutella style network. The simulation is initialized with a network topology which resembles the small world properties of file sharing networks.[5]We simulated 1024

[4] The ACM-Index file is available online at:
http://www.aifb.uni-karlsruhe.de/WBS/pha/bib/acmtopics.rdf

[5] We used the Colt library `http://acs.lbl.gov/~hoschek/colt/`

peers. In the simulation, peers were chosen randomly and they were given a randomly selected query to question the remote peers in the network. The peers decide on the basis of their local short cut which remote peers to send the query to. Each peer uses INGA to select up to $pmax = 2$ peers to send the query to. Each query was forwarded until the maximal number of hops $hmax = 6$ was reached – unless the peer selection algorithm choose not to forward further also at an earlier point in time.

Volatile network and interest shifts. The simulation incorporates dynamic network model observed for Gnutella networks by [28] and models incorporating IP-address aliases found by [8]. Hence only a small fraction of peers is available more than half of the simulation time, while the majority of the peers is only online a fraction of the simulation time. The real online and off-line times of peers are determined randomly at simulation runtime so that their cumulative online times resemble the distribution found in [28]. Figure 4 visualizes the number of peers online during the simulation run. A peer ensures before it forwards a query that it is connected to at least five online remote peers on the network layer. If a remote peer is off-line it is exchanged with an online peer. A peer discovers with rendezvous techniques online peers on the network layer.

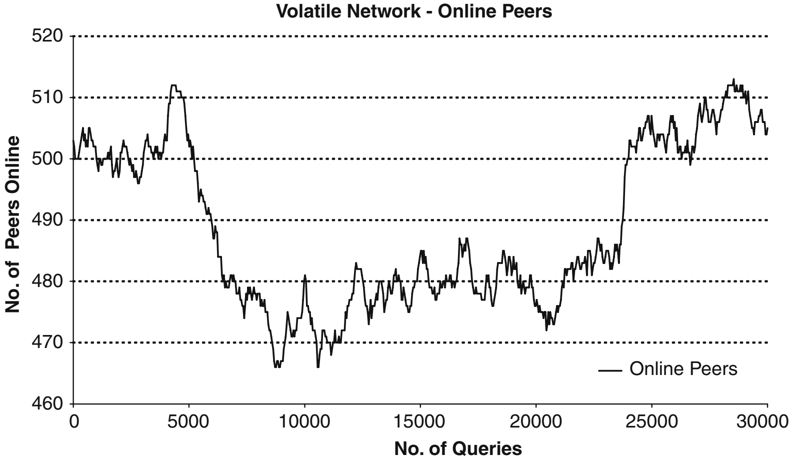


Fig. 4 Volatile network: number of peers online

We also consider changing user interests. Users' interests may change over time, e.g., to account for different search goals. To simulate changing interests, after 15 queries, equal to ca. 15.000 queries over all peers, each peer queries a completely different, previously unused set of topics.

5.4 Simulator Setup and Simulation Statistics

We used a round based simulation framework. Figure 1 presents the main parameters of the simulation framework. In total we simulated 1024 peers. To determine the standard error of our observations of 95% confidence interval ($p<0.05$) each simulation was executed six times. We set the greedy search threshold for algorithm to 0.15 and the amount of random contribution to 0.20

Parameter	Value
Queries	30.000
Queries per peer	ca. 30
Query time to life	6
Selected peers per query (k)	2
Greedy search threshold (t_{Greedy})	15%
Random contribution (f)	20%
Index size (if no other size is mentioned)	40
Open Directory data set	
Topics	1646
Before interest shift	823
After interest shift	823
Simulated peers	1024
Bibster data set	
Topics	1293
Queries	3319
Simulated Peers	520
Synthetic data set	
Topics	1000
Queries before interest shift	1641
Queries after interest shift	1640
Simulated Peers	1024

Table 1 Simulation parameter setting

5.5 Evaluation Measures

We measure the search efficiency using the following metrics:

- *Recall* describes the proportion between all relevant documents in peer network and the retrieved ones. Hereby, we defined "relevant" as "matches the query". We did not use any gold standard document set where relevance to a query would be assigned by a user. Therefore, *precision* would have been meaningless in our evaluation.

- *Messages* represent the required search costs per query that can be used to indirectly justify the system scalability.
- *Message Gain* is defined as the recall per message, hence we divide the recall of a query with the proportion of messages to achieve the recall.
- *Clustering coefficient* represents the compactness of the network. It captures how many of a node's neighbors are connected to each other. We define the clustering coefficient as

$$\mathscr{C} = \frac{1}{|V|} \sum_{v \in V} \frac{|E(\Gamma_v)|}{k_v \cdot (k_v - 1)} \tag{5}$$

  where V denotes the set of peers in the network, k_v denotes the maximum number of shortcuts for a peer v, Γ_v the direct neighbors of a peer and $E(\Gamma_v)$ represents a function that counts the number of links in Γ_v.
- *Average path length* A short average path length denotes a highly directed information flow between two peers in the network. Given two arbitrary selected peers $v_1, v_2 \in V$ and $d_{min}(v_1, v_2)$ the minimum path length between v_1 and v_2, we define the average path length as

$$d = \frac{1}{\genfrac{}{}{0pt}{}{|V|}{2}} \sum_{v_1 \neq v_2} d_{min}(v_1, v_2) \tag{6}$$

6 Evaluation and Optimization

We conducted a large number of experiments regarding the performance of INGA in contrast to state-of-the-art approaches, the optimal parameter setting and the applicability of our algorithm to different scenarios. Before we present the final evaluation results, we here summarize the major hypotheses we wanted to investigate:

1. Shortcut networks outperform the *Default* approach.
2. INGA outperforms state-of the art shortcut networks.
3. Semantic similarity supports the peer selection process. It helps to improve recall and to reduce the number of messages. However, shortcut networks perform reasonably even without the support of a semantic similarity function.
4. Each layer contributes to improve routing efficiency. Depending on the scenario, dynamic bootstrapping peers help to reduce the number of messages, while recommender peers increase the recall.
5. Shortcut networks show small world characteristics.
6. Our algorithms performs well with a limited index size.
7. Combining different index policies supports efficient routing much more than relying on a simple LRU strategy.
8. Shortcut networks are capable to handle conjunctive queries efficiently.
9. Shortcut networks perform well in both, dynamic and static, networks.

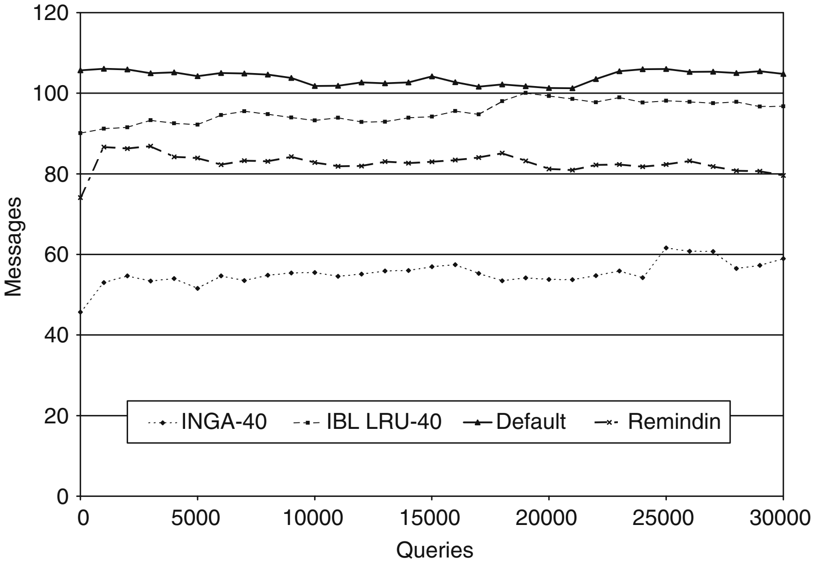


Fig. 5 Messages: Related Approaches

6.1 Performance Against State-of-the-Art Approaches

As a baseline we compare INGA with an index size of 40 entries against the interest based locality strategy (IBL) of [29] with an LRU strategy and an index size of 40 entries, the default algorithm of Gnutella (*Default*) and Remind'in [32].

Figure 6 shows the recall in contrast to the maximum possible recall in a dynamic network. After only 15 queries per peer, INGA nearly doubles the recall of the default approach and drastically outperforms *IBL*. Since INGA and Remind'in use similar strategies for creating shortcuts both achieve a similar recall. Figure 5 shows the number of messages. Due to bootstrapping peers, which focus queries to a fraction of peers in the network, INGA outperforms and halves the messages in contrast to a default approach. In contrast to Remind'in, INGA reduces the number of messages from about 85 to 58 messages in average. While the maximum possible recall in our dynamic scenario is ca. 55% after 15 queries of each peer we achieved a recall of ca. 25%. One reason for this decent recall is due to the very low replication of topics in our data set. Furthermore to stress our algorithm further, we introduce a hard topic shift after 15 queries. However another study [21] shows a higher replication and a smoother interest shift, both factors would raise the recall further.

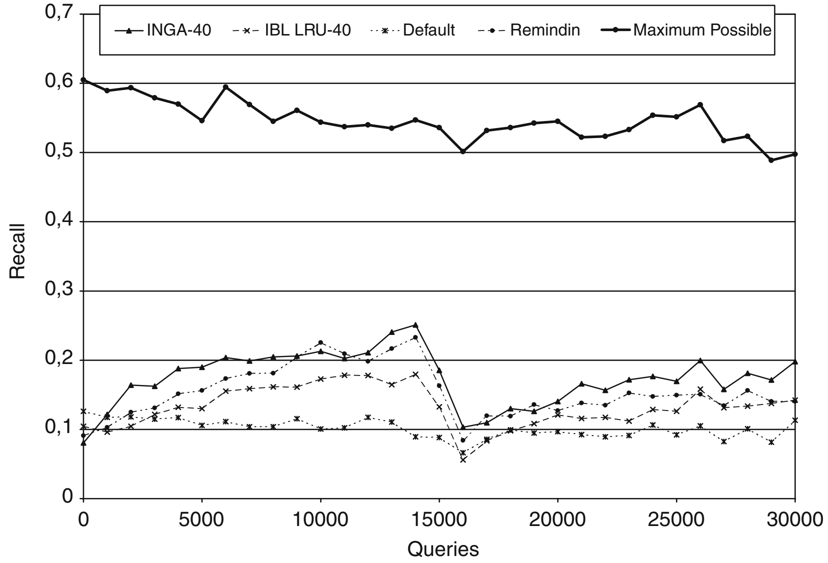


Fig. 6 Recall: related approaches

6.2 Layer and Semantic Similarity Function Contribution

Figure 7 shows the message gain of the different layers. Only using content provider shortcuts (Content-40) performs poorly, a combination of content and recommender shortcuts raise the message gain (Content Recommender-40) and finally the introduction of bootstrapping peers (INGA-40) additionally boosts INGA performance. The introduction of the recommender layer has the strongest influence on the performance of INGA. One reason is the caching of frequent queries issued by the peers in the network. If the network becomes clustered, especially neighboring nodes with similar interests will benefit of cached queries. Furthermore query routing is based on a similarity function, so queries are routed along shortcuts representing similar queries. Especially for nodes that are not clustered so far, this similarity helps to find adequate clusters and route queries to nodes that have similar interests. Thus, including a similarity in the peer selection process speeds up the clustering process, however shortcut networks will exploit small world characteristics even if an exact match paradigm is used only. In this case shortcut networks profit in particular from popular queries that are used to establish topic specific shortcut between remote peers.

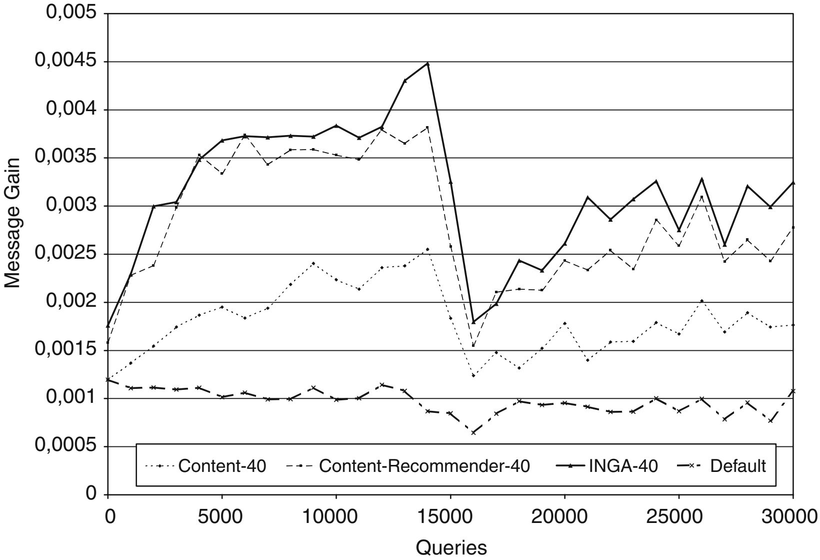


Fig. 7 Message gain for each layer

6.3 Tradeoff Between Clustering and Recall

Small-world graphs are defined by comparison with random graphs with the same number of nodes and edges: first, a small-world displays a small average path length, similar to a random graph; second, a small-world has a significantly larger clustering coefficient than a random graph of the same size [16]. To measure the small world characteristics for different index settings we conducted experiments where we only consider the similarity locality (a = 10, b = 0, c = 0), only community locality (with a = 0, b = 10, c = 0), only LRU-locality (a = 0, b = 0, c = 10) and the combination (a = 3, b = 6, c = 1), that produces in our dynamic setting optimal results in terms of messages and recall.

We discover that the INGA data-sharing graph displays small-world properties. Figure 8 shows, that all index settings reduce the average path length in contrast to the default network. Due to the peer dynamics the default network selects the most stable peers with a path length of four hops while all index settings reduce the path length to 2 hops. However, *INGA LRU-40* stabilizes less than the other approaches.

The clustering coefficient (cf. Fig. 9) increases for most of our index configurations. Only *INGA LRU-40* decreases after a slight increase the clustering coefficient. Hence a LRU strategy alone is not able to create a highly clustered network. However, a high clustering coefficient does not correlate with a high recall: Figure 9 shows that the high clustering coefficient of *INGA SIM-40* outperforms while Fig. 10 shows that the highest recall is achieved through the optimal setting *INGA 40*. Since clustering in the network focusses queries to a small set of peers storing similar shortcuts it reduces the number of randomly discovered peers as well.

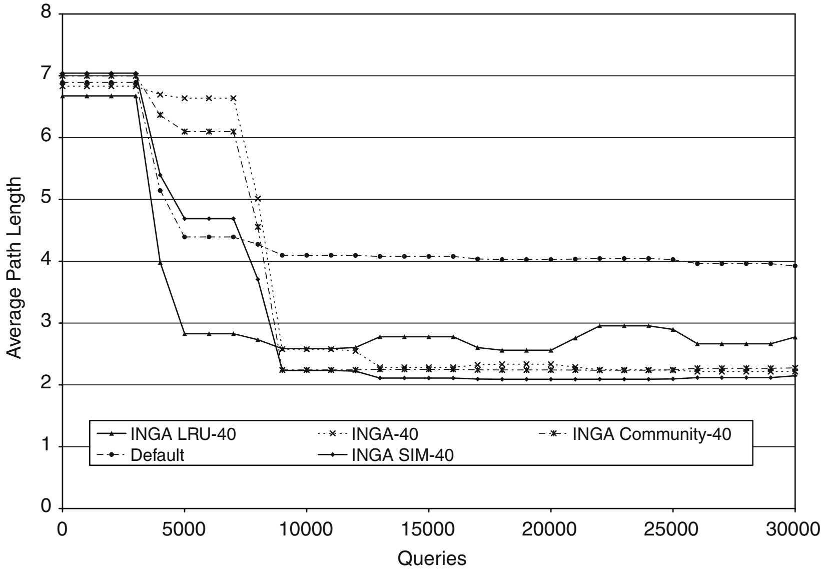


Fig. 8 Average path length for different index weights

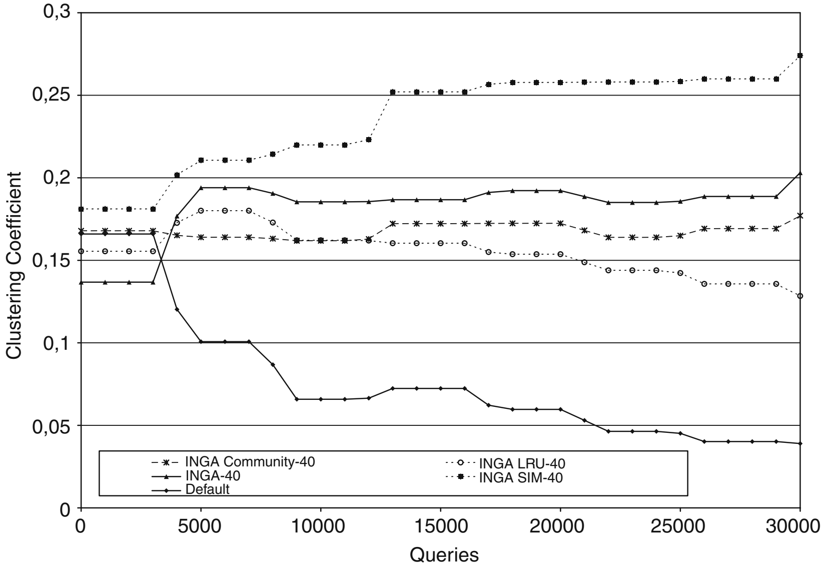


Fig. 9 Clustering coefficient for different index weights

However, such randomness is crucial in a highly dynamic setting to achieve a high document recall. For applications that prefer a high document recall, we recommend the setting of *INGA 40*.

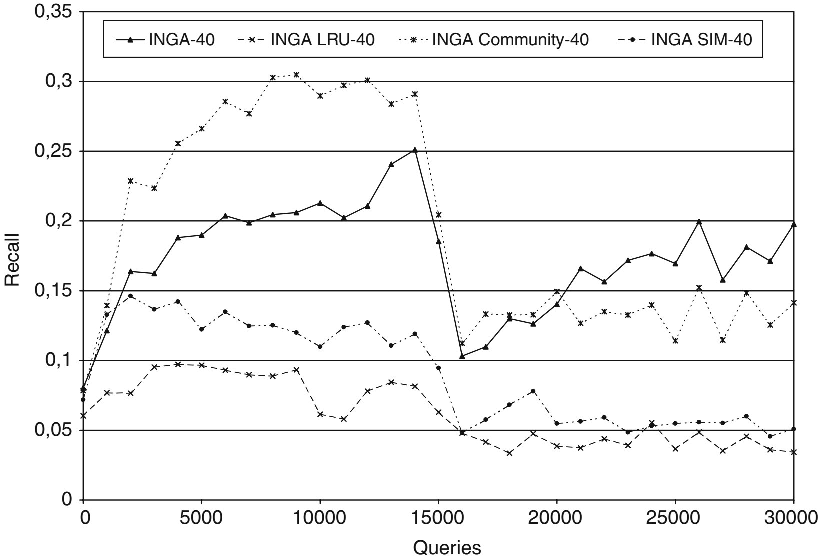


Fig. 10 Recall for different index weights

6.4 Setting Optimal Index Size

In this experiment observe the effect of different index sizes in the message gain of INGA. We found out that limiting index size performs similar to an unlimited

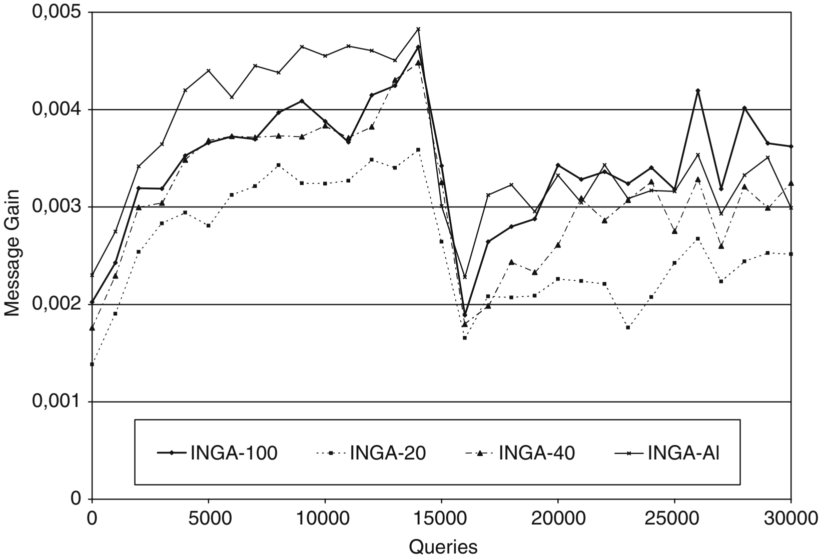


Fig. 11 Message gain for different index size

index. We conducted experiments with an unlimited index size and a maximum size of 100, 40, 20 shortcut entries. To message the tradeoff between messages and recall we use the message gain again. Figure 11 shows that an index size of 100 entries performs as good as an unlimited index while an index of 40 entries still is a reasonable tradeoff between size and routing efficiency.

6.5 Setting Optimal Index Weight

To determine an optimal weighting of the parameter (a,b,c) of the index policy, we conducted experiments where we only consider the similarity locality ($a = 10, b = 0, c = 0$) , where we only consider the community locality (with $a = 0, b = 10, c = 0$), where we only consider the LRU-Locality ($a = 0, b = 0, c = 10$) and an "optimal" combination ($a = 3, b = 6, c = 1$). Reference [29] proposes a LRU strategy to update the index. We found out that there are better strategies. Figure 12 shows a similarity and LRU strategy, both perform worse and are alone not capable to adopt to the dynamics of the network and the changing interests of each peer. The community locality raises the message gain, even after changing the interests of each peer, while the combined strategy performs best.

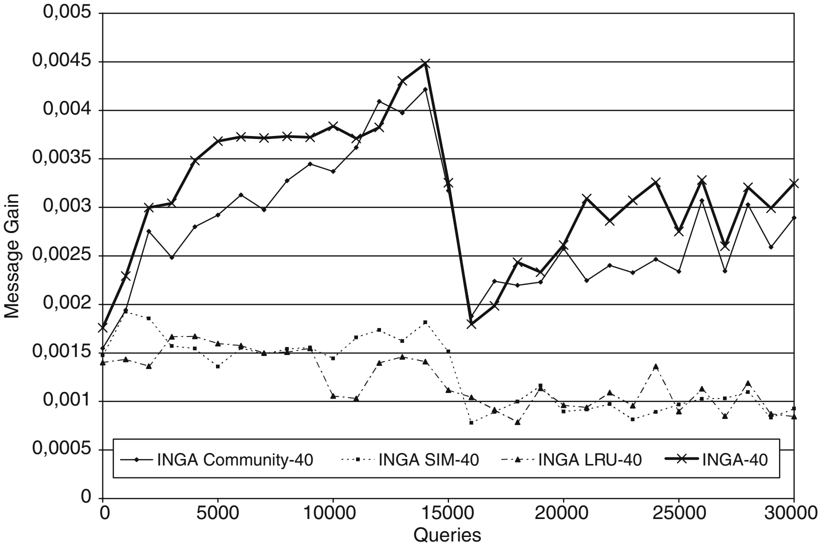


Fig. 12 Message gain for different index weights

6.6 Performance for Conjunctive Queries

In Fig. 13 we have plotted the results comparing the recall for different selection strategies. We refer to the ranking based on equation (2) as *Multiply*. After a warm up phase of 2.000 queries, or approximately two queries per peer, we constantly reach around 75% of the available content. Not all peers are always online, thus we have plotted the maximum available content as *Online Available* in the graph. We compare our selection strategy to the *Default* approach, which represents the baseline for all algorithms likewise. In this approach we use only the default layer to select peers as in Gnutella. From the known peers on the default layer, we randomly select two remote peers to send and forward a query to.

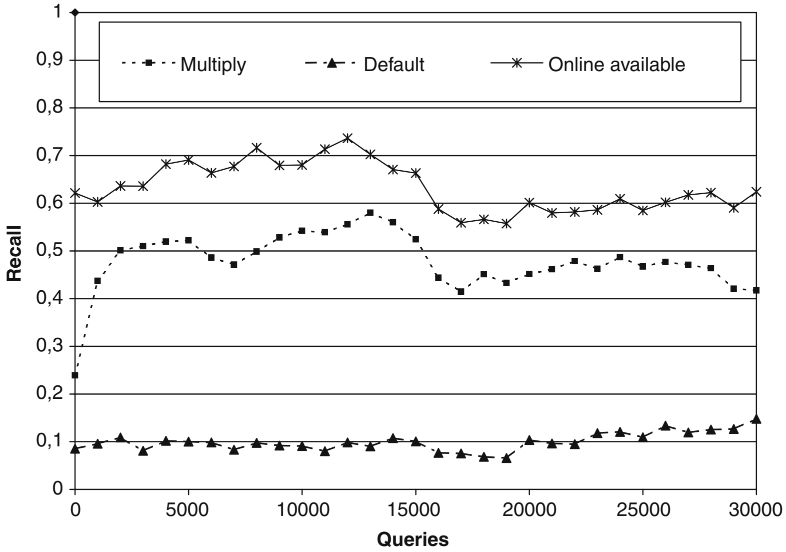


Fig. 13 Recall for conjunctive queries

In Fig. 14 we have plotted the number of messages produced to achieve this recall. The number of messages produced by the *Default* approach slightly increases over time. Due to the high network churn the peers have to discover new remote peers since the available ones assigned in the setup phase are off line. Thus, the necessity to send a query to a remote peer which has not received the query yet increases over time. In contrast to the observation made for the *Default* approaches the number of messages produced based on the shortcut selection decreases significantly. The number of messages decreases because of the small world properties of the network queries are only forwarded to a focused set of peers. I decreases further, since we do not forward queries to peers that have already received a query (see Section 4 and algorithm 8).

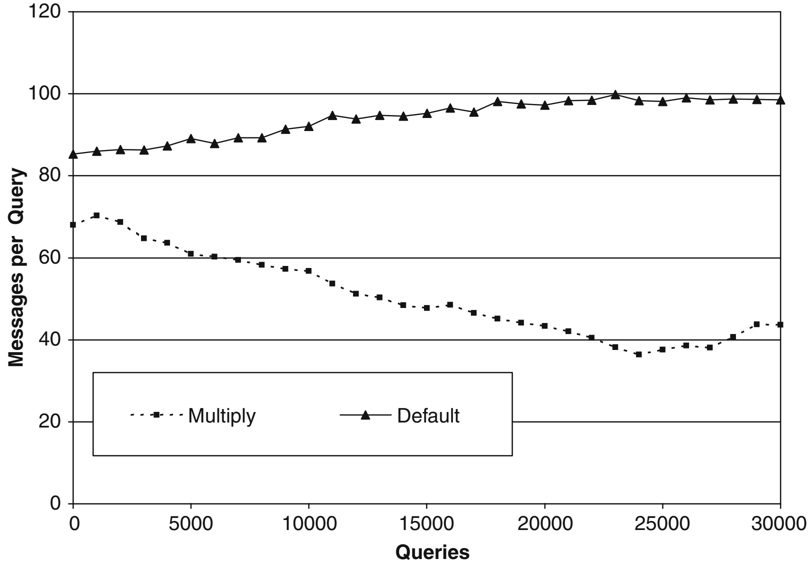


Fig. 14 Messages for conjunctive queries

7 Related Work

Their exist a number of different approaches to routing in P2P networks. They can be categorized according to the two dimensions *centralization* and *structure*. The discussion of related routing algorithms follows this categorization. and was influenced by the surveys [4, 13, 26, 34].

Routing Algorithms for Centralized Peer-to-Peer Networks. First approaches for efficient indexing in P2P architectures were central indices, that have to transmit either meta data about the available content to central indexing peers, like, e.g., GlOSS [14] or *Napster*.

Routing Algorithms for Structured Decentralized Peer-to-Peer Networks. One of today's main technique for indexing P2P systems are so-called distributed hash tables (DHTs), (e.g., [1] or see [5] for a survey) that without need of a central index allows to route queries with certain keys to particular peers containing the desired data. But to provide this functionality all new content in the network has to be published at the node for the respective key, if new data on a peer arrives or a new peer joins the network. And in case that a peer leaves the network the information about its content has to be unpublished. Recent research in [22] shows that due to the publishing/unpublishing overhead, DHTs lack efficiency when highly replicated items are requested and in practical settings perform even worse than flooding approaches degrading further if network churn is introduced.

Routing Algorithms for Super-Peer-Based Peer-to-Peer Networks. The system EDUTELLA uses a super-peer-based routing mechanism based [27]. Peers which have topics in common are arranged in a hypercube topology. This topology guarantees that each node is queried exactly once for each query. It is thus a very efficient approach to flood queries to all online peers. From the requirements perspective this solution for query routing is not adequate in our use case as the peers have to publish their expertise to the super-peers. Their are also technical problems in highly volatile P2P environments, as the hypercube topology must be maintained. Our algorithm is not based on an explicit topology, thus it does not generate any overhead to establish it. Our simulations illustrate that we need much less messages per query than the number of peers available in the network in order to reach the most knowledgeable ones. INGA cannot ensure a complete answer, though.

A second routing algorithm for the EDUTELLA system uses content provider shortcuts for document retrieval [6]. Only exact matches between content provider shortcuts and queries are considered for peer selection. The peers publish their local indices in a static super-peer network. The shortcut index policy considers temporal locality, each index entry has a certain time to live after which the shortcut has to be reestablished for the next query on that topic. In contrast to INGA they only consider exact matches between shortcuts and queries, hence they do not investigate complex ranking metrics for semantically similar shortcuts. They do not use the concept of recommender peers.

Routing Algorithms for Unstructured Decentralized Peer-to-Peer Networks. While the visualization of keys and objects in the same name space used in structured overlays provides an elegant clean solution to routing within logarithmical bounds it comes at the significant cost of destroying the locality of the content: Content at a user's desktop is co-located with other relevant items, structured overlays destroy this locality meaning that enhanced opportunities for browsing and pre-fetching are lost [17]. Unstructured networks, such as Gnutella, keep this locality, since a query is forwarded to randomly picked neighbors. To bound the number of hops it can travel, each query is tagged with a maximum number of hops (TTL). In addition Gnutella employs a duplicate detection mechanism, so that peers do not forward queries that they have already previously forwarded. To improve the efficiency of Gnutella routing indices local index information are first introduced by [11]. This indexing strategy locally stores information about specific queries and what peers were successfully queried in the past.

In the work described here, we do not make any assumptions about this infrastructure or the overlay topology, and rather assume that the query routing decision has all the information about other peers that it needs and chooses peers solely by benefit/cost considerations. We will disregard the cost aspects for this paper and focus on the much less explored benefit issues. Following the authors of [7] we distinguish three broad families of strategies:

- *Semantic query routing:* The peers to which a query is forwarded are chosen based on the *content similarity* between the query and the data held by the candidate target peers (or the corresponding peer synopses).
- *Social query routing:* The target peers are chosen based on *social relationships* like the explicitly listed friends of the query initiator or peers that belong to the same explicit groups.

- *Spiritual query routing:* The target peers are chosen based on behavioral affinity such as *high overlap* in tag usage, bookmarked pages, or commenting and rating activity.

The authors in [29] first consider the semantics of the query to exploit interest-based locality in a static network. They use shortcuts that are generated after each successful query and are used to further requests, hence their strategy generates what we call content provider shortcuts (cf. Section 3). However their search strategy differs from ours, since they only follow a shortcut if it exactly matches a query, else they use a flooding approach. To update the index they use a LRU strategy. Similar, [6] uses a local routing index for content provider shortcuts for the specific scenario of top k retrieval in P2P networks. Local indices are maintained in a static super-peer network. Their index policy considers temporal locality, each index entry has a certain time to live after which the shortcut has to be reestablished for the next query on that topic. REMINDIN [32] used a routing table storing content provider shortcuts and a relaxation based routing strategy. The approach was only designed for a static setting without any index size limitation, an assumptions that is not realistic. The authors in [18] introduce the notion of peer communities that consist of active peers involved in sharing, communicating, and promoting common interests. These communities are self-organizing using distributed formation and discovery algorithms. Finally, the authors of [7] combine semantic social and spritual query routing for delivering high-quality results. Instead of a top-level ontology, they consider keyword queries and use IR quality measures like precision and recall. In their work the present hybrid strategies that combine elements from both semantic and social or semantic and spiritual search.

8 Summary

The novel design principle of the INGA query routing strategy lies in the dynamic adaptation of the network topology, driven by the history of successful or semantically similar queries. This is realized by using bounded local shortcut indexes storing semantically labeled shortcuts and a dynamic shortcut selection strategy, which forwards queries to a community of peers that are most promising to best answer a given query. Shortcuts connect peers that share similar interests and thus spontaneously form semantic communities that show typical small world characteristics, e.g., a high clustering coefficient and a low average path length. The clustering of peers within semantic communities drastically improves the overall performance of our algorithm even in a highly volatile setting.

In extensive simulations with different index strategies we have shown a trade-off between recall and clustering: Especially in volatile networks “over clustering” may easily lead the query into a subnetwork of peers that constitutes a locally, but not globally optimal choice of peers, thus reducing the recall for a given query. At the same time INGA is non-intrusive as it is solely based on the observation of network behavior keeping the load for administration messages at nil.

Thus, INGA exhibits a characteristics of properties that sets it apart in intriguing ways from structured DHT-style networks and it may provide a substantial benefit to all unstructured peer-to-peer networks.

Acknowledgment

Research reported in this paper has been partially financed by EU in the IST projects SEKT (IST-2003-506826), SWAP (IST-2001-34103) and ASG (IST FP6-004617). Alexander Löser was supported by the German Research Foundation, Berlin-Brandenburg School for Distributed Information Systems (DFG grant GRK 316/3).

References

1. Aberer, K., Cudre-Mauroux, P., Hauswirth, M., van Pelt, T.: GridVine: Building Internet-Scale Semantic Overlay Networks. In: 3rd. International Semantic Web Conference (ISWC), Hiroshima, Japan (2004)
2. Adar, E., Huberman, B.: Free riding on gnutella. First Monday **5**(10) (2000)
3. Aho, A.V., Denning, P.J., Ullman, J.D.: Principles of optimal page replacement. J. ACM **18**(1), 80–93 (1971). DOI http://doi.acm.org/10.1145/321623.321632
4. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Comput. Surv. **36**(4), 335–371 (2004)
5. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Comput. Surv. **36**(4), 335–371 (2004). DOI http://doi.acm.org/10.1145/1041680.1041681
6. Balke, W.T., Nejdl, W., Siberski, W., Thaden, U.: Progressive distributed top-k retrieval in peer-to-peer networks. In: 21st International Conference on Data Engineering (ICDE). Tokyo, Japan (2005)
7. Bender, M., Crecelius, T., Kacimi, M., Michel, S., Parreira, J.X., Weikum, G.: Peer-to-peer information search: Semantic, social, or spiritual? IEEE Data Eng. Bull. **30**(2), 51–60 (2007)
8. Bhagwan, R., Savage, S., Voelker, G.M.: Understanding availability. In: IPTPS, pp. 256–267 (2003)
9. Cholvi, V., Felber, P., Biersack, E.: Efficient search in unstructured peer-to-peer networks. European Transactions on Telecommunications: Special Issue on P2P Networking and P2P Services **15**(6), 535–548 (2004)
10. Cooper, B.: Guiding queries to information sources with InfoBeacons. In: ACM/IFIP/USENIX 5th International Middleware Conference. Toronto (2004)
11. Crespo, A., Garcia-Molina, H.: Routing indices for peer-to-peer systems. In: International Conference on Distributed Computing Systems (2002)
12. Crespo, A., Garcia-Molina, H.: Semantic Overlay Networks for P2P Systems. Tech. rep., Computer Science Department, Stanford University (2002)
13. (Ed.), A.O.: Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly, Sebastopol, USA (2001)
14. Gravano, L., García-Molina, H.: Generalizing GlOSS to vector-space databases and broker hierarchies. In: International Conference on Very Large Databases, VLDB, pp. 78–89 (1995)
15. Haase, P., Broekstra, J., M.Ehrig, Menken, M., P.Mika, Plechawski, M., Pyszlak, P., Schnizler, B., Siebes, R., Staab, S., Tempich, C.: Bibster – a semantics-based bibliographic peer-

to-peer system. In: Proceedings of the International Semantic Web Conference, pp. 345–354. Hiroshima, Japan (2004)

16. Iamnitchi, A., Ripeanu, M., Foster, I.: Small-World File-Sharing Communities. In: 23th. IEEE InfoCom HongKong (2004)
17. Keleher, P.J., Bhattacharjee, B., Silaghi, B.D.: Are virtualized overlay networks too much of a good thing? In: IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems, pp. 225–231. Springer-Verlag (2002)
18. Khambatti, M., Ryu, K.D., Dasgupta, P.: Structuring peer-to-peer networks using interest-based communities. In: DBISP2P, pp. 48–63 (2003)
19. Kleinberg, J.: Navigation in a small world. Nature **406**(6798), 845 (2000)
20. Li, Y., Bandar, Z., McLean, D.: An Approach for messuring semantic similarity between words using semantic multiple information sources. In: IEEE Transactions on Knowledge and Data Engineering, vol. 15 (2003)
21. Liang, J., Kumar, R., Xi, Y., Ross, K.: Pollution in p2p file sharing systems. In: IEEE INFOCOM. Miami, FL (2005)
22. Loo, B., Hellerstein, J., Huebsch, R., Shenker, S., Stoica, I.: Enhancing p2p file-sharing with an internet-scale query processor. In: In Proc. of Int. Conf. on Very Large Databases (VLDB). Toronto (2004)
23. Loser, A., Staab, S., Tempich, C.: Semantic social overlay networks. Selected Areas in Communications, IEEE J. **25**(1), 5–14 (2007). DOI 10.1109/JSAC.2007.070102
24. Löser, A., Tempich, C., Quilitz, B., Balke, W.T., Staab, S., Nejdl, W.: Searching dynamic communities with personal indexes. In: International Semantic Web Conference, pp. 491–505 (2005)
25. Milgram, S.: The small world problem. Psychol. Today **67**(1) (1967)
26. Milojicic, D.S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., Xu, Z.: Peer-to-peer computing. Tech. rep., HP Laboratories Palo Alto, Technical Report HPL-2002-57 (2002)
27. Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M.T., Brunkhorst, I., Löser, A.: Super-peer-based routing strategies for rdf-based peer-to-peer networks. J. Web Sem. **1**(2), 177–186 (2004)
28. Saroiu, S., Gummadi, P.K., Gribble, S.D.: A measurement study of peer-to-peer file sharing systems. Multimedia Systems **9**(2) (2003)
29. Sripanidkulchai, K., Maggs, B., Zhang, H.: Efficient Content Location Using Interest Based Locality in Peer-to-Peer System. In: Infocom. IEEE (2003)
30. Tempich, C., Ehrig, M., Fluit, C., Haase, P., Marti, E.L., Plechawski, M., Staab, S.: Xarop: A midterm report in introducing a decentralized semantics-based knowledge sharing application. In: Practical Aspects of Knowledge Management: 5th International Conference, PAKM 2004, Vienna, Austria, no. 3336 in Lecture Notes in Computer Science (2004)
31. Tempich, C., Löser, A., Heizmann, J.: Community Based Ranking in Peer-to-Peer Networks. In: Ontologies, Databases and Applications of Semantics (ODBASE) 2005, *LNCS*, vol. 3761. Springer (2005)
32. Tempich, C., Staab, S., Wranik, A.: REMINDIN:Semantic Query Routing in Peer-to-Peer Networks based on Social Metaphers. In: Proceedings of the 13th WWW Conference New York. ACM (2004)
33. Tempich, C., Volz, R.: Towards a benchmark for semantic web reasoners – an analysis of the DAML ontology library. In: Y. Sure (ed.) Evaluation of Ontology-based Tools (EON2003) at Second International Semantic Web Conference (ISWC 2003) (2003)
34. Tsoumakos, D., Roussopoulos, N.: Adaptive probabilistic search for peer-to-peer networks. In: Peer-to-Peer Computing, pp. 102–109 (2003)

Part III
Structured P2P Overlay Architectures

Overview of Structured Peer-to-Peer Overlay Algorithms

Krishna Dhara, Yang Guo, Mario Kolberg, and Xiaotao Wu

Abstract This chapter provides an overview of structured Peer-to-Peer overlay algorithms. The chapter introduces basic concepts including geometries, routing algorithms, routing table maintenance, node join/leave behaviour, and bootstrapping of structured Peer-to-Peer overlay algorithms. Based on these key concepts, a number of key overlay algorithms are classified into categories and a brief over-view of these algorithms is presented. Finally, the chapter presents an "on-a-glance" comparison of the presented algorithms and provides an outlook on open research issues.

1 Overview

Large scale peer-to-peer systems have been deployed for file, music, and other data sharing applications over the internet. The core of these systems is a peer-to-peer network overlay that could connect millions of users or systems and a network that could dynamically discover data stored at any node. Early versions of such peer-to-peer systems mainly consisted of unstructured overlays that organize nodes into random data structures. These unstructured overlays use techniques such as walking or flooding the nodes in the system for lookup, and are often optimized for some common lookup queries. But, in general, these unstructured overlays are quite un-

Krishna Dhara
Avaya Labs Research, Basking Ridge, NJ 07920, USA, e-mail: dhara@avaya.com

Yang Guo
Thomson Corporate Research, Princeton, NJ 08542, USA, e-mail: yang.guo@thompson.net

Mario Kolberg
University of Stirling, Stirling, UK, e-mail: mko@cs.stir.ac.uk

Xiaotao Wu
Avaya Labs Research, Basking Ridge, NJ 07920, USA, e-mail: xwu@avaya.com

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_9, © Springer Science+Business Media, LLC 2010

predictable for finding rare items and for some real-time applications such as voice, video sharing etc.

To overcome these issues, structured overlays are developed to provide deterministic bounds on the data discovery. Structured overlays provide scalable network overlays based on a distributed data structure that supports deterministic behaviour for data lookup. Structured P2P overlays impose restrictions on node placement in the overlay and hence, improve the efficiency of data lookup. In this chapter we take a closer look at these structured peer-to-peer overlays. Earlier surveys of structured overlays can be found in [1–4]. Here, we present different geometries and their effect on the performance of structured P2P systems. We categorize structured P2P systems in terms of the bound on numbers of hops required for data lookup and present issues such as node lookup, finger table maintenance, and join/leave properties of the overlays. First we define various terms used in structured P2P systems and present the basic notions of a structured peer-to-peer system. We then introduce various classes of structured overlays and discuss their relative merits in the last section.

Some terms and notions are often used when describing P2P overlay algorithms. The most common ones are described briefly below.

Structured P2P overlay: A network overlay that connects nodes using a particular data structure or protocol to ensure that node lookup or data discovery is deterministic.

Distributed hash table (DHT): A decentralized or distributed hash table that stores (key, value) pairs and is used for data lookups using a key.

Key-based routing: The principle by which a message is routed to the owner of a key k from a node n following the principle that either the node n owns the key or points to a node that is closer to a node that owns k in terms of some key space defined by the DHT.

Routing table (Finger table): Data structure, usually a table, at nodes that maintain links to other nodes in the structure.

Churn: Rate of node joins and leaves in a peer-to-peer network.

2 Basic Features of Structured P2P Overlays/Networks

One way to understand structured P2P overlays/networks and to compare various such systems is to study their defining aspects. These aspects include the geometries or data structures used in overlays, the routing algorithms that are enabled by these data structures, the affects of churn on various geometries, the maintenance of the data structure, and the bootstrapping mechanism. These aspects collectively describe the behaviour of structured P2P overlays. In this section, we present what each of these aspects are and how they impact P2P performance in terms of lookup speed, space consumption, and bandwidth requirement. Notations introduced herein are used in later sections to describe various representative structured P2P overlays.

2.1 Geometries

Structured P2P overlays use a number of different geometries to accommodate participating nodes in P2P overlays. The term geometry is referred to as a structure to organize nodes in a P2P overlay. The primary goal of these geometries is to enable the deterministic lookup. The cost or performance of lookups in a structured P2P overlay is directly related to how nodes are arranged and how the geometry is maintained when new nodes arrive and when old nodes leave. Further, these geometries have a direct impact on the space requirements and on the churn performance of the P2P overlay.

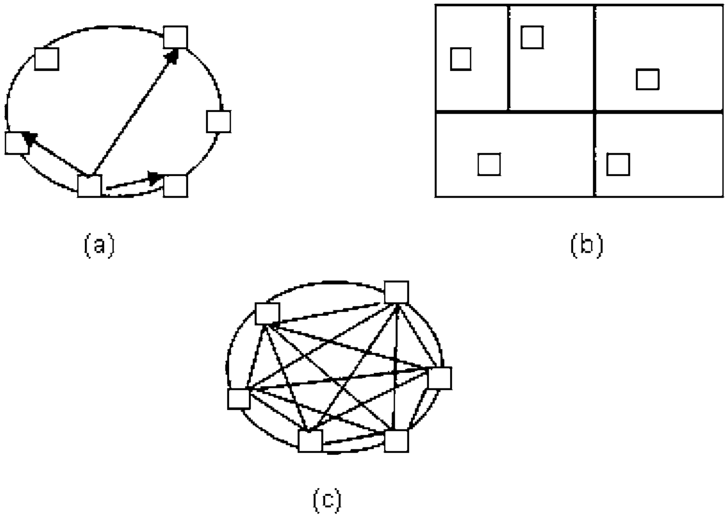


Fig. 1 Examples of structured P2P overlay geometries

Figure 1 shows a few examples of P2P geometries. As depicted in the figure, nodes can be organized in various ways. There are two ways of looking at these structures. One is how the nodes are mapped. In other words how a search of this space proceeds. Another way, though closely related but distinct, is to look at the connectivity of these nodes. For example in Fig. 1a nodes are organized in a way such that the lookups proceed clock-wise in powers of 2. Each node knows only about a certain number of other nodes in the network. In Fig. 1b lookups proceed in powers of 2 but in a geometric space. Figure 1c shows a node organisation with a high connectivity among nodes. Here the lookups are relatively simple and often only take a single overlay hop.

While structured peer-to-peer overlays offer a uniform distribution of nodes that help in the lookup, the costs associated with the distribution and lookups need to be balanced with the performance under churn, network latency, and space. The scale of peer-to-peer overlays requires significant efforts in maintaining membership changes in overlays. This aspect is further enhanced by the churn of nodes. One way to mitigate this problem is to maintain small tables that do not require high maintenance. However, small routing tables increase the lookup latency, which is often O(log N). If the size of routing tables is increased, then structured overlays can reduce the lookup latency as fewer overlay hops are required to reach the destination node. The latency cost can vary between O(log N) to O(1). There are some variations to the O(1) latency overlay algorithms that minimize peer dynamics.

Logarithmic Overlays: Structured overlays use different approaches to route objects. A class of overlays reduce the lookup space by half in each step resulting in a logarithmic number of hops (based on the number of nodes in the overlay). Such overlays are referred to as logarithmic overlays and they guarantee on average $O(logN)$ hops for lookups. Examples of such logarithmic overlays are Chord, Pastry, Tapestry. While Chord uniformly distributes a node across the search space, overlay algorithms like Pastry and Tapestry exploit inter-node proximity while choosing the node's routing table entries. While the average number of hops remains in the same order of complexity with this approach, lower network latency reduces the routing and maintenance costs.

O(1) Overlays and Constant Overlays: In cases where the peer churn is low, the size of the overlay is relatively small, or network latencies for high bandwidth nodes make the routing table maintenance less expensive, constant or O(1) overlays become practical. There are studies that shows that for overlays with millions nodes or more the bandwidth requirements become large and the O(1) overlays become expensive [5] and multi-hop approaches might be preferable. After initial studies on O(log N) overlays, currently there is extensive research on minimizing lookup latency and optimizing the table maintenance costs using constant overlays [6, 7].

2.2 Routing Algorithm

Structured P2P overlays use routing algorithms to locate node(s) in an overlay and retrieve data items from them. The routing algorithm defines how a target node is located in the overlay network. This lookup is closely associated with the geometries of the P2P overlay and the connectivity or information stored at each node.

DHT-based routing algorithms use the hash of a node ID to form a node ID space, which typically is uniformly distributed (however some overlays purposefully break this to achieve a closer relation between the underlying physical network and the overlay). A commonly used hashing function is SHA-1.

The identifier for data items (file name etc) is created by applying the same hashing function. Hence the node IDs and data IDs fall into the same ID space. Data items are typically stored on the closest node with node ID greater than or equal to the data ID. Using this approach each node can find a particular data item using its name. If the node with the closest node ID does not store the data item, it is not available in the network. Using this approach any existing data item can be found by any node in the overlay.

Based on these characteristics a number of different routing algorithms have been defined by various overlays. Major approaches are logarithmic routing (e.g. used by Chord [8]), One-hop routing (EpiChord [9]), XOR routing (e.g. used by Kademlia [10]), and the Content Addressable Network (CAN) [11].

Logarithmic routing means that it takes O (log N) steps to route a message from source to destination node. N is the maximum number of nodes in the overlay. Each node will route a message closer to the destination node by selecting the entry in its routing table whose node ID is closest but smaller or equal to the destination node ID. Logarithmic routing guarantees that with high probability that it does not take more than log N steps to reach a destination node. Overlays described in Section 3 belong to this category.

Clearly, the more accurate and the larger routing tables are, the fewer hops are required to route a message from source to destination. Constant degree overlays guarantee that routing from source to destination is achieved in a certain number of hops, independent of the size of the overlay. Overlays discussed in Section 4 belong to this category. This approach is pushed to the extreme in one-hop overlay networks which have almost complete routing tables in each node and hence can transmit messages in (almost) a single hop from source to destination. Overlay algorithms described in Section 5 belong to this category.

2.3 Join/Leave Mechanisms

In the previous sections, we discussed the P2P overlay geometies and the routing mechanisms. P2P systems are highly dynamic in nature. They need robust mechanisms for nodes to join or leave the system at any time with minimal impact to the functioning of the P2P overlay. However, the need for a geometry that leads to a deterministic routing behaviour and the need for autonomous nodes provide a dichotomy for P2P systems. Structured P2P systems use specific join and leave mechanisms for nodes to resolve this dichotomy. These mechanisms provide a balance between high dynamism of P2P systems and a predictable or deterministic P2P overlay behavior.

Peers join an overlay network by connecting themselves to any of the existing peers. But in structured P2P systems, a peer cannot randomly pick exiting peers to join. Instead, it must connect itself to well-defined peers based on its logical identifier and on the geometry of the P2P overlay. Because of this controlled manner, the join and leave mechanisms can greatly affect the performance of structured P2P systems.

Peer Join: Typically, there are three steps for a node to join a structured P2P overlay.

1. The first step is to get a unique identifier for the node. As discussed above, the hashing scheme is based on unique properties of the node. For example, MAC/IP addresses could be used to obtain such an identifier.
2. The second step is to position itself into the overlay structure. This positioning is based on the node's id and the geometry of the P2P overlay. During this step, the node needs to know the entry point or the identity of an existing node to insert itself into the overlay. This process is called bootstrapping, which is discussed later in this section.
3. Finally, in the third step, the new joining peer and all the affected peers update their routing tables to stabilize the overall P2P overlay. By stability we mean the predictable behaviour of the P2P overlay. This step is often referred to as routing table maintenance.

Routing table maintenance is discussed in the following sections, so we focus on the second step where a node, after finding an existing peer, inserts itself into an overlay. The joining node contacts a peer in the overlay to find out its appropriate position in the overlay. The existing peer uses techniques that are associated with the overlay geometry to find out the new node's neighbours. For example, in Chord, the existing node will issue a lookup to find the successor of the new node based on the new node's identifier. The new peer will then connect to its successor and join the overlay.

Once a new node joins a structured P2P system, the new node and affected peers need to update their properties, usually routing tables, to keep the invariants of the overall system. The number of peers being affected varies for different systems. For example, a new node in Chord affects O(logN) nodes, where N is the number of peers in the system. A new node in CAN affects O(d) nodes, where d is the dimension of the system. However, in O(1) systems, all other nodes in the overlay may need notifications. Clearly, the complexity is dependent on the particular overlay geometry.

Peer Leave: When a node leaves or becomes unreachable in a structured P2P overlay, nodes that point to that node are affected. Their routing table entries will be stale and have to be updated. A timely update results in preserving the invariant of the overlay and guarantees the deterministic lookup in the overlay.

A gracefully departing peer may notify its neighbours about its departure and transfer necessary information to its neighbours for updating their routing tables. Its neighbours then propagate the changes if needed until the invariants of the system are preserved. For example, in Chord, after the update, each node's successor should be correctly maintained and for every key k, node successor (k) should be responsible for k.

In some cases, a node may leave the system unexpectedly, e.g., due to network failure or power outage. Under these circumstances, the node will not notify its neighbours and cannot send necessary information for routing table updates. Hence the system must have a failure detection and stabilization mechanism. Failure detection is usually handled by heartbeat messages or periodic checking. For example, CAN nodes send periodic update messages to their neighbours. The prolonged absence of an update message from a neighbour signals its failure. Chord nodes periodically do random checking on its finger tables to detect failures. Once the failure of a node is detected, usually by its neighbours, the neighbours will start the stabilization process to update routing tables.

2.4 Routing Table Maintenance

As discussed in the previous sections, in structured P2P overlays, each node maintains a routing table to find other peers in the network. Routing table sizes depend heavily on the overlay geometry. In *Multi-hop overlays* (that is a lookup takes multiple overlay hops from source to destination) generally use smaller routing tables than one-hop overlays. In one-hop overlays, ideally every node is aware of every other node, hence routing tables need to include references to every other node in the system. This requirement results in large routing tables and poses an additional problem in maintaining the routing tables. Hence, the improved latency behaviour of one-hop overlays comes at a cost of increased maintenance traffic.

Routing table entries need updating if new nodes join or existing nodes leave the network. Usually a join and a graceful leave are propagated through the network by a defined algorithm. However, ungraceful leave events are harder to detect. Generally, two main approaches have been defined for keeping routing tables up-to-date: *opportunistic maintenance and active maintenance.*

With opportunistic maintenance, an overlay uses lookup messages and responses to distribute routing table entries. For example, a node will attach entries from its routing table to a response message. The receiver of this response can then augment its routing table with these nodes. This is efficient in terms of number of dedicated maintenance messages required, however, the accuracy of the routing tables is dependent on the number of lookup messages sent. During periods of high churn there is an increased demand for routing table updates. During such periods, opportunistic maintenance may not be sufficient on its own and hence nodes may insert additional lookup messages to receive more routing table updates. As an example, EpiChord [9] discussed in Section 5.3 employs opportunistic routing table maintenance.

With active maintenance, there is a specific algorithm and dedicated messages to propagate routing table entries between nodes. Typically these messages are distributed when a node-join or node-leave event is detected. Usually, neighbouring nodes pick up these events and then distribute these events to all the other nodes in the overlay. Alternatively, update requests are sent after a time interval has expired. Clearly, active maintenance requires a higher bandwidth for distributing node join/leave events, but achieves better routing table accuracy than with opportunistic maintenance schemes. D1HT [46] as discussed in Section 5.4 uses active routing table maintenance.

2.5 Bootstrapping

Bootstrapping is a key operation in structured peer-to-peer overlay network. Bootstrapping operation is executed when a peer joins the P2P system for the first time. It enables the initial discovery of other nodes/peers participating in the P2P network. Nascent peers perform such an operation to join the P2P network. Bootstrapping include the period from peer arrival to the point when the nascent peer becomes a functioning peer of P2P overlay.

One common approach for bootstrapping is through a bootstrapping server. The bootstrapping server maintains a list of participating peers. When contacted by a nascent peer the bootstrapping server returns a partial list of existing peers. The nascent peer connects to the peers in the returned list to join the P2P network. The address of the bootstrap server is usually obtained out-of-band.

Another common approach for bootstrapping is to let nascent peers know in advance an entry point into the network. The entry point can be a list of known peers of a P2P overlay, or a list of non-public bootstrapping servers.

Once a nascent peer gets in touch with some existing peers in the network, it starts the joining process. Different P2P networks employ different strategies, and typically, the joining process is closely related to P2P overlay's routing strategy. The bootstrap server can look at the requesting node's hashed identity and can return the list of existing peers so that the joining process could be optimized. The key issue in designing a good bootstrapping strategy is how to support peers to connect into the network quickly.

3 Logarithmic Degree Overlays

In this section as well as Sections 4 (Constant Degree Overlays) and 5 ($O(1)$-hop overlays), we use the key mechanisms described in the previous section to describe various structured overlays. That is, for each overlay we describe its address space along with its geometry, its routing table and lookup algorithm, and its join and leave mechanisms. In terms of bootstrapping, not all overlays specify a particular approach, however, it appears that all overlays require that a new node knows about at least one peer in the overlay. Common approaches to find an overlay node are IP multicast, using a well-known bootstrap node or using the DNS. Here the overlay service is associated with a DNS domain name. IP addresses of one or more overlay bootstrap nodes are retrieved using the DNS lookup service.

3.1 Chord

Chord [8], developed by a group of researchers at MIT, is one of the first P2P overlay system based on distributed hashing table (DHT).

Address space: Chord uses the so-called consistent hashing to assign each node and each key an m-bit identifier (Id), where m is a pre-defined system parameter. Ids fall into the range from 0 to $2m-1$. Nodes are ordered on an identifier circle modulo $2m$, as shown in Fig. 2. A key is cached at its successor node, defined to be the next node in the identifier circle in the clockwise direction. The predecessor node to a node or key is the next node in the identifier circle in the counter-clockwise direction.

Routing table and key lookup: The routing table of Chord nodes consists of two parts. The first part includes a finger table with m entries, and the predecessor of this node. Assume the node Id is n. The i-th entry in the finger table, where , points to the node whose Id is the closest to $n+2i-1$ in the clock wise direction at identifier circle. Notice that the first entry in the finger table is the successor node of the current node. Predecessor node plus the finger table guarantees the correctness of key lookup service, as described below.

The second part of the routing table is a successor list of size r. In addition to the immediate successor node maintained in the finger table, other closest $(r-1)$ success nodes are also recorded. The successor list improves the robustness of Chord protocol, and allow Chord to perform correctly in the face of peer churn, i.e., dynamic peer arrivals and departures.

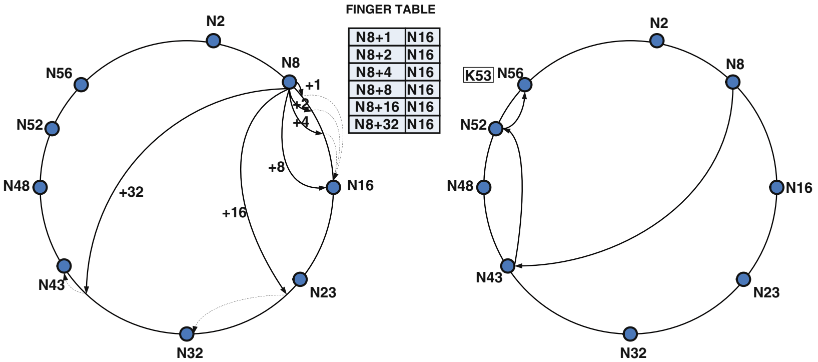
A key lookup request is routed along the identifier. Upon receiving a lookup request, the node first checks if the lookup key Id falls between this node's Id and its successor's Id. If it does then it, returns the successor node as the destination node and terminates the lookup service. On the other hand, if the lookup key Id does not belong to the current node, the node relays the lookup request to the node in its finger table with Id closest to, but preceding, the lookup key Id. The relaying process proceeds recursively (or iteratively) until the destination node is found. A key lookup example is depicted in Fig. 2. In the figure, the left-hand side shows the finger table of Node 8 (N8). Node 16 (N16) appears in four entries in the finger table, while Node 32 (N32) and Node 43 (N43) are also in the finger table. The right-hand side figure depicts the stages for the lookup of key 53 starting from Node 8. It has been shown that the number of routing steps is at the order of $O(\log N)$, where N is the total number of Chord nodes. Refer to [8] for more detailed treatment on Chord routing algorithm.

Node join and leave: The newly arrived node in Chord first uses consistent hashing to generate its Id. It then contacts the bootstrapping node, the node already in the Chord, to lookup the successor of its Id. This successor node becomes new node's successor node. The new node uses the stabilization protocol, which is described below, to have a fully correct routing table.

Stabilization protocol is designed to maintain routing tables' correctness in the face of peer churns. It is executed periodically at the background of individual nodes. The stabilization protocol includes following two major functions:

Stabilize( ): allows nodes to learn about newly joined nodes and to update their successor(s) and predecessor.

Fix_fingers( ): ensures finger tables are current and correct.



N8+1	N16
N8+2	N16
N8+4	N16
N8+8	N16
N8+16	N16
N8+32	N16

Fig. 2 Example scenario for key lookup in Chord

Node failure/departure creates another challenge to the Chord protocol. The departure of a node leaves its predecessor node's successor pointer invalid, which could affect the routing correctness. To address this issue, Chord maintains a successor list of size r. The successor list can be stabilized using slightly changed stabilization protocol. It's proven that with size $r = S(logN)$, where N is the total number of nodes in Chord, the lookup can still succeed with high probability even if every node fails with probability of 1/2. A study of Chord's behavior under churn can be found in [12].

3.2 Pastry

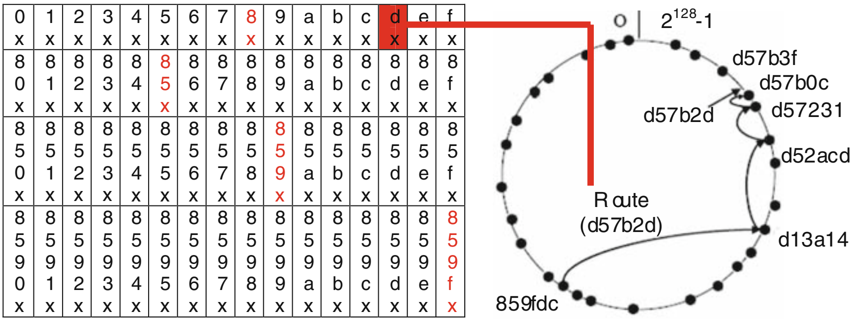
Pastry [13–17] is developed by researches from Microsoft Labs Research, Rice University, Purdue University, and University of Washington. There are several applications built on Pastry for different purposes, such as SCRIBE [18–21] for group communication/event notification, PAST [22, 23] for archival storage, SQUIRREL [24] for co-operative web caching, SplitStream [25, 26] for high-bandwidth content distribution, POST [27] for co-operative messaging, and Scrivener [28] for fair sharing of resources. Two implementations of Pastry are available for download: FreePastry [29] from Rice University and SimPastry and VisPastry [30] from Microsoft Research.

Address space: Each Pastry node has a unique, 128-bit nodeId. Node IDs are chosen randomly and uniformly. One way of generating nodeIds is by hashing nodes' IP addresses.

Routing table and key lookup: Pastry uses prefix matching to route messages. Each Pastry node keeps a routing table with $\lceil \log_2^b N \rceil$ rows and $2^b - 1$ columns. The entries in row n share the first n digits with the present node. In addition to the routing table, each node also maintains a leaf set that contains the IP addresses of nodes with $l/2$ numerically closest larger nodeIds, and $1/2$ nodes with numerically closest smaller nodeIds, relative to the present node's nodeId.

Given a message with its key, the node first checks its leaf set. If there is a node whose nodeId is closest to the key, the message is forwarded directly to the node. If the key is not covered by the leaf set, then the node checks the routing table and the message is forwarded to a node that shares a common prefix with the key by at least one more digit. This way, with $\lceil \log 2^b N \rceil$ steps, the message can reach its destination node.

Figure 3 shows an example lookup scenario. The left-hand side table shows the routing table and the right-hand diagram shows the route. The node 859fdc looks up a key d57b2d. From its routing table, it gets d13a14, which shares one digit common prefix with the key. d13a14 then checks its routing table and get d52acd, which shares two digit common prefix with the key. This step keeps on until the key is covered by the node d57b0c.



0 x	1 x	2 x	3 x	4 x	5 x	6 x	7 x	8 x	9 x	a x	b x	c x	d x	e x	f x
8 0 x	8 1 x	8 2 x	8 3 x	8 4 x	8 5 x	8 6 x	8 7 x	8 8 x	8 9 x	8 a x	8 b x	8 c x	8 d x	8 e x	8 f x
8 5 0 x	8 5 1 x	8 5 2 x	8 5 3 x	8 5 4 x	8 5 5 x	8 5 6 x	8 5 7 x	8 5 8 x	8 5 9 x	8 5 a x	8 5 b x	8 5 c x	8 5 d x	8 5 e x	8 5 f x
8 5 9 0 x	8 5 9 1 x	8 5 9 2 x	8 5 9 3 x	8 5 9 4 x	8 5 9 5 x	8 5 9 6 x	8 5 9 7 x	8 5 9 8 x	8 5 9 9 x	8 5 9 a x	8 5 9 b x	8 5 9 c x	8 5 9 d x	8 5 9 e x	8 5 9 f x

Fig. 3 Example lookup scenario in Pastry

Node join and leave: In order to join a Pastry network, a new node must know an existing node. The new node can initialize its state by contacting the existing node by sending a *join* message with its nodeId as the key. The message is routed to another existing node with nodeId numerically closest to new node's nodeId. Then all nodes encountered on the routing path send their state tables to X. The new node X then initializes its own state tables based on the new information. Finally, the new node informs any nodes that need to be aware of its arrival. Routing table maintenance is handled by periodically exchanging keep-alive messages among neighbor nodes. Upon detecting node failure, all members of the failed node's leaf set are then notified and they update their leaf sets.

3.3 Kademlia

The basic principle of Kademlia [10] is to successively find nodes that are half the distance to the target node. Kademlia differs from Pasty and other such overlays in mainly two different aspects. One difference is a new notion of node closeness based on XOR of the node identities. The other difference is Kademlia nodes contain lists of entries, referred to as buckets, which are used to send parallel requests.

Address Space: Kademlia system assigns 160-bit node IDs. The lookup algorithm uses a XOR-based closeness to reduce the lookup space. The intuition behind the XOR based closeness is that node IDs that are different at higher order bits matter more than node IDs that are different in lower order bits and hence, the XOR distance would be higher. Using this XOR metric, Kademlia's topology orders nodes as a tree where subtree nodes are closer together than other subtrees.

Routing table and key lookup: Routing tables contain separate lists for each bit in the node ID. Hence if a network uses 128 bits for node IDs each node will have 128 lists (called buckets in Kademlia). Each list corresponds to a particular distance to nodes. Distance is measured in matching bits in the node IDs. Nodes in the nth list have a differing nth bit from the current node's ID whereas the first $n-1$ bits match those of the current node's ID. To define distance between nodes, Kademlia uses XOR metrics. Here the result of the XOR operation applied to two node IDs (returning 0 for identical bits and 1 for differing bits) is the distance between two nodes. Like Chord, Kademlia nodes know about more nodes near to them and fewer nodes further away.

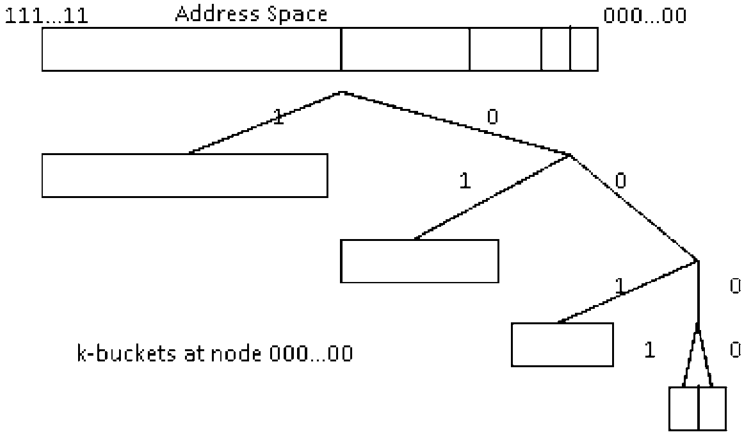


Fig. 4 Kademlia routing table data source

Figure 4 shows a routing table for a node with ID 000..00. Note that there are k-buckets, each of which covers an address space based on the XOR metric of node IDs. Each of these buckets is a list that may contain multiple contacts for a given subtree. Maintenance of these buckets is straightforward though highly unbalanced trees are handled separately. Maintenance of the nodes in the lists could be dependent on the applications.

A Kademlia lookup node first finds the *k*-closest nodes to the given node ID. Kademlia recursively picks a subset, *l*, of these *k*-closest nodes and sends a request to all the *l* nodes. In the next recursive step, the Kademlia lookup node again picks a subset of nodes from the nodes it learned about from the previous request. Intuitively, each of these recursive reduces the XOR metric distance by 1/2 and results in the smaller size *k*-buckets. The concurrent lookup provides a trade off between bandwidth and lookup latency.

Node join and leave: Node joins mirror node lookups. That is, a node, *u*, that wishes to join adds a previously known contact, *w*, to its bucket and performs a node lookup. It fills up its routing table based on the responses and inserts itself into the *k*-buckets of the other nodes in the system. There is no specific mechanism for node departures as other nodes may discover through the PING mechanism.

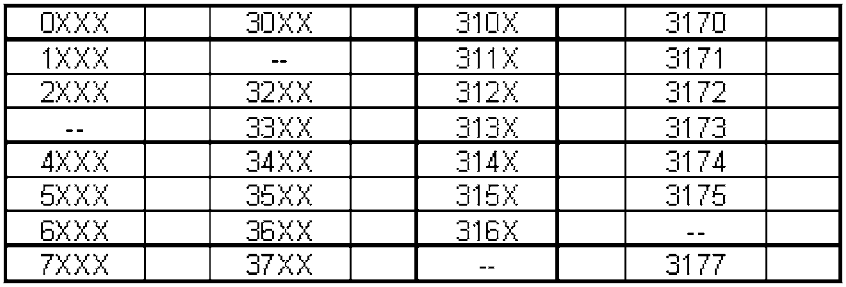
3.4 Tapestry

Tapestry [31] was developed by a team of researchers from University of California, Berkeley, and MIT. Tapestry has close links with Pastry in that both offer a prefix-based routing of messages. Tapestry aims at providing high-performance, scalable and location-independent routing of messages to near-by endpoints. Tapestry exploits locality when routing messages, including object replicas. Especially, Tapestry allows applications to place object replicas according to the application's need. Bayeux [32] as a Application Lay Multicast approach has been implemented on Tapestry. Chimera [33] is a more recent and updated Java-based implementation which uses Tapestry concepts.

Address Space: Tapestry nodes are assigned node IDs uniformly at random from a large identifier space. Typically a 160 bit values are used together with a globally defined radix. Usually the radix is defined as hexadecimal resulting in 40-digit identifiers. The SHA-1 hashing algorithm may be used to create node IDs. Data items (or Application specific endpoints) are assigned unique identifiers from the same ID space.

Routing table and key lookup: Each node maintains a routing table whose entries consists of node IDs and the corresponding IP addresses. All nodes represented in a routing table are called 'neighbours' of that node. Routing corresponds to forwarding messages across neighbour links to nodes which are closer, i.e. matching more digits of the prefix, to the key of the endpoint. An example routing table is shown in Fig. 1. This routing table belongs to node 3176 in an overlay which uses 4-digit octal IDs. Each routing tables has a number of levels corresponding to the number of digits used in the IDs. For the example shown in Fig. 5 this corresponds to 4 levels.

Each level contains links to nodes matching a prefix up to a digit position in the ID. Each level contains a number of entries equal to the radix used (in Figs. 5, 8 as octal). Further, the primary ith entry in the jth level corresponds to the closest node whose ID begins with the corresponding prefix. Using this 'closest node' approach provides the locality properties of Tapestry.



0XXX		30XX		310X		3170	
1XXX		--		311X		3171	
2XXX		32XX		312X		3172	
--		33XX		313X		3173	
4XXX		34XX		314X		3174	
5XXX		35XX		315X		3175	
6XXX		36XX		316X		--	
7XXX		37XX		--		3177	

Fig. 5 Routing table example for Node ID 3176

Each hop in the routing of a message takes the message closer to its destination. Specifically, the node for the nth hop shares a prefix of at least *n* digits with the destination ID. This approach guarantees that any node in the system can be reached in at most log $\cdot N$ overlay hops, where N is the size of the namespace and $\cdot$ is the radix used.

Node Join and Leave: Inserting a new node N starts at the node that is responsible for the ID of N in the overlay. This surrogate node S determines *p*, the number of digits its ID shares with N's ID. S then sends a multicast message to all nodes which share the same prefix. These nodes will add N to their routing table and in turn contact N, so N can add these nodes to its own routing table. N then carries out an iterative nearest neighbour search starting at level *p*. N may trim the list to the closest *k* nodes. N then requests these *p* nodes to send their backpointers at that level. This results in a set of all nodes that point to any of the *k* nodes at the previous routing level. Next N decrements *p* and repeats the process for all remaining levels.

If a node N decides to leave the network, it notifies all nodes in N's backpointers about it leaving. With each notification N provides a replacement node from its own routing table. Any object references stored on N are rooted to their new hosts. Nodes that left the network ungracefully are detected using periodic beacons. Such leaving events trigger repair of the overlay and initiate redistribution of object references.

3.5 P-Grid

P-Grid [34–36] uses a virtual binary tree to form an overlay. The virtual tree is used to distribute the data items to be stored in the overlay to one or more peers. P-grid achieves O(log n) performance for search operations where n is the number of data items in the overlay.

Address Space: P-Grid is similar to Kademlia in that it also uses tree-based routing. However, in P-Grid the node IDs are disentangled from the key IDs. In fact, there is no requirement to hash node IDs in P-Grid. Due to the nature of the construction of key IDs, P-Grid supports substring queries. This is probably one of the most distinguishing features of P-Grid when compared with other DHT based overlays. Like some other DHTs Pastry, Tapestry), P-Grid uses a prefixed based search algorithm.

Routing table and key lookup: Each peer in P-Grid contains a routing table which contains an entry for every bit in the binary tree. The example shown in Fig. 6 uses a 3-bit binary tree. Hence each routing table has 3 entries. Each entry corresponds to a bit in the path towards that node and stores at least one peer that is responsible for the other side of the binary tree at that level. As an example Node 6 is linked to 011 in the binary tree. Hence it has entries for the other side of the tree at the topmost level, in this case 1 (Node 5), the second level, in this case 00 (Node 4), and the bottom level of the tree, here 010 (Node 2).

The binary search tree can be constructed for any set of strings. To construct a tree a sample search string database is used. Firstly, the length of the common prefix of the strings in the database is calculated. This database is lexicographically sorted, and the string at the middle position in the sorted database is selected. The prefix of this string (length is the common prefix + 1) is determined and used to split the database in two equally sized parts. The prefix is then stored at the root of the tree. This splitting proceeds until the desired depth of the tree is achieved. The binary key of a string is then calculated by comparing the string to the tree's root value. If the string is smaller than this then 0 is appended to the key and the left subtree is considered next, and if it is greater then 1 is appended to the key and the right subtree is considered next. This algorithm is carried out for every level in the tree.

In P-Grid a number of nodes might be responsible for the same part of the tree. For instance in Fig. 6, Nodes 1 and 6 are responsible for data with the prefix 000, Nodes 9 and 2 are responsible for data with the prefix 010, Nodes 13, 20, and 5 are responsible for data with the prefix 101 and Nodes 11 and 15 are responsible for data with the prefix 110. If a peer receives a query it cannot directly satisfy it will forward the query to a node closer to the destination. For instance, in Fig. 6, if Node 4 receives a query for 101, it will forward the query to Node 3 as the query starts with 1. Node 3 will forward the query to Node 5 which is responsible for this data and will return it to the original requesting node.

Node Join and Leave: P-Grid construction is carried out by local interactions between peers only. It is assumed that by some mechanism peers will meet and interact. Initially, all peers are responsible for the entire search space, i.e. all keys. As two peers meet, they divide the search space into two halves, with each node taking responsibility for one half. This approach is carried out whenever two peers meet which have responsibility for the same address space. If peers meet, which are responsible for data items whose keys have a common prefix, they can initiate

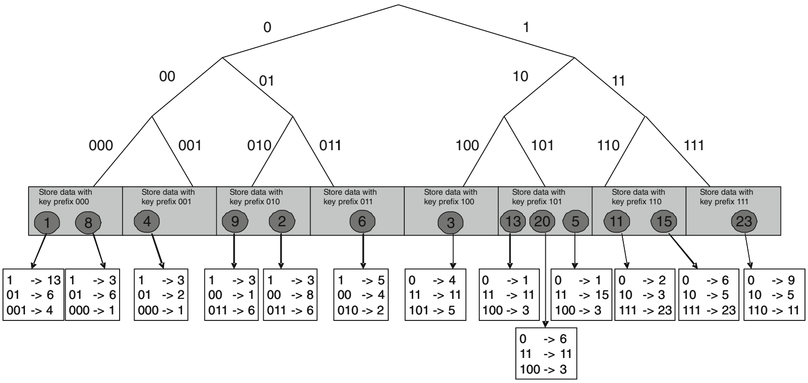


Fig. 6 Example P-Grid with a 3-bit binary tree

additional meetings by forwarding each other to peers in their respective routing tables. If the meeting peers have a different path length the peer with the shorter path can specialise by extending its path in the opposite direction from the other peer at that level. This algorithm is uniform and self-stabilising.

4 Constant Degree Overlays

4.1 CAN

CAN, or Content Addressable Network [11], is one of the first DHTs (distributed hashing table) proposed.

Address Space: CAN utilizes a virtual d-dimensional Cartesian coordinate space to host both keys and nodes. Keys and nodes are mapped to corresponding points/-coordinates in this d-dimensional space using a uniform hash function. Hence the address of a node is its location within the logical coordinate system. As with all DHT based algorithms, the location of a node is calculated using a hash function. Then, the entire coordinate space is divided into "zones" where each node owns one zone. Node that owns the zone is responsible for the keys sitting in the same zone.

Routing table and key lookup: A CAN node's routing table contains coordinates and IP addresses of each of its neighbours in the coordinate space. Neighboring nodes are the nodes whose zones adjoin each other. In a d-dimensional space, two zones adjoin if their coordinate spans overlap along $d-1$ dimensions.

The key lookup starts with hashing the search key to a coordinate in d-dimensional space. A CAN message is then formed carrying the destination coordinates. The lookup message is forwarded toward the destination zone in a greedy fashion: the node always forward the message to its neighbor node that is closer to the destination. A tie is broken arbitrarily. Figure 7 illustrates such an example in a 2-dimension CAN space.

For a d-dimensional CAN space with n nodes, the average routing path length is $(d/4)(n^{1/d})$ hops. The routing table size is $2d$. Hence the routing table size is independent of number of available CAN nodes, and the routing path length grows in the order of $O(n^{1/d})$.

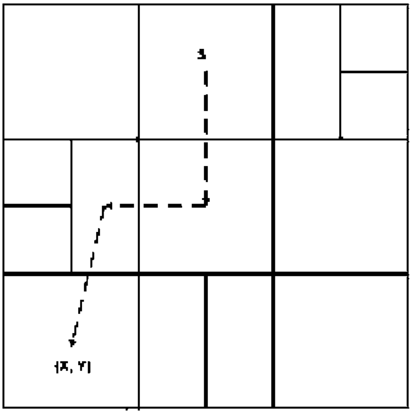


Fig. 7 A 2-dimension CAN space with 14 zones. Node 3 initiates a lookup for key (X,Y) using greedy algorithm

The routing table in CAN is maintained through periodic update messages. As long as the routing table contains the right neighbors, the lookup service will succeed. In case a node loses multiple entries in the routing table simultaneously, or the rebuilding process has not fully recovered the routing table, a node may use stateless, controlled flooding to locate a node closer to the destination. The closer node then takes over the lookup and uses greedy forwarding thereafter.

Node join and leave: A newly arrived node knows at least one existing node in CAN. It randomly generates its coordinate, *P*, in the virtual space. A JOIN request with destination *P* is sent through the known CAN node. Once the destination CAN node receives the JOIN request, it will split its zone into half and assign one half to the new node.

The new node builds up its routing table through learning previous owner node's routing table. The routing table comprises of the subset of retrieved routing table plus the occupant node as neighbor. The routing tables of neighboring nodes also need to be updated. The new and previous occupant nodes send out update messages to neighboring nodes. In fact, the zone information of a node is periodically sent to a node's neighbors, which ensures the correctness of routing table.

When a node leaves the system gracefully, its zone and associated (key, value) database is handed over to one of it neighbors. The new zone owner merges the handed over zone with its original zone.

In case a node fails or departs unexpectedly, the periodical update message will discover such departure/failure. The takeover mechanism is automatically trigged at nodes that discover the failure. The takeover timer is started. When the takeover timer expires, the node sends a TAKEOVER message with its own zone information to all of the failed node's neighbors. The receiving node either cancels its own timer if the zone volume in the message is smaller than its own zone volume, or it replies with its own TAKEOVER message. The goal is to have the neighbor node that has the smallest zone to take over the orphan zone.

4.2 Ulysses

Ulysses [37] can achieve tries to achieve $\log_2 \log_2 n$ end-to-end routing latency with a routing table size of about $\log(n)$.

Address space: The structure of Ulysses is based on the butterfly topology [38], but it improved the static butterfly topology by accommodating the dynamics of peer-to-peer networks. In addition, it solves the problem of high edge stress of static butterfly topology by adding shortcut links.

Figure 8 shows a Ulysses network with 2 levels and 11 nodes. In a Ulysses network, a node is identified by a tuple (P,l), where l is the level number and P is a binary string uniquely identifying the node in the level. P can be mapped to a k-dimensional row identifier $(x_0,x_1,\ldots,x_{k-1})$ in a static butterfly is as follows : The bits at location i, $i+k$, $i+2k,\ldots$ in P represent x_i in $(x_0,x_1,\ldots,x_{k-1})$. The length of P in a Ulysses network with n nodes and k levels is expected to be $log_2(n/k)$. But the length of P for individual nodes changes due to dynamic arrival and departure of nodes.

Routing table and key lookup: In a Ulysses network with k levels, a query for the key (a,i) originates at a random node. The query keeps getting forwarded to next level. In each forwarding step, the forwarded node can match one additional dimension of the key. After the k steps the query reaches a node (Q,l) such that a lies within the zone Q in all the k dimensions. If the level l is the same as the level i of the key that is being searched, then Q must contain a, and the routing is complete. Otherwise the node (Q,l) forwards the query on its shortcut link to node (Q,i), which must be responsible for the key (a,i).

Node join and departure: A new node must know an existing node to join a Ulysses network. It then generates a random key and sends a query for this key through the existing node. This query will eventually reach the node $O(Q,l)$ responsible for the key. Node O then splits its zone of responsibility in two and assigns one half to the new node. The identifiers of node O and the new node will then be $Q0$ and $Q1$, respectively. Both nodes remain in level l. The node O informs the new node N

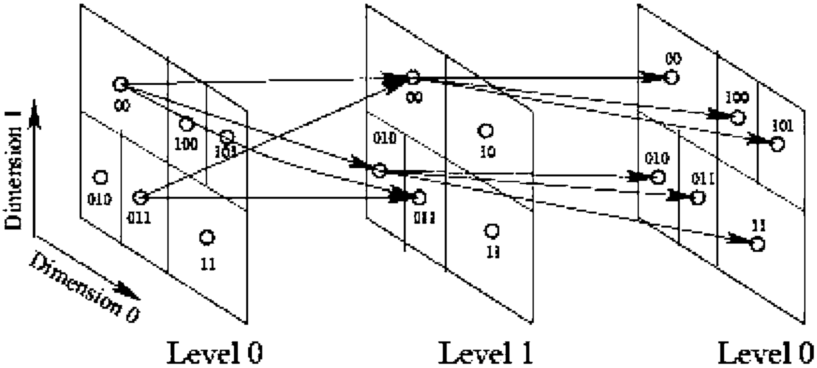


Fig. 8 A Ulysses butterfly with 2 levels and 11 nodes

about its original neighbors. When a node with identifier (P, l) leaves the network, it needs to hand over its keys to another node at the same level.

4.3 Cycloid

Cycloid [39] presents an overlay that combines hypercube routing with overlay routing that reduces the lookup path by key matching as in Pastry. Cycloid uses Cube-Connected-Cycles graph as its geometry. The geometry and the routing algorithm ensure that the lookup time is O(d), where ‘d’ is the network dimension.

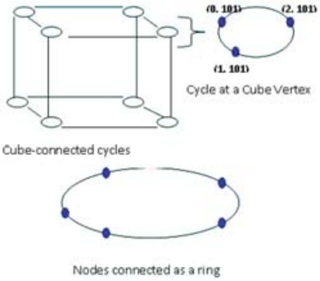


Fig. 9 Various linkages of a full 3-dimensional cycloid

Address Space: A cycloid can be viewed as a d-dimensional cube where each vertex is replaced by a cycle of d-nodes, with $n = d.2^d$, where n is the number of nodes in the network. The connectivity of each node is constant and the finger table size (discussed later) is constant. Each node in a cycloid is represented as a tuple $(k, x_{d-1}, x_{d-2}, \ldots, x_0)$ with k as the cyclic index and $x_{d-1}, x_{d-2}, \ldots, x_0$ as the cubical index. The cyclical index is an index between 0 to $d-1$ and the cubical index is binary number between 0 and 2_{d-1}.

Figure 9 shows various linkages of a 3-dimensional cycloid. The cube shows how cycles at different levels are connected. Each cycle connects nodes that have the same cubical index but with different cyclical index. Finally, nodes with different indices are connected as a large ring that enables nodes on cycle to reach nodes in another cycle directly or indirectly.

To minimize the finger table size and maintenance, each node in a ring is connected only to a primary node with highest cyclical index in its preceding cycle and succeeding cycle. The predecessor and the successor of a node, say $(k, x_{d-1}, x_{d-2}, \ldots, x_0)$, in such a ring are chosen such that the most significant different bit (MSDB) with the current node is no larger than $k-1$. The predecessor and successor are chosen such that they are the first such largest and first such smallest nodes. Note that this arrangement of nodes with different cubic indices as a ring gives cycloid a lookup ability to select a cubical index that is closest to its destination from different cycles. Within each cycle, a node is connected to its predecessor and successor nodes. Hence the finger table of a node has seven entries and has the following entries.

	Node ID $(\mathbf{k}, x_{d-1}, x_{d-2}, \ldots, x_0)$
1	Cubical Neighbor – $(\mathbf{k-1}, x_{d-1}, x_{d-2}, \ldots, x_k, x, x, x, x)$
2	Cyclic Neighbor – node at k − 1 with max cyclical index less than $x_{d-1}, x_{d-2}, \ldots, x_k, x, x, x, x$
3	Cyclic Neighbor – node at k − 1 with min cyclical index greater than $x_{d-1}, x_{d-2}, \ldots, x_k, x, x, x, x$
4	Inside leaf set predecessor
5	Inside leaf set successor
6	Outside leaf set predecessor – primary node of preceding cycle
7	Outside leaf set successor – primary node of succeeding cycle

Table 1 Routing table of a cycloid node

Cycloid key assignment consists of generating a pair of cyclic and cubic indices. For a give key, the cyclic index is its hashed value modulated by d and the cubic index is the hash value divided by d.

Routing table and key lookup: The routing algorithm of a cycloid DHT consists of three steps. The first step uses the outside ring or the outside leaf sets of the finger table to find out the closest cubical neighbour or the closest cyclical neighbour. Then the inside leaf sets are used to find appropriate node. The following steps are performed by a source node $(k, x_{d-1}, x_{d-2}, \ldots, x_0)$ to route to a destination node $(l, y_{d-1}, y_{d-2}, \ldots y_0)$. MSDB represents the most significant different bit between the source and the destination nodes.

1. **Ascending:** If $k <$MSDB then it forwards request to a node in the outside leaf set until $k >=$MSDB. This step helps in either finding the closest cubical neighbour or closest cyclical neighbour.
2. **Descending:** If $k =$MSDB, then request is forwarded to the inside cubical neighbour else if $k >$MSDB then the request is forwarded to the closest cyclical neighbour.
3. **Traverse Cycle:** If the target ID is within the inner leaf set, then inside leaf set entries from the finger table are used for lookup.

Node Join and Leave: A joining node X will route the joining message through a bootstrapping node to a node Y whose ID is numerically closest to X. The finger table of X forms its leaf sets based on the finger table of Y.

1. If X and Y are in the same inside leaf set then the outer leaf set values of X are the same as Y. The inside leaf set values of X and Y are modified according to the position of X with respect to Y and others in the inner leaf set.
2. If X is the first in its cycle, then it has to form the links to the outside leaf sets. If Y's cycle is the succeeding remote cycle of X, then Y's left outside leaf node and primary node are the left and right nodes in X's outside leaf set. Otherwise the right outside leaf and the primary node are used. Since X is the only node in its cycle, its inside leaf sets point to itself.

Once a node joins, it propagates the join information to its entries which update themselves. Inner leaf sets update themselves while the outer leaf sets update themselves and propagate the join to their inner leaf sets.

When a node is leaving, it notifies inside leaf set nodes. If it is a primary node then the leaving node has to update its outside leaf sets. Upon receiving such message, the outside leaf set nodes update themselves and transfer the leave notification to its inside leaf set. The cycloid DHT leaves the updating of cubical and cyclical neighbours of leaving nodes and of failed nodes as the responsibility of system stabilization.

5 O(1)-Hop Overlays

5.1 Kelips

Kelips [40] is based on a DHT overlay that uses increased memory and increased background overhead for efficient O(1) lookup. Kelips overlay is simple and dif fers from other structured P2P overlays in mainly two ways. One difference is that it is loosely structured and the other is that the loose structure does not preserve any invariant. However, Kelips uses increased memory and sophisticated broadcast mechanism to achieve a reliable O(1) lookup. Hence, though Kelips lookup, join, and leave mechanisms are quite simple they rely on sophisticated broadcast mechanisms among its members.

Address Space: Kelips consists of k virtual affinity groups that are formed by hashing a node's identifier. Each node's finger table consists of a set of other nodes in its affinity group, a set of nodes in all the foreign groups, and a set of file tuples that give details of a file and the id of the node storing the file. Hence the total storage requirements of Kelips could be of the order $n/k + cX(k-1) + F/n$, where n is the number of nodes, k is the number of affinity groups, c is the contact size, and F is the file size.

There is no geometry to the address space and each node knows about a larger set of nodes. Kelips relies on a gossip-style epidemic [41, 42] protocol to ensure that with high probability the finger table information is transmitted to all nodes. To ensure this, nodes in Kelips overlay use a light weight protocol to transmit limited information, such as keep alive packets and filetuple information, to nodes in their affinity group and contacts group. These nodes in turn chose other nodes from their finger tables to propagate such information. There are studies that show that such a gossip protocol is quite robust against packet losses and node failures [43, 44].

Routing table and key lookup: A querying node maps a file name to appropriate affinity group and sends a look up request to the topologically closest node in that affinity group. The receiving node looks up its table and returns to the querying node the filetuple with the address of the homenode storing the file. The querying node then requests the homenode directly for the file.

Nodes that wish to insert a file follow the same procedure. They send the request to the topologically closest node from the hashed affinity group. The receiving node randomly picks one node from its affinity group and assigns the file to it and designates it as the homenode. The homenode then inserts the file, creates a new filetuple and inserts that information into the gossip stream.

Node Join and Leave: A bootstrapping node allows the joining node to create a soft finger table and allows it to join the gossip stream. Since there is no structure or invariant that Kelips has to preserve, join is complete with the node participating in the gossip stream. Node leaving or failures are updated through out the system through the gossip mechanism. If other nodes notice the lack of updates from the failed node, they update their entries accordingly.

5.2 OneHop

OneHop [45] was developed by a team at MIT. Some members were also involved in the development of Chord and EpiChord. So in many aspects OneHop relates to these two overlay algorithms.

Address Space: OneHop nodes are assigned a 128-bit random node ID. These IDs are ordered in a ring modulo 2^{128}. Identifiers are uniformly distributed as can be achieved by using hash functions such as SHA-1. Like in Chord, every node has a predecessor and a successor in the identifier ring. Each node periodically sends keep-alive messages to its predecessor and successor. Data items are assigned an ID in the same ID space. The node which should store a data is the successor, i.e. the first node in the ring clock-wise from the key.

Routing table and key lookup: In OneHop, every node maintains a full routing table, that is a routing tabe contains references to all other nodes in the network. This is to allow sending a lookup request from the source node to the destination node in a single hop. The source node will look up the successor node of the data key is requires, and send a lookup message to this node. Key to this scheme is that the routing tables are up-to-date. Hence node leave and join events need to be propagated to all nodes.

Node Join and Leave: Join and leave events need to be send to local nodes, but also to all the other nodes in the overlay. Local updates include updates to successor and predecessor nodes. Every node *n* runs a stabilisation routine periodically, which involves sending keep-alive messages to its successor *s* and predecessor *p* nodes. Node *s* checks if *n* is indeed its predecessor. If not, it informs *n* that there is another node between them. Similarly, *p* checks if *n* is its successor, and if not it notifies *n*. If either *s* or *p* do not respond, *n* will ping them repeatedly and after a timeout interval conclude that the node is dead.

A joining node contacts any other node in the overlay and gets its routing table, similarly to the approach in Chord. With this information the new node can establish its successor and predecessor and inform them of its existence.

Both join and leave events also need to be forwarded to all the other node in the system within a certain time. This is achieved by imposing a hierarchy on the system, forming a tree. This hierarchy is introduced by dividing the 128-bit identifier space into k equal contiguous intervals (slices). All slices will have roughly the same number of nodes as nodes have uniformly distributed random identifiers. Each slice has a slice leader which is the successor of the mid-point of the slice identifier space. New nodes learn about their slice leader from one of its neighbours. Slices are again divided into equal-sized intervals (units). Each unit has also a unit leader which is the successor of the mid-point of the unit identifier space. As a node detects a change in the membership of the overlay it informs its slice leader. The slice leader aggregates notifications from its members for a certain interval before sending them out to other slice leaders. The slice leaders again aggregate notification messages for a certain interval before sending them on to the unit leaders within their slice. Unit leaders piggy back these update information on keep-alive messages to their successor and predecessor nodes. Other nodes propagate this information also via keep-alive messages – if they receive the information from their predecessor they forward it on to their successor and vice versa, but not beyond unit boundaries preventing duplicate messages. If a slice leader fails, this will be detected by its successor, and this node will become the new slice leader.

5.3 EpiChord

EpiChord [9] is a variation of Chord that intends to speed up the lookup process. The speedup is achieved using parallel queries and by allowing nodes to cache more routing entries than $O(\log N)$ entries in original Chord protocol. EpiChord is able to achieve $O(1)$-hop lookup performance under lookup-intensive workloads, and at least $O(\log N)$-hop lookup performance under churn-intensive workloads in the worst case.

Same as in Chord, the nodes and keys are mapped to m-bit identifiers using consistent hashing. Nodes are ordered on an identifier circle modulo 2^m. A key is cached at its successor node. The successor node to a node or key is the next node in the identifier circle in the clockwise direction; while the predecessor node to a node or key is the next node in the identifier circle in the counter-clockwise direction.

Routing table and key lookup: To guarantee the routing correctness in the face of peer churn, each EpiChord node maintains a list of k successor nodes and k predecessor nodes, respectively. Furthermore, EpiChord divides the address space into two half circles, with each half circle being further divided into a set of exponentially smaller slices (see Fig. 10). It is required that at least $j/1-q$ entries are maintained in the routing table for individual slices, where j is predetermined number of entries per slice, and q is the probability that a entry is out-of-date. Since EpiChord node maintains k successor nodes and k predecessor nodes and both should fit in two smallest slices, the number of slices can be estimated. Further, the parameters j and k satisfy the following relationship, $k = 2j$.

The key lookup process utilizes p parallel queries. For a given key k, the node selects one node immediately succeeding k and $p-1$ nodes preceding k from its routing table. Upon receiving the search query, if the probed node owns the key, it responds as the destination node and the searching process finishes. If the probed node is not the destination node, it returns l best next hops from its routing table. In addition, the probed node returns its immediate successor if it is a predecessor of key k, or its immediate predecessor if it is a successor of key k. Both p and l are configuration parameters. When these replies are received by the original node, further queries are dispatched if returned nodes are closer to the key k than the nodes that have already responded. Notice that only original node issues queries and the lookup proceeds in an iterative fashion.

In EpiChord, each entry in the routing table is associated with a lifetime. Routing entries are flushed whenever lifetime expires. In addition, the routing entry associated with a node is purged if the node does not respond to some number of queries.

EpiChord nodes also monitor the number of entries available at each slice. Should a slice be found not to have sufficient routing entries, a node makes a

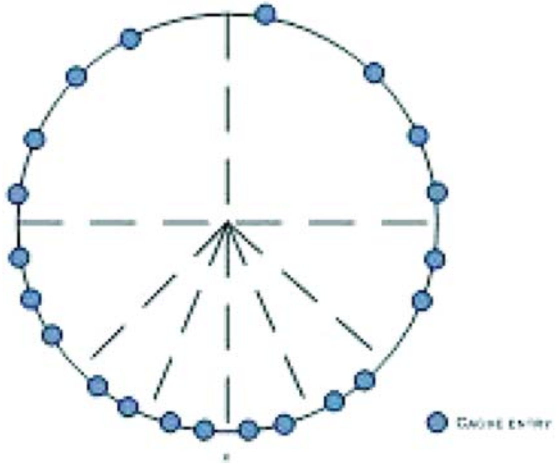


Fig. 10 Division of address space into exponentially smaller slices with respect to node x

lookup to the midpoint of that slice, and increases routing entries from the lookup responses.

Node join and leave: A new node knows at least one node already in EpiChord. It sends queries to this node. Since EpiChord nodes constantly update their routing tables by observing lookup traffic, other nodes learn about this node eventually. Besides, the new node obtains a full cache transfer from one of its two immediate neighbors.

EpiChord employs stabilization strategy to resolve the routing table inconsistency problem when multiple nodes join EpiChord at about the same location, or nodes leave the system unexpectedly. The stabilization strategy comprise of weak stabilization protocol and strong stabilization protocol.

Weak stabilization protocol: Weak stabilization protocol maintains weak stable relationship among nodes, i.e., predecessor(successor(n))=n. To achieve this, nodes periodically probe their immediate neighbors to check if they are alive. It is individual node's responsibility to maintain its successor and predecessor. When a node with closer Id than a node's successor or predecessor is discovered, either through observing lookup traffic, or through active probing neighbors, the node update its successor or predecessor, correspondingly.

Strong stabilization protocol: for a weak stable EpiChord ring, it is still possible the ring is loopy. Strong stabilization protocol ensures that the loop will be detected and fixed. The basic idea of loop detection is to let a query traverse the entire ring. If a loop exists, the traversal allows EpiChord nodes to know nodes whose Ids are closer than its successor or predecessor, thus break the loop by updating successor/predecessor nodes.

5.4 D1HT

D1HT [46] was first introduced in 2005 by researchers from Federal University of Rio de Janeiro, Brazil. The design goal of the overlay is to maximize performance with reasonable maintenance traffic overhead even for huge and dynamic peer-to-peer (P2P) systems. The philosophy of D1HT design is that the tradeoff between latency and bandwidth usage should favor latency because speed and information are critical while network bandwidth improves over time.

Address space: A D1HT system is composed of a set ***D*** of n peers and maps items (or keys) to peers based on consistent hashing, where both peers and keys are hashed to integer identifiers (IDs) in the same ID space $[0..N], N >> n$. Typically a key ID is the cryptographic hash SHA-1 of the key value, a peer ID is based on the SHA-1 hash of it's IP address (or the SHA-1 hash of the user name), and $N = 2^{160} - 1$. As in Chord, D1HT uses a ring topology where ID 0 succeeds ID N, and the successor and predecessor of an ID i are respectively the first living peers clockwise and counterclockwise from i in the ring.

Routing table and key lookup: Each peer in a D1HT system maintains a routing table with the IP addresses of all peers in the system, and so any lookup is trivially solved with just one hop, provided that the local routing table is up to date. As each peer in a D1HT system should know the IP address of every other peer, any join/leave events should be acknowledged by all peers in the system in a timely fashion in order to avoid stale entries in routing tables.

Node join and departure: A new node must know an existing D1HT node to join a D1HT system. The joining peer hashes its IP address (or some other unique value) to get its ID p and asks the existing node to issue a lookup for its ID. The query will return p's successor. The joining peer then contacts its successor to join the network and get the information about the keys it will be responsible for. The successor will also send the IP addresses of a number of peers to the new node. The new node will then ping those peers and choose the nearest ones to get the routing table. D1HT also uses a *Quarantine* mechanism to handle highly dynamic nodes, where a joining peer will not be granted to immediately take part of the D1HT overlay network, though it will be allowed to perform lookups at any moment. When a node leaves, instead of sending the leaving event to all the other peers, D1HT uses the EDRA (Event Detection and Reporting Algorithm) algorithm to propagate the event. The details of EDRA algorithm can be found at [46]. A study [47] found some shortcomings with the EDRA algorithm and termed the updated version EDRA*.

6 Comparison and Analysis

In this section we compare and contrast different structured P2P overlay technologies from the geometries used, the routing algorithms, routing performance, join/leave efficiency, routing table maintenance, and bootstrapping strategy. The results are summarized in the Tables below.

Table 2 Summary of Chord, Pastry and Kademlia overlays

	Chord	**Pastry**	**Kademlia**
Geometry	Circular Node-ID space, logarithmic degree mesh	Plaxton-style mesh network, prefix routing	XOR metric for distance between points in the key space
Routing algorithm	Search query forwarded to "closer" node	Matching Key and prefix of Node-ID	(XOR) Matching Key and Node-ID based routing done parallely
Routing performance	$O(\log N)$, where N is the number of peers	$O(\log BN)$, where N is number of peers, and $B = 2b$, b is number of bits of NodeID	$O(\log BN) + c$, Where N is number of peers, $B = 2b$,b is number of bits of Node-ID, and c is a small constant
Join/leave performance	$(\log N)^2$	$\log_B N$	$\log_B N + c$
Routing table maintenance	Periodic stabilization protocol at nodes to learn about newly joined nodes, update successor and predecessor, and fix finger tables	Neighboring nodes periodically exchange keep-alive messages. The leaf sets of nodes with adjacent Node-Id overlap	Failure of peers will not cause network-wide failure. Replicate data across multiple peers
Bootstrapping	A new node knows an existing Chord node	A new node knows a nearby Pastry node	A new node knows an existing Kademlia node

Table 3 Summary of Tapestry, P-Grid and CAN overlays

	Tapestry	***P*-Grid**	**CAN**
Geometry	Uniformly at random from a large identifier space (typically 160bit with a globally defined radix)	Binary tree	d-dimensional Cartesian coordinate space
Routing algorithm	Matching Key and prefix in Node-ID	Binary tree search and prefix matching	Forward the search message to neighbour node closer to the destination
Routing performance	$\log_\beta N$, where N is the size of the identifier space, and β is the radix used	$O(\log N)$, where N is number of data items in the overlay)	$(d/4)(N^{1/d})$, N is number of nodes, and d is dimension
Join/Leave performance		$O(\log N)$	$2d$
Routing table maintenance			Through periodic update messages. Controlled flooding is used in case a node loses multiple entries simultaneously
Bootstrapping	A new node knows a nearby Tapestry node	Know at least one node	Know at least one node. May get this node through DNS

Table 4 Summary of Ulysses, Cycloid and Kelips overlays

	Ulysses	**Cycloid**	**Kelips**
Geometry	Butterfly topology with shortcut links	Cube-Connected-Cycles graph	No geometry to the address space, each node knows about other nodes
Routing algorithm	For a Ulysses network with k levels and n nodes. to search a key a, in each step, the query gets locked in one additional dimension, after the first k steps the query reaches a node (Q,l) such that a lies within the zone Q in all the k dimensions	Uses the outside leaf sets of the finger table to find closest cubical neighbour or the closest cyclical neighbour. Inside leaf sets are used to find appropriate node. Routing table includes nodes in the affinity group, nodes in all the foreign groups, and filetuples	Querying node maps file name to affinity group and sends lookup request to topologically closest node in affinity group
Routing performance	$\log_2 \log_{2n}$	$O(d)$, where d is network dimension, n is number of nodes, $n = d.2d$	$O(1)$
Join/Leave performance	Find corresponding node with a randomly generated key, then split the zone. $\log 2 \log 2n$	$O(d)$	No structure or invariant. Join is complete with node participating in gossip stream. Node leaving are updated through the gossip system
Routing table maintenance		Leaving nodes notify nodes inside leaf set. If primary node then need also update outside leaf sets. Stabilization process detects failed nodes	Kelips routing tables are maintained through a low bandwidth gossip style mechanism
Bootstrapping	A joining node needs to know an existing node in Ulysses network	No specific bootstrapping mechanism discussed	Bootstrapping node allowing joining node to join gossip stream

Table 5 Summary of OneHop, EpiChord and D1HT overlays

	OneHop	**EpiChord**	**D1HT**
Geometry	128-bit random node ID ordered in a ring modulo 2128.	Circular node ID space, logarithmic degree mesh	Hashing keys and peers into an ID space $[0,N]$, $N >> n$
Routing algorithm	Every node maintains a full routing table	Use multiple parallel queries to locate node that owns the key	Every node maintains a full routing table
Routing performance	$O(1)$	$O(1)$ under lookup intensive workloads, and $O(\log N)$ in the worst case	$O(1)$
Join/Leave performance	$\lceil \log_{2n} \rceil$, where n is number of nodes		
Routing table maintenance	Nodes run stabilisation routine sending keep-alive messages to successor and predecessor	Routing entries are flushed whenever lifetime expires, or the corresponding node does not respond to queries. If slice entries are insufficient, lookup to midpoint of slice, add routing entries from the lookup response	Propagate join/leave messages with TTL values. Use a Quarantine mechanism to handle highly dynamic nodes
Bootstrapping	A new node knows an existing OneHop node	Knows at least one existing node	The new node must know an existing D1HT peer already in the system

7 Conclusions

7.1 Open Research Issues

We see the future research of structured P2P overlays focussing on the following three aspects, namely algorithms, frameworks, and applications.

1. **Algorithms:** As discussed in this chapter, many structured P2P algorithms exist. Each of these structured overlays carries its own advantages and disadvantages. The following are key issues that needs to be properly addressed for successful structured P2P algorithms:

    a. Scalability and performance: In general, the structured P2P overlays can help overcome the scalability and performance problems faced by unstructured ones. However, for some real-time applications there are additional performance requirements that need to be addressed, such as Voice over IP (VoIP) and IPTV. Therefore, how to ensure guaranteed performance required by certain applications while keeping the overall system scalable and balanced is an important issue for structured P2P algorithms.
    b. Security, privacy, trust and reputation: Trust and reputation are important to support secured and trustworthy P2P overlay communications among peers. There are many research topics in this area for P2P overlays such as anonymity, denial-of-service attacks, malicious node behavior, reputation and incentives.
    c. Convergence of Peer-to-Peer systems and other established field of distributed computing such as Grid computing.

2. **Frameworks:** Algorithms theoretically define P2P overlays. In practice, there are many practical issues to be dealt with:

    a. Protocols and interoperability: Peers need to talk to each other. In some scenarios, peers belonging to different P2P overlays may also need to talk to each other. This requires well-defined protocols/interface, and careful study of interoperability among P2P nodes.
    b. Heterogeneity: In reality, many aspects can affect the performance of P2P overlays, such as network availability/bandwidth, latency, peers' computational power and storage space, etc. Therefore, supporting heterogeneity is an important issue from a practical point of view.
    c. Handle general Internet services: general Internet services, such as spam handling and directory services, are also important to P2P overlays.

3. **Applications:** Many overlay algorithms and frameworks are developed are with the intention to build novel and useful applications. P2P applications can be in many fields. We just name a few below.

    a. Content sharing/distribution: this category of P2P applications may be the most popularly one so far. There is still plenty of ongoing research work in this area, such as providing better performance, good scalability, fairness, or strong security.

b. Enterprise applications: P2P applications in enterprise environment typically need to meet more stringent security and performance requirement. It also faces other issues such deployment easiness and monitoring capability.
c. Communication: For P2P communication applications, the real time constraint (both signaling and media transmission) requires special consideration in P2P algorithm and framework design. In addition, issues such as lawful interception, enabling communication features, and interop with the existing communication networks also demand special attention.
d. P2P applications in mobile and ad-hoc wireless networks: application of P2P overlay approaches would allow mobile peers to have optimized flow control, load balancing mechanism, and proximity awareness.
e. The semantic grouping of information in peer-to-peer networks: This direction shares many commonalities with efforts in the semantic Web domain.

7.2 Summary

Existing products and research projects demonstrate that structured P2P network is an important technology of practical value. It helps overcome the scalability and performance problems faced by many unstructured P2P technologies. This chapter provides an overview of several representative structured P2P overlays, and analyzes and examines key aspects that affect a P2P overlay's performance.

We believe that structured P2P overlay remains to be a viable solution to many problems in distributed computing. There are still many open research questions in this field, such as new algorithms, practical frameworks, and novel applications.

References

1. S. Androutsellis-Theotokis, D. Spinellis. A Survey of Content Distribution Technolgies. ACM Computing Surveys, Vol. 36, No. 4, December 2004.
2. E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. IEEE Communications Surveys and Tutorials, Second Quarter 2005, Volume 7, No. 2.
3. J. Risson, T. Moors. Survey of research towards robust peer-to-peer networks: search methods. Computer Networks 50, 17 (Dec. 2006), 3485–3521.
4. S. El-Ansary, S. Haridi. An Overview of Structured P2P Overlay Networks. Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks (ed. J. Wu). Auerbach Publications, 2006, pp. 665–683.
5. A. Gupta, B. Liskov, R. Rodrigues. Efficient routing for peer-to-peer overlays. Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI 2004), 2004, pp. 113–116.
6. J. Buford, A. Brown, M. Kolberg. Exploiting Parallelism in the Design of Peer-to-Peer Overlays. Journal of Computer Communications, Special Issue on Foundations of Peer-to-Peer Computing. 2008.

7. M. Kolberg, F. Kolberg, A. Brown, J. Buford. A Markov Model for the EpiChord Peer-to-Peer Overlay in an XCAST enabled Network. IEEE International Conference on Communications (ICC) 2007.
8. I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Trans. Netw. 11, 1 (Feb.2003), 17–32.
9. B. Leong, B. Liskov, E. D. Demaine. EpiChord: Parallelizing the Chord Lookup Algorithm with Reactive Routing State Management. Computer Communications, Elsevier Science, Vol. 29, pp. 1243–1259.
10. P. Maymounkov, D. Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In Proc of IPTPS02, Cambridge, USA, March 2002.
11. R. Sylvia, F. Paul, H. Mark, K. Richard, S. Scott. A scalable content-addressable network, Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, p.161–172, August 2001, San Diego, California, United States.
12. S. Krishnamurthy, S. El-Ansary, E. Aurell, S. Haridi. A statistical theory of Chord under churn. In: The 4th International Workshop on Peer-to-Peer Systems (IPTPS'05).
13. A. I. T. Rowstron, P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems, Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, p.329–350, November 12–16, 2001.
14. M. Castro, P. Druschel, Y. C. Hu, A. Rowstron. Exploiting network proximity in peer-to-peer overlay networks. In International Workshop on Future Directions in Distributed Computing (FuDiCo), June 2002.
15. M. Castro, P. Druschel, Y. C. Hu, A. Rowstron. "Proximity neighbor selection in tree-based structured peer-to-peer overlays", Technical report MSR-TR-2003-52, 2003.
16. R. Mahajan, M. Castro, A. Rowstron. "Controlling the Cost of Reliability in Peer-to-peer Overlays", IPTPS'03, Berkeley, CA, February 2003.
17. M. Castro, P. Druschel, A-M. Kermarrec, A. Rowstron, "One ring to rule them all: Service discovery and binding in structured peer-to-peer overlay networks", SIGOPS European Workshop, France, September, 2002.
18. M. Castro, P. Druschel, A-M. Kermarrec, A. Rowstron. "SCRIBE: A large-scale and decentralised application-level multicast infrastructure", IEEE Journal on Selected Areas in Communication (JSAC), Vol. 20, No, 8, October 2002.
19. M. Castro, M. B. Jones, A-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, A. Wolman. "An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer overlays", Infocom 2003, San Francisco, CA, April, 2003.
20. M. Castro, P. Druschel, A-M. Kermarrec, A. Rowstron. "Scalable Application-level Anycast for Highly Dynamic Groups", NGC 2003, Munich, Germany, September 2003.
21. A. I. Rowstron, A. Kermarrec, M. Castro, P. Druschel. SCRIBE: The Design of a Large-Scale Event Notification Infrastructure. In Proceedings of the Third international Cost264 Workshop on Networked Group Communication (November 07 – 09, 2001). J. Crowcroft and M. Hofmann, Eds. Lecture Notes In Computer Science, vol. 2233. Springer-Verlag, London, 30–43.
22. P. Druschel, A. Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In Proc. HotOS VIII, Schloss Elmau, Germany, May 2001.
23. A. Rowstron, P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In Proc. ACM SOSP'01, Banff, Canada, Oct. 2001.
24. S. Iyer, A. Rowstron, P. Druschel. "SQUIRREL: A decentralized, peer-to-peer web cache", 12th ACM Symposium on Principles of Distributed Computing (PODC 2002), Monterey, California, USA, July 2002.
25. M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh. "SplitStream: High-bandwidth multicast in a cooperative environment", SOSP'03, Lake Bolton, New York, October, 2003.

26. M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron, A. Singh. “SplitStream: High-bandwidth content distribution in a cooperative environment”, IPTPS’03, Berkeley, CA, February, 2003.
27. A. Mislove, A. Post, C. Reis, P. Willmann, P. Druschel, D. Wallach, X. Bonnaire, P. Sens, J. Busca. POST: a secure, resilient, cooperative messaging system, Proc. of the 9th conference on Hot Topics in Operating Systems, pp. 11–11, Hawaii, May 2003.
28. T.-W. J. Ngan, D. S. Wallach, P. Druschel. “Enforcing Fair Sharing of Peer-to-Peer Resources”, IPTPS’03, Berkeley, CA, February, 2003.
29. Rice University, FreePastry, http://freepastry.rice.edu/
30. Microsoft, SimPastry/VisPastry, http://research.microsoft.com/en-us/um/people/antr/pastry/download.htm
31. B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, J. Kubiatowicz. Tapestry: A resilient globalscale overlay for service deployment, IEEE Journal on Selected Areas Communications, vol. 22, no. 1, pp. 41–53, Jan. 2004.
32. S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, J. Kubiatowicz. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination, The 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Port Jefferson, New York, 2001.
33. Chimera, http://current.cs.ucsb.edu/projects/chimera/
34. K. Aberer, M. Hauswirth, M. Punceva, R. Schmidt. Improving Data Access in P2P Systems, IEEE Internet Computing, 6(1), January/February 2002.
35. K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, R. Schmidt. P-Grid: A Self-organizing Structured P2P System, SIGMOD Record, 32(2), September 2003.
36. K. Aberer, A. Datta, M. Hauswirth. P-Grid:Dynamics of self organization processes in structured P2P systems, Peer-to-Peer Systems and Applications, Lecture Notes in Computer Science, LNCS 3845, Springer Verlag, 2005.
37. A. Kumar, S. Merugu, J. Xu, E. W. Zegura, X. Yu. Ulysses: A Robust, Low-Diameter, Low-Latency Peer-to-peer Network, In European Transactions on Telecommunications (ETT) Special Issue on P2P Networking and P2P Services, 2004. Vol. 15, pages 571–587.
38. H.J. Siegel, Interconnection Networks for SIMD machines, Computer 12(6), 1979.
39. H. Shen, C.-Z. Xu, G. Chen. Cycloid: A scalable constant-degree lookup-efficient P2P overlay network, Journal of Performance Evaluation’s Special Issue on Peer-to-Peer Networks (6/29), 2005.
40. I. Gupta, K. Birman, P. Linga, A. Demers, R. van Renesse. Kelips: building an efficient and stable P2P DHT through increased memory and background overhead. Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS ’03). 2003.
41. D. Kempe, J. Kleinberg, A. Demers. “Spatial gossip and resource location protocols”, Proc. 33rd ACM Symp. Theory of Computing (STOC), pp. 163–172, 2001.
42. R. van Renesse, Y. Minsky, M. Hayden. “A gossip-style failure detection service”, Proc. IFIP Middleware, 1998.
43. K.P. Birman, M. Hayden, O. Ozkasap, Z. Xiao,M. Budiu, Y. Minsky. Bimodal Multicast, ACM Transactions on Computer Systems, 17:2, pp. 41–88, May 1999.
44. A. Demers, D.H. Greene, J. Hauser, W. Irish, J. Larson. “Epidemic algorithms for replicated database maintenance”, Proc. 6th ACM Symposium Principles of Distributed Computing (PODC), pp. 1–12, 1987.
45. A. Gupta, B. Liskov, R. Rodrigues. “Efficient routing for peer-to-peer overlays. Proceedings of the 1st Symposium on Networked Systems Design and Implementation” (NSDI 2004), 2004, pp. 113–116.
46. L. R. Monnerat, C. L. Amorim. “D1HT: A Distributed One Hop Hash Table. Proc. of the 20th IEEE Intl Parallel & Distributed Processing Symp.” (IPDPS), April 2006.
47. J. Buford, A. Brown, M. Kolberg. “Analysis of an Active Maintenance Algorithm for an O(1)-Hop Overlay”, IEEE Globecom 2007.

Distributed Hash Tables: Design and Applications

C.-F. Michael Chan and S.-H. Gary Chan

Abstract The tremendous growth of the Internet and large-scale applications such as file sharing and multimedia streaming require the support of efficient search on objects. Peer-to-peer approaches have been proposed to provide this search mechanism scalably. One such approach is the distributed hash table (DHT), a scalable, efficient, robust and self-organizing routing overlay suitable for Internet-size deployment. In this chapter, we discuss how scalable routing is achieved under node dynamics in DHTs. We also present several applications which illustrate the power of DHTs in enabling large-scale peer-to-peer applications. Since wireless networks are becoming increasingly popular, we also discuss the issues of deploying DHTs and various solutions in such networks.

1 Introduction

The Internet has grown to an enormous size, with nearly 600 million hosts as of early 2008 [1]. Coupled with this intense growth in network size is the proliferation of large-scale applications such as file sharing and multimedia streaming. These applications require the support of fast search on objects. Since traditional server-client model is no longer scalable to large group of hosts, peer-to-peer approaches have been proposed. One of such approaches is the distributed hash table (DHT), a scalable, efficient, robust and self-organizing overlay routing infrastructure for millions of hosts. DHTs have the potential to enable large-scale peer-to-peer applications, such as distributed file systems and on-demand video streaming. This chapter introduces DHTs and discusses their design issues and applications.

C.-F. Michael Chan
Stanford University, Stanford, CA, USA, e-mail: mcfchan@stanford.edu

S.-H. Gary Chan
Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, e-mail: gchan@cse.ust.hk

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_10, © Springer Science+Business Media, LLC 2010

DHT provides a unified platform for managing application data. Due to the specific nature of application data, traditionally a protocol designed for one type of application may not work well for other types of applications. DHTs break this barrier by providing a flat identifier-based routing and location framework, and a very simple API to applications.

DHT maps application data to keys, which are m-bit identifiers drawn from the identifier space. Nodes participating in the DHT are distinguished by unique identifiers drawn also from the same identifier space. Each node is responsible for a subset of the space, i.e. stores a subset of keys. Typically, a value is associated with a key, and is also stored at the node responsible for the key. Depending on the specific application, the value could be the address of the node storing the data or the data itself.

A DHT scheme defines how the overlay is structured, how node state is maintained and how routing is carried out. Regardless of the details, DHTs provide a two-method interface for applications:

- *insert(k, v):* Insert a data item with key-value pair (k,v) into the DHT.
- *lookup(k):* Retrieve the value v associated with key k. Return *null* if k is not found.

In contrast to IP routing, DHT routing is identifier-based. Each node stores about $O(\log n)$ overlay neighbors and employs a deterministic algorithm to route queries from requestor to the node storing the target key in $O(\log n)$ overlay hops. In Section 3, we discuss various DHT schemes. In particular, we shall see how the overlay is structured for efficient and scalable routing. We also outline techniques used to improve a DHT's query response time and robustness.

There are many applications making use of DHT. We present several examples that demonstrate how DHTs enable large-scale peer-to-peer services.

Another interesting DHT topic is its deployment in wireless networks which are becoming increasingly popular nowadays. Applications are being developed for mobile ad-hoc networks (MANETs), wireless sensor networks (WSNs) and wireless mesh networks (WMNs). It is desirable to have a DHT-like framework for wireless networks to support object location. We describe some major challenges in deploying DHT in wireless networks, and outline some solutions.

This chapter is organized as follows. We present the performance characteristics and design considerations of DHTs in Section 2. We discuss some examples of DHTs and how DHT is used in large-scale applications in Sections 3 and 5 respectively. In Section 6, we highlight the challenges of applying DHTs in wireless networks and present some solutions. We conclude in Section 7.

2 Performance Characteristics and Design Considerations

In this section, we describe some common performance characteristics of DHTs, and discuss some design issues (Section 2.1), followed by design considerations of DHT (Section 2.2).

2.1 Common Performance Characteristics

A DHT usually has the following desirable properties:

1. *Efficiency in routing:* A DHT overlay is structured so that queries for keys may be resolved quickly. Typical DHT schemes have a $O(\log n)$ bound on the length of search path (in overlay hops). To reduce routing delay, in recent years, locality-aware DHTs have been proposed so that routing hops are of short Internet length by taking advantage of locality information.
2. *Scalability:* Node storage and maintenance overhead grows only logarithmically with the number of nodes in DHT. This leads to its high scalability to large number of users.
3. *Self-organization:* A DHT protocol is fully distributed. Node joins, departures and failures are handled automatically without the need of any central coordination.
4. *Incremental deployability:* A DHT overlay works for arbitrary number of nodes and adapts itself as the number of nodes changes. It is functional even with one node. This is a highly desirable feature as it enables deployment without interrupting normal operations when new nodes join the overlay.
5. *Robustness against node dynamics:* Queries are resolved with high probability even under node dynamics. Further optimizations may further increase system robustness.

2.2 Design Considerations

In designing a DHT to achieve the above desirable properties, several issues need to be considered:

1. *Node dynamics:* Peers may join and leave at any rate. Since it is not possible for a central server to record the status of each peer, a DHT must be able to dynamic node departure or failure. In other words, the DHT structure must be maintained to ensure correct and efficient routing in the presence of node dynamics.
2. *Overlay path stretch:* Compared with IP routing, overlay routing to a certain node has in general higher latency, since overlapping traversals of some physical links is inevitable. The path stretch is defined as the ratio of the overlay route's latency to the underlying IP route's latency. In order to avoid large path stretch, DHT routes need to be optimized for low response time and such optimization techniques should be scalable to large groups.
3. *Hotspots:* Application data is generally skewed in terms of access probability. For instance, in video streaming, some segment of a video may be more popular than the other. A DHT scheme needs to consider the high lookups for a particular key (i.e., popular segment) by load balancing request processing among many nodes. The load balancing algorithm should be scalable.

3 DHT Schemes

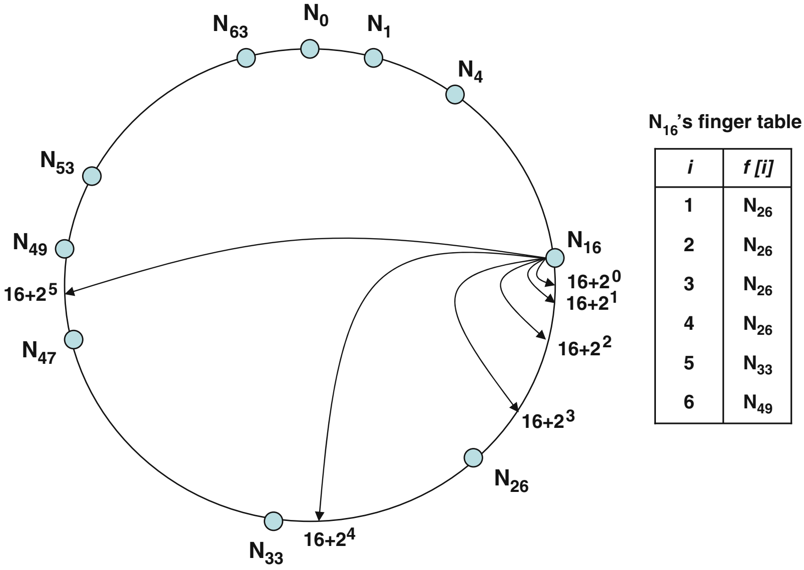
We describe several DHT schemes in this section. Note that keys and node IDs are drawn from the same m-bit identifier space and that keys are typically obtained by hashing meta-data, while node IDs are hashes of IP addresses or public keys. For a good hashing functions, nodes and keys are uniformly distributed in the overlay. Consequently, each node stores a similar share of keys. In our discussion below, we focus on operations including how keys are inserted, stored and looked up, and how the overlay is constructed and maintained. We assume m-bit identifiers are used, i.e. the identifier space is $[0, 2^m - 1]$. We denote a node with ID i as N_i and a key with ID j as K_j.[1]

3.1 Chord

The Chord DHT places nodes and keys in an identifier ring [23]. It is a simple DHT with great flexibility, which allows optimizations such as load balancing to be readily added as extensions to the basic scheme. We first describe the basic Chord, and then briefly discuss some simple but important extensions. The basic components and operations of Chord are as follows:

- *Key placement:* A key K_j is stored by the node N_i immediately following j in the identifier ring. In other words, N_i is chosen such that there is no node $N_{i'}$ where $j \leq i' < i$. We also call N_i the *successor* of j, denoted by $successor(j)$. Note that key and node IDs may coincide, in which case the key (K_i) is stored at the node with the same ID (N_i).
- *Successor Links:* Each node N_i maintains a link to its successor, the node N_j immediately following it. Such definition of successors for keys is also applicable to nodes. Denote N_j as $successor(i)$. As long as successor links are correct, a lookup is guaranteed to reach the key's successor, albeit via a long path going around the identifier ring one node at a time. A node achieves this by periodically running the following $stabilize()$ procedure. It asks its successor N_s for N_s's predecessor N_j. If $N_j \neq N_i$, N_i sets N_j as its successor. This happens if N_j is a newly joined node. Suppose $i < j < s$ and N_i and N_s are existing nodes in the DHT. When N_j first joins the overlay, it looks up j and gets N_s's address. It then sets N_s as its successor. Finally, N_s transfers keys in $(i, j]$ to N_j. It is shown in [24] that all successor links always converge correctly for any sequence of node joins and stabilizations.
- *Fingers:* A node N_i also maintains a *finger table*, which contains m entries, where m is the identifier length. The j-th finger stores the address of $successor(i+2^{j-1})$. An example of finger tables with 6-bit identifiers is shown in Fig. 1. Nodes periodically run a $fix\_fingers()$ procedure to refresh the finger table entries. The

[1] For simplicity, we also use i in place of N_i and j in place of K_j when there is no ambiguity.



N$_{16}$'s finger table

i	*f [i]*
1	N$_{26}$
2	N$_{26}$
3	N$_{26}$
4	N$_{26}$
5	N$_{33}$
6	N$_{49}$

Fig. 1 Example of Chord fingers

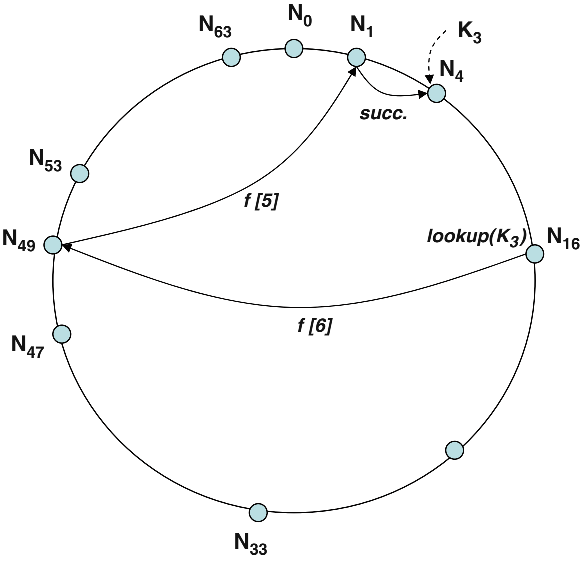


Fig. 2 Example of Chord routing with fingers. Each search hop is labeled with the finger table entry used in forwarding the lookup message. *succ* means the key lies within the segment between the current node and its successor

table is iterated in a round-robin fashion, where each call to the procedure refreshes the next finger. A refresh is achieved by looking up the finger's successor.

- *Key lookup:* The simple lookup algorithm forwards the request hop by hop using successor links until the key's successor is reached. The search path's length is $O(n)$ hops. Fingers drastically shorten the lookup path to $O(\log n)$ hops. In general, to look up key k, node i does the following. First, it checks if k is in $(i, successor(i)]$. If so, simply forward the request to the successor. Otherwise, forward the request to the largest finger $i + 2^{j-1}$ immediately preceding k. The same procedure is carried out at each intermediate node. Figure 2 shows an example of Chord routing with fingers. Notice that each hop (via a finger) reduces the distance to the key's successor by approximately half, thus giving the $O(\log n)$ path length. A more formal proof is given in [23].

The basic Chord is extended for greater robustness and load balancing. Two main ideas are key replication and employing virtual nodes. For key replication, instead of storing key k at only $successor(k)$, it is replicated on the r successors of k. This way, even if $j = successor(k)$ fails, the $successor(j)$ is still available for answering lookups for k. Setting $r = O(\log n)$ allows for robustness against very high degrees of node dynamics [23]. A node may also run multiple virtual nodes depending on its processing power and bandwidth. This way, heterogeneous peers may fully contribute their spare and varied resources to enhance the DHT's quality.

3.2 Pastry

In Pastry, an identifier is made of D digits. For example, a 128-bit identifier is broken up into 32 4-bit digits. To simplify the exposition, we assume 2^{bD}-bit identifiers, where b is the number of bits per digit. The components of Pastry are as follows:

- *Key placement:* A key k is placed at the node whose ID i is numerically closest to k.
- *Leafset:* Each node i keeps a list of L neighbors, where L is an even number. $L/2$ of those have IDs numerically closest to and smaller than i. The other $L/2$ have IDs numerically closest to and larger than i. This is similar to Chord's successor links in that a node may employ correct leafset information to resolve lookups in $O(n)$ hops.
- *Routing table:* A Pastry routing table consists of D rows, one for each digit. A row consists of 2^b entries. The j-th entry of the i-th row stores the address of a neighbor whose ID shares the same i significant digits with the current node's ID, but with the $i+1$-th digit equal j. Note there may be no node matching the requirements of some entries. In this case, the entry is left empty (Fig. 3).
- *Key lookup:* To look up a key k, a node i first checks its leafset. If a neighbor is numerically closest to k, the request is forwarded to that neighbor, and the lookup is complete. Otherwise, the routing table is checked for a node whose ID shares one more digit with k than i. If there is no such node (i.e., the route entry is empty), the message is forwarded to the node j matching as many digits with

Routing table of node 3123

Level *i*	***i+1* digit**			
	0	**1**	**2**	**3**
0	0xxx	1xxx	2xxx	–
1	30xx	–	32xx	33xx
2	310x	311x	–	313x
3	3120	3121	3122	–

Fig. 3 Pastry routing table of node 3123. 8-bit identifiers are divided into 4 2-bit digits. All numbers are in base 4. "xxx" is an arbitrary string of base-4 numbers

k as i does, but is numerically closer to k. As shown in [22], node j exists with high probability given that L is reasonably large. A lookup fails, however, if $L/2$ nodes with consecutive IDs fail at the same time. Note that each hop effectively brings the lookup one-digit closer to the target key, thus the $O(\log n)$ bound on the search path length.

- *Node dynamics:* Node joins are handled similarly as in Chord. A new node looks up its own ID i. Suppose node j is responsible for the key i. Node i asks node j for its leafset and turns it into its own leafset by adding j and removing the node farthest away in terms of ID distance. Up to L nodes in j's leafset will need to be contacted so that they could update their leafsets to include i. j also updates its leafset to include i. The two adjacent neighbors in i's leafset then transfer keys to i.

  Node i's routing tables are populated by route entries from intermediate nodes along the search path it initiated when it looked up its own identifier. For each route entry, there may be multiple candidates. The entry with the lowest distance is chosen. The distance metric reflects the end-to-end delay between i and the neighbor. Once the route entries are filled, i asks for those neighbors for their routing tables, and updates the route entries again by replacing route entries with lower-distance entries. This way, locality-awareness is built-in during overlay construction. Reference [22] gives a formal proof that locality is preserved by using routing entries from physically close nodes.

  A node handles neighbor departures and failures lazily. Replacement of a failed neighbor occurs when the node forwards a message in vain to the neighbor. A failed leafset neighbor is replaced as follows. The node contacts the live leafset neighbor P with the smallest ID or the largest ID depending on which side of the node the failed neighbor resides in the leafset. P returns its leafset to the node, which then updates its own leafset by removing the failed neighbor and adding a new neighbor from P's leafset. The j-th entry of the i-th row is handled as follows. The node successively asks neighbors in the other i-th row entries for their j-th route entry. This way, a number of candidates for this particular entry is found for replacing the failed neighbor. The one closest in terms of the distance metric is chosen.

3.3 Kademlia

Kademlia is a widely implemented DHT [12, 16, 25]. It stands out among other DHTs by using an uncommon metric – the XOR metric. The distance between two nodes is defined as the bit-wise XOR of their identifiers. For instance, the distance between nodes with IDs 1100 and 0010 is 1110.

The following description of the Kademlia protocol is facilitated by a binary-tree view of the network. Each node is a leaf and the (labeled) path from the root to the leaf is the unique prefix of the node's ID. Figure 4 shows an example of this binary-tree representation of a Kademlia network.

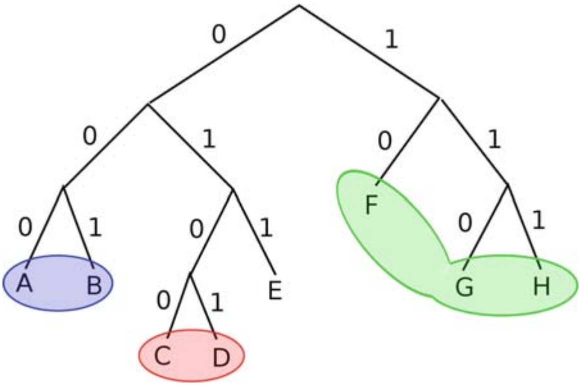


Fig. 4 Binary tree abstraction of a Kademlia network

- *Node state:* As in Chord and Pastry, Kademlia nodes keep track of peers in certain ID ranges. Peers are put into *k-buckets*, where bucket i is for peers at a distance between 2^i and 2^{i+1}. Each node must know of at least one node in each bucket, if there is some node in that ID range. For example, node E in Fig. 4 needs information of at least one node from each circled group. The size of a bucket is limited by the system parameter k. If a bucket is full, a new node will not be added unless the least-recently-used node in the bucket fails to respond. This modified LRU replacement scheme has the benefit of defending against malicious attempts to flush node state with new (and bogus) information.
- *Key placement:* A key is placed at the k nodes whose IDs are closest to the key.
- *Key lookup:* A key x is looked up as follows. The querying node n first scans stored information to locate the k nodes closest to x. It then sends the query to α ($< k$) nodes from the set. Upon receiving replies, n chooses again the k closest nodes to x, and then sends queries to α of them. The formal analysis of this algorithm is involved, but the idea is like walking down the binary tree from the root, where at each internal node, n queries some node it knows about in that subtree, receives information about a node further down the subtree and closer

to the destination, and repeats the process by moving down the correct edge. The algorithm is also similar to that of Pastry's prefix-based routing, where with each step, n gets closer to peers storing x by "correcting" significant bits in the nodes queried. The system parameter α governs the degree of parallelism in lookups. By employing parallel look ups, Kademlia sacrifices some (constant) increase in bandwidth for flexibility in selecting low-latency paths.

- *Node dynamics:* Unlike other DHTs, nodes in Kademlia learn about each other through queries. When a node joins the network, it looks up its own ID, so that peers along the query paths know about it. Similarly, there is no explicit messaging when a node leaves. The departure is only discovered when other nodes attempt to ascertain the node's presence before evicting it from a bucket. There is also the need for republishing keys to ensure that they are stored at the k closest nodes. Periodically, a key is republished (by a lookup for the key and data transfer to new holders). To reduce overhead, a node receiving this republication will not republish the key in the next period. This way, as long as some node first republishes the key, other nodes who were also obliged to do so would not need to waste the bandwidth to perform the same operations.

In practice, further optimizations are employed to speed up the protocol. When a bucket is full, new information for that bucket is cached. If some node in the bucket fails afterwards, the cached information can be used to fill in the gap immediately. Lookups are sped up by expanding the routing table so that multiple bits may be matched in each step. This is similar to setting $b > 1$ in Pastry.

3.4 Other DHTs

Several DHTs based on other types of overlay structures. The Content Addressable Network (CAN) constructs a d-dimensional hypercube [20]. Identifiers are divided into d digits. Each node owns a subset of the coordinate space and maintains a set of $d-1$ neighbors. A neighbor shares the same range of values with the node in $d-1$ dimensions, but not in the remaining dimension. Lookups are resolved by forwarding along the correct dimensions such that each hop matches one more digit in the key. A lookup is thus resolved in $O(d)$ hops. Note that by setting $d = \log n$, we get the $O(\log n)$ bound as in Chord and Pastry.

Tapestry is similar to Pastry in that it constructs a prefix-based routing table [29]. The main difference is that there is no leafset.

Viceroy maintains a butterfly structure [15]. Key management is similar to that in Chord. Nodes are distributed on an identifier ring. The difference is in the overlay neighbor selection. A node joins one of $\log n$ level rings. Each level ring is connected by level-ring links between nodes with adjacent identifiers on the same level. A node at level i maintains some pointers to neighbors at levels $i-1$ and $i+1$ in a way that lookups are resolved in $O(\log n)$ hops. Viceroy is most characteristic in its constant node state.

4 Design Fundamentals

In this section, we discuss some fundamental design issues. With so many DHT protocols, it is natural to ask how they fair against one another in various scenarios. A detailed comparison between the various DHT schemes is presented in [9], which studies the effect of the overlay's structure on system performance in terms of static resilience, path latency and local convergence. We also briefly discuss interesting findings concerning tradeoffs between the network diameter of a DHT and the amount of routing information a node stores [26].

4.1 Static Resilience

The robustness of a DHT depends on both its recovery mechanisms and structural properties. The study of static resilience reveals how well a DHT's structure copes with node failures without any recovery operations, and sheds light on how such mechanisms should be provisioned. For instance, if the structure is inherently resilient against node failures, then recovery mechanisms need not be frequently carried out.

Intuitively, a DHT is more robust when there are more alternative routes a query can take to the key holder. If some nodes fail, it is still possible to route around the failures to the destination. Clearly, the DHT's routing algorithm dictates how flexible queries may be forwarded. The more flexible the DHT's structure, the more robust the system.

Take Chord and Pastry as examples. In the original Chord proposal, a query is resolved by halving by the distance to the destination at each step. However, this is really a restricted version of the more general algorithm in which the distances covered by each step need not be in decreasing order. In other words, we may take the steps in any order, as long as there is one step for each distance (2^i) to be covered. The prefix-based phase of Pastry's routing algorithm is vastly different. At each step, we must fix the most significant unmatched bit. This means only one route entry can be used. If the node in that entry has failed, the query has to be redirected to leafset neighbors. Clearly, if not for the leaf sets, Pastry's routing algorithm results in little flexibility in routing, and hence lower static resilience.

4.2 Path Latency

The structure's flexibility also affects the latency of query paths. Once the routing table is fixed, there is limited choice in a query's next hop. A flexible structure enables selection of neighbors with better latency, and hence better choices in choosing next hops. As a result, the overall path latency can be reduced.

In Chord, a node n's i-th finger is originally defined to be the successor of $n+2^i$. However, this requirement can be relaxed to include nodes between $[n+2^i, n+2^{i+1})$, making neighbor selection very flexible. In Pastry, a route entry in the i-th row can be chosen from a maximum of $2^{b(D-i-1)}$ peers. A Kademlia node also has great flexibility in filling its k-buckets. For the i-th bucket, there are up to 2^i choices. On the other hand, CAN requires a neighbor to only differ by a single-bit, thereby restricting the selection to one neighbor.

4.3 Local Convergence

Consider two queries from two nodes that are nearby in the physical network. A DHT exhibits good local convergence properties if the queries converge at a common peer that is also close to the two nodes. The most significant advantage of a high degree of local convergence is that caching can be made more effective. Frequently, an application on top of DHT caches lookup results along the query path in hopes that future queries would hit the cached entries early in their lookup paths. The Cooperative File Storage (CFS) to be discussed in Section 5.1 is an example of such an application.

It is discovered that a structure with flexible neighbor selection exhibits good local convergence properties. Chord, Pastry and Kademlia with low-latency neighbor preference are desirable designs in this regard.

4.4 Network Diameter and Node State Tradeoffs

Another interesting observation is that for most of the DHTs we have discussed, the size of a node's routing table is either $O(\log n)$ (Chord, Pastry, Kademlia, Tapestry) or $O(d)$ (e.g. CAN), with network diameters of $O(\log n)$ and $O(n^{1/d})$ respectively. (The exception is Viceroy, which has constant node state with high probability and $O(\log n)$ diameter). The question asked in [26] is whether these are actually lower bounds on the network diameter, i.e. $\Omega(\log n)$ and $\Omega(n^{1/d})$. The answer will give us an idea of whether the tradeoff has been optimized by existing DHTs, and whether there is room for improvement.

It is, in fact, possible to go beyond the $\Omega(\log n)$ lower bound. The idea is to construct a (directed) $\log n$-ary tree. Each node except the root has a edge pointing back to the root. This way, all nodes are reachable from one another (thus satisfying routing correctness) and any path between two nodes is at most $h+1$, where $h = \log_{\log n} n = \frac{\log n}{\log\log n}$ is the height of the tree. Obviously the longest path (i.e., the network diameter) is $O(\frac{\log n}{\log\log n})$, which is asymptotically smaller than $O(\log n)$. Also, each node only maintains $O(\log n)$ neighbors. Unfortunately, this structure puts the root under heavy congestion, an unacceptable state of affairs in peer-to-peer systems.

Another discovery is that DHT routing algorithms can be classified into two major categories – uniform and others. Uniform routing algorithms treat every query without discrimination against the source of the query and the location of the processing node. For instance, a Chord node would process a query by looking up its finger table no matter where the query came from or where the node resides in the identifier ring. In this case, where nodes have $O(\log n)$ state, $\Omega(\log n)$ is indeed the lower bound on the network diameter. The reason is that the DHTs aim to provide a uniform load across nodes (i.e., to avoid congestion), therefore they cannot achieve the lower bound of the $\log n$-ary tree for network diameter, but only the larger lower bound of $\Omega(\log n)$.

It turns out that a butterfly network can be (deterministically) designed to achieve a $O(\frac{\log n}{\log\log n})$ diameter with $O(\log n)$ node state and a non-uniform routing algorithm such that no extra load is imposed on any node. However, this bound is known so far to be possible with static nodes, and that some links would still have $O(\log n)$ extra load. It remains unknown whether the bound can be achieved deterministically in the presence of node dynamics.

5 Applications

In this section, we present several applications of DHTs. We first describe the application briefly, and then discuss their properties with reference to those of DHTs.

5.1 Cooperative File Storage (CFS)

CFS is a large-scale distributed storage system supporting multiple file systems [7]. Files are divided into blocks, which are stored in CFS servers. Blocks are distinguished by unique IDs obtained by hashing block contents and public keys of block publishers. Clients read a file by looking up and retrieving blocks that make up the file. File publishers insert and periodically refresh files so that the blocks do not get deleted by CFS servers after a fixed expiry time. Each file system is a tree rooted at a *root block*, which is signed by the publisher with its private key. The root block's ID is the publisher's public key. Clients then name a file system according to the system publisher's public key.

CFS consists of three layers. The (lowest) *Chord layer* is responsible for looking up blocks given their IDs. A *DHash layer* is built on top of the Chord layer. It manages block storage, replication and caching. The *file system layer* converts blocks obtained from the DHash layer to files and provides users and applications a file system interface. We focus on how the Chord and DHash layers collaboratively gives CFS various desirable performance qualities:

- *Highly scalable:* CFS inherits scalability from the underlying Chord layer. Node state and control overhead in searching and routing table maintenance grows logarithmically with the number of CFS servers. Since files are broken into blocks, a large file will not exhaust a particular server's storage, as will be the case in file-based storage systems. Also, since block IDs are hashes of block contents, and servers are distributed approximately uniformly around the Chord ring, it is expected that each server stores around the average number of blocks.

- *Robust to massive server failures:* With the original Chord protocol, a block is stored at the successor of its ID. The DHash layer improves robustness by replicating the block in the k successors of the ID. Since consecutive servers on the overlay are likely to be distributed in the physical network, their failure rates are likely to be indepedent. If a server fails with probability p, then a block is inaccessible (removed from CFS due to failure of servers storing it) with probability approximately equal to p^k, which could be made small with modest values of k.
- *Effective load balancing:* A popular file may put excessive load on its server. CFS proposes two approaches to balance the load for popular files. First, for large popular files, CFS inherently balances the load by breaking them into blocks. Furthermore, block replication allows lookups to be directed to a number of servers instead of the immediate successor of the block ID. Second, for small popular files (and blocks), caching is employed to reduce the load on the block successors. In a nutshell, the queried blocks are cached at intermediate nodes along the overlay search path. Since the Chord lookup algorithm halves the distance to the target node every hop, cached copies near the target node tend to overlap, thereby boosting the cache hit rate.
- *Fast file access:* The Chord searching algorithm is improved to provide server selection in CFS. At every hop during lookup, a node may choose among the set of successors and fingers to forward the query to. Two potentially conflicting considerations are latency to the next hop and the ID space covered by this forwarding. A hybrid metric balances the latency and overlay progress (see Section 4.3 of [7] for details.) With server selection enabled, CFS retrieval rates are comparable to and has lower variance than that of direct TCP-based transfers such as FTP.
- *Secure against data change and flooding attacks:* Two major security concerns of distributed file systems are unauthorized data modification and flooding attacks. CFS servers mandates node ID to be the SHA-1 hash of the node's IP address, which is in turn authenticated via challenges with random nounces sent to the claimed address. An attacker wishing to modify a specific data block would have to control a large set of IP addresses to be able to store the target block. However, entities owning many IP addresses are more easily identified (such as big organizations) than individual attackers owning a small number of addresses. This way, the system is quite secure against intentional deletion of data. CFS also limits the amount of data publishers can put into the system to guard against flooding attacks. Suppose the quota is f, a (small) fraction of the total storage space in the CFS system, then an attack would have to employ $1/f$ hosts to exhaust the system's storage.

5.2 Scribe

Enabling large-scale multicast in the Internet is an important topic of research. Traditional (IP) multicast has been proposed long ago, but has seen limited deployment on a large scale due to network management issues and other deployment concerns. To support large-scale multicast, all networks involved must have multicast support enabled, which is often not possible. There are also issues of assigning (unique) multicast addresses to groups and authorization of operations such as group creation, sending messages to a group and receiving messages. The lack of multicast support has stirred interest in application-layer multicast (ALM) where end-hosts provide multicast by relaying messages at the application layer.

Scribe is an ALM scheme based on the Pastry DHT and solves several important issues with IP multicast [5]. While ALM is in general less efficient in terms of packet delivery time and link stress, Scribe provides scalable multicast with acceptable delay penalty and link stress as compared to IP multicast. In Scribe, a group ID (multicast addresses) is generated by hashing the creator address and group name. The creator then sends a create message to the node responsible for the group ID via Pastry routing. This node is the rendezvous point (RP) for node joins and message sending for the group. A joining node sends a join message with the target group ID to the RP. Intermediate nodes on the path become forwarders in the resultant multicast tree. They store a children list so that future multicast messages may be forwarded to the downstream. Senders forward packets to the root for dissemination throughout the tree. We focus our discussion on Scribe's scalability, efficiency in propagating multicast messages and fault tolerance.

- *Scalability in group size:* An important hurdle in scaling multicast to large groups is the node joining process. In particular, it is essential to enable nodes to join with low control overhead. This is obviously not possible with a centralized server storing a (sub)set of existing nodes in the group. Scribe enables efficient node joins by exploiting Pastry's properties. First, Pastry's routing mechanism allows the join overhead to be distributed evenly among nodes. Second, the RP need not handle all joins, as join messages may hit an existing node (either a forwarder or receiver) along the path and processed there locally. Third, the uniform distribution of nodes in Pastry ensures a balanced multicast tree. Furthermore, since Pastry employs prefix-based routing, the multicast structure is guaranteed to be loop-free.
- *Scalability in number of groups:* Two problems that hinder scalability in supporting many multicast groups are the assignment of group addresses and load balancing control information of groups among participating nodes. The address assignment problem is solved by employing DHT keys as group IDs, which are obtained by hashing the group creator and group's name. With a suitable hash function, the probability of group ID collision is low. This way, group IDs can be assigned in a totally distributed manner. Hashing also distributes the group roots uniformly in the Pastry overlay. Since nodes are also distributed uniformly in the overlay, the control information for all groups, i.e. parent and children pointers,

is likely to be distributed uniformly among the nodes. Indeed, for fairly large networks (100000 nodes and 1500 groups), Scribe nodes have on average less than 10 children pointers. Additionally, nodes may offload children connections by asking some of them to connect to their siblings instead. This so-called *bottleneck remover* algorithm dynamically adapts node stress to network conditions, and effectively reduces the maximum number of children connections under the same network conditions.

- *Multicast efficiency:* Scribe achieves low delay penalty compared to IP multicast for two reasons. First, the paths from receivers to the RP are short – with DHT routing, the path length is logarithmic in number of nodes in the network. Second, since Pastry enforces locality in choosing overlay neighbors, each overlay hop in the multicast tree is likely to be short in terms of latency. These two factors contribute to an overall low delay in disseminating messages in the multicast trees.
- *Fault tolerance:* Scribe exploits efficient DHT routing to localize repairs of the multicast trees in case of node failures. If a node fails, its children can simply route a join message to the group ID to discover a new parent. Failure of the RP has a more detrimental effect on the multicast group since it stores, apart from children points, other control information such as authorization credentials, identity of the group creator and senders. Scribe replicates the state at the RP to its *k* closest nodes in the Pastry overlay. If the RP fails, its children use Pastry to find the new root, which is the closest overlay node of the failed RP. Since this new root already possesses the group state, normal operation may resume quickly.

5.3 VMesh

Multimedia streaming is one of the most popular services in the Internet. A particular type of multimedia streaming is video-on-demand (VoD), in which users may request for any video at any time. An important feature of VoD is user interactivity, i.e. users should be able to start viewing the video at any point, and may jump forward and backwards as they wish. VMesh provides scalable and efficient VoD with rich user interactivity based on DHT [27].

In VMesh, a video is divided into segments initially stored at the video server. A peer downloads some segments from the server to its local storage and searches for segments it does not own via a DHT and a video mesh. The stored segments are not removed even if the peer is not viewing them. Instead, the peer streams these segments to other peers.

A DHT is constructed to facilitate segment location. Each peer registers the segments they store by inserting keys into the DHT. A segment's key comprises three parts – the video ID, segment ID and segment owner's network location, in decreasing order of significance. The network location field encodes the node's location in the network via space filling curves, which provide a one-dimensional proximity metric for parent selection during segment location streaming. Since the video and segment IDs occupy more significant bits, keys for the same segment from different peers are stored in the same region in the overlay (e.g., in a (small) arc of the Chord ring). With key replication, a node may answer segment queries with a list of parents owning the segment. Once the requesting peer receives this list, it could ask nearby parents to stream the segment.

Peers maintain in addition to the DHT a *video mesh*. In a nutshell, peers storing segment i maintains pointers to peers storing the next ($i+$ 1-th), previous ($i-$ 1-th) and the same (i-th) segments. These pointers are obtained by locating the segments using DHT and storing the addresses returned. Whenever a peer is about to exhaust its current segment, it asks its parents for the addresses of the peers holding the next segment and starts buffering it for better video continuity. When a peer wants to jump to a new (far-away) segment, it queries the DHT for the segment's owners. If the segment ID is close to the current segment's ID, the parent pointers are followed instead.

VMesh also addresses non-uniform segment popularity. Typically, certain segments of a movie are more popular and thus accessed more frequently. It is desirable to have more replica of popular segments for load balancing purposes. VMesh peers employ a distributed averaging algorithm to estimate the popularity and number of segment replica in the network, and adjust the segments they store accordingly [17].

The following are desirable properties of VMesh:

- *Scalability:* VMesh scales well to large number of peers. By having peers store and stream segments to other peers, the video server's workload is greatly reduced. The peer-to-peer segment location algorithm is locality-aware. This allows peers to find physically nearby parents, thereby reducing streaming overhead and latency. Popularity-aware adjustment in segment storage balances streaming load among a suitable number of peers, thereby avoiding hotspots for popular segments.
- *Robustness to parent failure:* Since a peer receives a list of parents as an answer to a DHT query, it may stream the segment in parallel from multiple (nearby) parents. If a parent fails, the other parents may share the failed parent's load while the peer searches for a new parent.
- *Efficient bootstrap and jumping:* VMesh provides low startup delay for newly joined peers and low delay in jumping to arbitrary positions in the video. Since segment location is performed as a DHT search, queries are resolved efficiently. Also, a new peer is given a list of parents with their location in the network. Such locality awareness allows the peer to choose nearby parents to reduce streaming delay. The same principle applies for jumping peers.

5.4 Internet Indirection Infrastructure (i3)

Unicast routing has been the main service provided by the network layer in the Internet. The need for large-scale multicast, anycast and host mobility services has sparked much interest in the research community. The *Internet Indirection Infrastructure* (*i*3) is an overlay framework supporting very general routing services. While it does not require any particular structure in its implementation, a DHT is a natural and desirable candidate.

Routing in *i*3 is entirely identified-based. Instead of an address, peers send messages to identifiers. Receivers interested in those messages place triggers in the overlay. Basic triggers are the ordered pair $< id, addr >$, where *id* is the identifier messages are sent to, and *addr* is the receiver's IP address. Triggers with the same *id* are stored in the same DHT node. Messages from the sender are routed to the node storing the triggers with the given *id*, who then forwards it to each registered address. Such indirection allows for natural multicast, anycast, host mobility and service composition.

- *Multicast:* A multicast group is identified by a group id G. Nodes join the group by inserting triggers $< G, addr >$ into the DHT. Senders simply have to send messages to the id G. A scalable multicast scheme which extends this basic idea is presented in [13].
- *Anycast:* Servers insert triggers of the form $< S, addr >$. Nodes then locate a server by sending a request to id S. The service id S is separated into a prefix and the suffix. The latter is used for load balancing or locality-awareness in selecting servers. For instance, location information may be enocded into the ID suffix and the suffix of the requested ID to achieve locality-awareness. On the other hand, servers could insert multiple triggers with random suffixes proportional to their capacity to achieve load-balancing.
- *Host mobility:* When a node moves, it inserts a trigger $< ID_{old}, addr_{new} >$, where ID_{old} is the ID of its previous address and $addr_{new}$ is its new address. One may consider them as the home address and foreign address respectively in IP mobility. Changes in the node's address is reflected by inserting new triggers with updated $addr_{new}$. The reader is referred to [30] for details.
- *Service composition:* In certain applications, messages may need to be processed in between the server and service requestor. *i*3 achieves this by making triggers more flexible. In its more powerful form, triggers allows the *addr* field to be an ordered list of identifiers (we may consider an address an identifier as well). This list behaves like a stack, with the top of the stack on the left end of the list. Suppose a packet from server S to requestor R needs to pass through a processing node P. Figure 5 shows how data is redirected to the processing node before reaching R. The advantage of this approach is that processing is receiver-driven. The server need only send one format of the application data, while different receivers may process the data individually for compatibility. Numerous examples of this redirection mechanism is given in [14].

From the above, we see the need to organize a large number of triggers and route messages to triggers efficiently. Thus, DHTs are a good choice for the overlay. DHTs can also provide robustness through trigger replication and reduce latency by locality-aware techniques. Given this scalable, efficient and robust overlay, the above routing services can be provisioned in large scale with desirable performance.

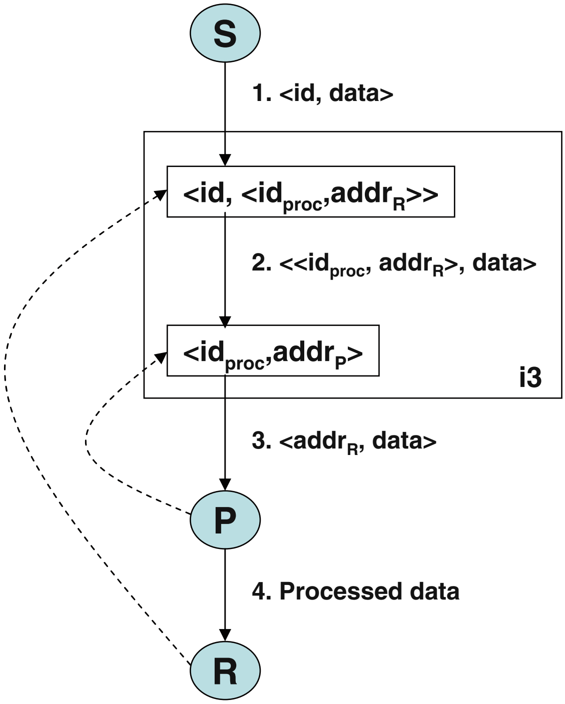


Fig. 5 Example of service composition in *i*3. The receiver *R* inserts the trigger $< id, < id_{proc}, addr_R >>$ and the processing node *P* inserts the trigger $< id_{proc}, addr_P >>$. The server *S* sends the data to *id* without knowing the data will be processed by *P*. The message is redirected to the trigger $< id_{proc}, addr_P >$ and then forwarded to *P* for processing. *P* obtains *R*'s address from the message and finally sends the processed data to *R*

6 DHTs in Wireless Networks

Recent advances in mobile technology, such as mobile computing power and wireless communications capabilities, has sparked great interest in building scalable applications for large-scale wireless networks. In this section, we highlight the characteristics of wireless networks, how such characteristics hinder the deployment of DHTs and various approaches to adapt DHTs to such environments.

6.1 Characteristics of Wireless Networks

There are three major types of wireless networks:

- **Mobile ad-hoc networks (MANETs)** are characterized by high node mobility and lack of infrastructure support. Nodes of low processing power and move in random directions at random speeds, thereby making the topology unstable. An example of MANETs is the vehicular ad-hoc network, where wireless devices in vehicles interact with each other while moving at high speeds.

- In **wireless sensor networks (WSNs)**, thousands of sensor nodes are scattered onto a large geographical area. The sensors are very low-profile devices with very limited processing power, memory and battery. The primary concern of a sensor network is its lifetime, therefore protocols for sensor nodes focus on power conservation. The network serves as a repository of sensed data, such as temperature and humidity. Queries are not directed towards a specific node. Instead, multiple nodes cooperate to reply with aggregated data that is sensible to the application.
- **Wireless mesh networks** provide an alternative way for network access in areas where infrastructure is difficult and/or costly to install. In its most general form, a wireless mesh network consists of a number of gateways, mesh points and end-hosts. The gatways provide access to the wired Internet. Mesh points are typically small devices with limited processing power and memory mounted on lamp posts or rooftops. They extend the network coverage of the gateway wirelessly. By associating with a default gateway, they forward Internet-bound traffic from associated users to the gateway. Routing is also conducted between different mesh points. While users are mobile and may switch their associated mesh points at any time, the gateway and mesh points are generally stationary.

While these wireless networks have different uses and greatly diversified node capabilities, they share some characteristics which are of paramount importance in considering the design of DHTs in such environments:

- *Limited shared bandwidth:* The (broadcast) wireless channel is shared by all nodes and has limited bandwidth. A DHT scheme has to be light-weight in terms of bandwidth consumption.
- *Lack of routing infrastructure:* Unlike the wired network, where routing is readily available at low cost, routing in wireless networks is non-trivial. The two problems associated with routing in this case are high maintenance overhead of routes and inefficient bandwidth utility due to long routes.
- *Low processing power and memory capacity:* Wireless devices are usually low in processing power and limited in memory capacity. A desirable DHT scheme should be light-weight in terms of node processing and storage.

In the following sections, we discuss two main issues with wired DHTs in wireless networks and outline solutions proposed in the literature.

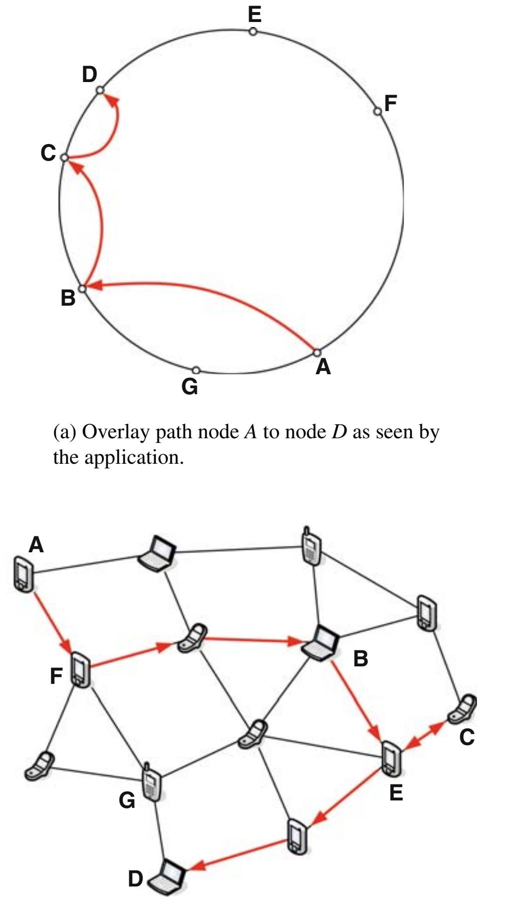
6.2 Challenges of Using DHTs in Wireless Networks

It is argued that DHTs designed for the wired network are not directly applicable to wireless networks. The two main issues are as follows.

1. *Overlay mismatch problem:* In the DHT overlay, any pair of nodes is considered to be “one-hop” apart. While this abstraction is somewhat valid in the wired network where routing is efficient and low-cost, it is definitely not the case in

wireless networks. In particular, a single overlay hop may map to multiple physical links, resulting in inefficient bandwidth utility [19, 21, 28].

Figure 6 shows an example of the overlay mismatch problem. The overlay path from node *A* to node *D* is 3 hops, as perceived by the application. However, it actually spans 8 physical links, one of which is traversed twice. Notice that the shortest path via nodes *F* and *G* cannot be obtained by the overlay lookup algorithm since the algorithm is fixed with routing on identifiers only and with no consideration of physical proximity of nodes.



(a) Overlay path node *A* to node *D* as seen by the application.

(b) Actual path traversed in the physical network. This path is much longer than the perceived length of the overlay path and may traverse a link multiple times, e.g. the link between nodes *C* and *E*.

Fig. 6 Illustration of the overlay mismatch problem. The overlay path from node *A* to node *D* spans 8 links, whereas the shortest path is only 3 hops. It is impossible to tell by considering the overlay alone that the shortest path is $A \rightarrow F \rightarrow G \rightarrow D$

2. *High maintenance overhead:* DHT maintenance procedures ensure routing convergence and efficient routing (in terms of number of overlay hops). While the overhead incurred by such procedures is acceptable in the wired network, the same procedures are demanding for bandwidth-limited wireless networks. For example, in Chord, a node periodically run the *stabilize(·)* and *fix_fingers(·)* methods to ensure that its successor and finger table entries are up to date. Each of these operations may require a route discovery. For reactive routing protocols, this incurs up to $O(n)$ overhead, where n is the number of nodes [10, 18]. Proactive routing requires periodic flooding of topology control, which is particularly costly in wireless sensor networks where power is of utmost importance [6]. It is also difficult to achieve convergence in MANETs as the topology changes quickly. Multiple route discoveries may be needed in the presence of mobility. Furthermore, overlay mismatch exacerbates this issue, since much bandwidth is spent obtaining routes that are unnecessarily long. The situation could be even worse than simple flooding in resolving requests for data items [19, 28].

6.3 Search Approaches for Wireless Networks

The overlay mismatch problem and high maintenance overhead incurred by traditional DHTs must be resolved in order to make DHTs feasible for wireless networks. In this section, we discuss optimizations and alternate approaches that aim to resolve these two issues.

- *Adding location awareness:* The overlay mismatch problem is caused by blindly layering the DHT layer directly on top of the routing protocol. An intuitive idea is to let nodes choose overlay neighbors that are physically close. This is achieved by cross-layering the two overlay routing and underlay routing. Several methods have been proposed.

  In Ekta, Pastry nodes collect DSR routes by overhearing control packets [19]. A node only maintains routes to physically close overlay neighbors as indicated by hop counts in DSR routes. The correctness of overlay routing is maintained because in prefix-based routing that Pastry employs, choosing any one of the possible nodes for a given bit position would ensure route convergence. The key idea here then is to choose the node that is closest in the underlay.

  MADPastry employs clustering to enforce physical proximity of overlay neighbors [28]. The identifier space is divided into m (equal) partitions, where m is the number of clusters. Each cluster's identifier space starts with a different prefix. A landmark heads a cluster and floods beacons throughout the cluster so that new nodes may join the appropriate cluster. A node identifiers is composed of its cluster's prefix and a suffix obtained via hashing its address or public key. The Pastry routing table is stripped down to contain only m entries, one for each cluster. When resolving a query, the first overlay hop is taken to the cluster housing the target key. After that, leaf-set routing is used to reach the reference node. As long as clusters are relatively small, the search path is confined within a small geographical region, thus solving the overlay mismatch problem.

If nodes know about their geographical locations, e.g. by means of GPS, it is possible to structure their identifiers to match the underlay topology. A common technique is to apply a specialized hash function to map node positions (in 2D) to a 1D identifier space. For instance, the hash function employed by Georoy and MeshChord maps node locations to identifiers on a unit ring in a way that nodes in the same geographical region will also be in the nearby each other (in the same segment) in the unit ring [3, 8]. An illustrative example is given in Eq. (3.1) and Fig. 3 of [8]. The concept of cell-addressing in Cell Hash Routing is of similar spirit to Georoy and MeshChord [2].

Another technique is to replace traditional DHT overlay routing with geographical routing. The Geographic Hash Table (GHT) and CHR use Greedy Perimeter Stateless Routing (GPSR) to route queries to reference nodes [21]. GPSR greedily forwards queries to the target location in the physical network [11]. There is no need for overlay routing, thereby solving the mismatch problem.

- *Reducing maintenance overhead:* Various approaches have been proposed to curb maintenance overhead. One method is to reduce the amount of overlay routing information. This may involve reducing the number of overlay neighbors and/or lowering the frequency of refreshing overlay connections. Ekta tries to reduce route discoveries by overhearing DSR routes to overlay neighbors. MAD-Pastry achieves low overhead by storing a degenerate Pastry routing table with only as many entries as there are clusters. Virtual Ring Routing (VRR) takes this approach one step further by only storing routes to successors and predecessors in a Chord-like identifier ring [4].

  Another method is to eliminate the need for overlay routing. For example, GHT and CHR employ geographical routing (GPSR), which requires local exchange of node locations only. No flooding for route discovery (as in reactive routing) or of topology information (as in proactive routing) is needed.

7 Conclusions

We introduced in this chapter the distributed hash table (DHT), a scalable, efficient, self-organizing and robust peer-to-peer routing infrastructure. Data is inserted into the DHT in the form of key-value pairs, where the key is an identifier uniquely distinguishing the data. DHT nodes store these key-value pairs and conduct efficient data lookup using a fully distributed algorithm. Scalable routing is realized by limiting lookups to $O(\log N)$ hops, where N is the number of nodes in the DHT. Important extensions such as locality-aware neighbor selection and key replication reduces response time, improves robustness against node dynamics and provides load balancing.

Several applications based on DHTs are presented, such as distributed storage systems (CFS), application-layer multicast (Scribe), peer-to-peer video-on-demand with rich user interactivity (VMesh) and a framework for general routing services (*i*3). These applications exploit the benefits provided by the underlying DHT for large-scale operation in the Internet.

DHT deployment in wireless networks is also discussed. The characteristics of wireless networks, such as bandwidth scarcity and limited node processing capacity, make it difficult to apply DHT schemes directly. Two main issues, namely the overlay mismatch problem and excessive maintenance overhead, hinder DHT deployment in such networks. Various approaches have been highlighted to address theses issues.

The role of DHTs in future networks is an open issue. While we have focused on DHT design on wired networks and wireless networks, the future of networking is most probably a hybrid of both. With millions of mobile devices participating in networked applications across the Internet, it is interesting to see how DHTs can be employed to achieve the required scalability and communications latency, while keeping up with high degrees of node dynamics.

References

1. Internet Domain Survey. www.isc.org/ds
2. Araujo, F., Rodrigues, L., Kaiser, J., Liu, C., Mitidieri, C.: CHR: A distributed hash table for wireless ad hoc networks. In: Proc. of IEEE Distributed Computing Systems Workshops (ICDCSW), pp. 407–413 (2005)
3. Burresi, S., Canali, C., Renda, M.E., Santi, P.: MeshChord: A location-aware, cross-layer specialization of Chord for wireless mesh networks (concise contribution). Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on pp. 206–212 (2008)
4. Caesar, M., Castro, M., Nightingale, E.B., O'Shea, G., Rowstron, A.: Virtual ring routing: Network routing inspired by DHTs. SIGCOMM Computer Communication Review **36**(4), 351–362 (2006)
5. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.: Scribe: a large-scale and decentralized application-level multicast infrastructure. Selected Areas in Communications, IEEE Journal **20**(8), 1489–1499 (2002)
6. Clausen, T., Jacquet, P.: Optimized link state routing protocol. In: IETF RFC 3626 (2003)
7. Dabek, F., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with CFS. In: SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles, pp. 202–215. ACM, New York, NY, USA (2001)
8. Galluccio, L., Morabito, G., Palazzo, S., Pellegrini, M., Renda, M.E., Santi, P.: Georoy: A location-aware enhancement to Viceroy peer-to-peer algorithm. Computer Network **51**(8), 1998–2014 (2007)
9. Gummadi, K., Gummadi, R., Gribble, S., Ratnasamy, S., Shenker, S., Stoica, I.: The impact of dht routing geometry on resilience and proximity. In: SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 381–394. ACM, New York, NY, USA (2003)
10. Johnson, D., Hu, Y., Maltz, D.: The dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4. In: IETF RFC 4728 (2007)
11. Karp, B., Kung, H.T.: GPSR: greedy perimeter stateless routing for wireless networks. In: MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking, pp. 243–254. ACM, New York, NY, USA (2000)

12. Khashmir: http://khashmir.sourceforge.net
13. Lakshminarayanan, K., Rao, A., Stoica, I., Shenker, S.: Flexible and robust large scale multicast using i3. Tech. Rep. CS-02, University of California, Berkeley (2002)
14. Lakshminarayanan, K., Stoica, I., Wehrle, K.: Support for service composition in i3. In: MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia, pp. 108–111. ACM, New York, NY, USA (2004)
15. Malkhi, D., Naor, M., Ratajczak, D.: Viceroy: A scalable and dynamic emulation of the butterfly. In: Proceedings of the 21st annual ACM symposium on Principles of distributed computing (2002)
16. Maymounkov, P., Mazires, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS), pp. 53–65 (2002)
17. Mehyar, M., Spanos, D., Pongsajapan, J., Low, S.H., Murray, R.M.: Asynchronous distributed averaging on communication networks. IEEE/ACM Transactions on Networking **15**(3), 512–520 (2007)
18. Perkins, C., Royer, E.: Ad-hoc on-demand distance vector routing. Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on pp. 90–100 (1999)
19. Pucha, H., Das, S.M., Hu, Y.: Ekta: An efficient DHT substrate for distributed applications in mobile ad hoc networks. In: Proc. of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), pp. 163–173. IEEE (2004)
20. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 161–172. ACM, New York, NY, USA (2001)
21. Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Yin, L., Yu, F.: Data-centric storage in sensornets with GHT, a geographic hash table. Mobile Networks and Applications **8**(4), 427–442 (2003)
22. Rowstron, A.I.T., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, pp. 329–350. Springer-Verlag (2001)
23. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 149–160. ACM, New York, NY, USA (2001)
24. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.C.: Chord: A scalable peer-to-peer lookup service for internet applications. Tech. Rep. TR819, Laboratory for Compuer Science, Massachuseets Institute of Technology (2001)
25. Vuze: http://wiki.vuze.com/index.php/distributed_hash_table
26. Xu, J.: On the fundamental tradeoffs between routing table size and network diameter in peer-to-peer networks. INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE **3**, 2177–2187 (2003)
27. Yiu, W.P., Jin, X., Chan, S.H.: VMesh: Distributed segment storage for peer-to-peer interactive video streaming. Selected Areas in Communications, IEEE Journal **25**(9), 1717–1731 (2007)
28. Zahn, T., Schiller, J.: MADPastry: A DHT substrate for practicably sized MANETs. In: Proc. of IEEE Workshop on Applications and Services in Wireless Networks (ASWN) (2005)
29. Zhao, B., Huang, L., Stribling, J., Rhea, S., Joseph, A., Kubiatowicz, J.: Tapestry: a resilient global-scale overlay for service deployment. Selected Areas in Communications, IEEE Journal **22**(1), 41–53 (2004)
30. Zhuang, S., Lai, K., Stoica, I., Katz, R., Shenker, S.: Host mobility using an internet indirection infrastructure. Wireless Networks **11**(6), 741–756 (2005)

The Gamut of Bootstrapping Mechanisms for Structured Overlay Networks

Anwitaman Datta

Abstract Structured overlays are an important primitive in building various peer-to-peer (P2P) systems, and is used for various functions including address independent end-to-end routing, managing multicast groups, indexing of content in a decentralized environment and P2P storage, among others. While they operate in a decentralized manner, and the self-stabilizing mechanisms to maintain the overlays are also decentralized, bootstrapping structured overlays have traditionally assumed implicit centralization and/or coordination. In this chapter, we provide a survey of different flavors of structured overlay construction mechanisms – including quasi-sequential mechanisms which are predominantly in use, followed by parallelized approaches, and finally looking into how two isolated overlay can be merged, which is key to decentralized bootstrapping.

1 Introduction

In recent years the concept of structured overlays[1] has attracted a lot of attention because of its potential to become a generic substrate for internet scale applications – used for applications as diverse as locating resources in a wide area network in a decentralized manner, address independent robust and flexible (group) communication – e.g., application layer multicast and internet indirection infrastructure and content distribution network to name a few.

The basic function of the structured overlay is to act as a decentralized index. To that end, for each resource, a globally unique identifier (called the key) is generated using some function suitable to the applications that are supposed to use the index.

Anwitaman Datta
NTU, Nanyang Avenue, Singapore, e-mail: anwitaman@ntu.edu.sg

[1] A special class of structured overlays are the *distributed hash tables* (DHTs), where the keys are generated from the resources (name or content) using uniform hashing, e.g., SHA-1 (Secure Hash Algorithm [17]).

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_11, © Springer Science+Business Media, LLC 2010

The codomain (loosely speaking, range) of this function is called the key-space. For example, the key-space may be the unit interval $[0,1]$ or an unit circle $[0,1)$, so that the keys can be any real number between 0 and 1. The key-value pair is stored at peers responsible for the particular key. Efficient search of keys based on decentralized routing helps the applications to access the resource itself in absence of central coordination or global knowledge.

Definition 1. Structured overlay networks comprise of the following principal ingredients:
(i) Partitioning of the key-space (say an interval or circle representing the real number between the range $[0,1]$) among peers, so that each peer is responsible for a specific key space partition. By being responsible, we mean that a peer responsible for a particular key-space partition should have all the resources (or pointers) which are mapped into keys which are in the respective key-space partition.[2]
(ii) A graph embedding/topology among these partitions (or peers) which ensures full connectivity of the partitions, desirably even under churn (peer membership dynamics), so that any partition can be reached from any partition to any other – reliably and preferably, efficiently.
(iii) A routing algorithm which enables the traversal of messages (query forwarding), in order to complete specific search requests (for keys).

Definition 2. A structured overlay network thus needs to meet two goals to be functionally correct:
(i) *Correctness of routing*: Starting from any peer, it should be possible to reach the correct peer(s) which are responsible for a specific resource (key).
(ii) *Correctness and completeness of keys-to-peers binding*: Any and all peers responsible for a particular key-space partition should have all the corresponding keys/values.

Various applications can use transparently the (dynamic) binding between peers and their corresponding key-space partitions as provided by the overlay for resource discovery and communication purposes in a wide area network.

One of the most important and distinguishing aspect of structured overlays is the peers' interconnection – the topology/geometry of the network.

How this topology is established in a dynamic setting, and whether it achieves some other properties (like – proximity and low stretch exploiting information from the underlying networking layer, load-balancing, security against various attacks, etcetera.) and how the invariants of the topology maintained over time in presence of membership dynamics and attacks are some of the most interesting questions that have been investigated in the P2P research community in these last years.

[2] It is also possible that keys are not strictly associated with a specific peer and instead have a looser coupling. For example, in Freenet [9] and FuzzyNet[15], this association of keys to peers can be thought to be in a best effort fashion, such that instead of choosing the peer which is globally the closest to a key, the locally closest peer is delegated the responsibility of the key. Such systems are also called semi-structured overlays.

In this article, we focus on the issue of how these topologies can be established (bootstrapped) in a dynamic setting while also meeting some other desirable properties like load-balancing. We will survey different kinds of bootstrapping mechanisms, including traditional quasi-sequential approaches, as well as subsequent parallelized approaches, and also looking at how two isolated overlays can be merged to build a larger network, paving way for decentralized boot strapping.

The existing literature presenting various bootstrapping mechanisms vary in rigor and details. To keep the presentation uniform as well as accessible to the general audience, we provide only a high level summary of the concepts.

2 A Taxonomy of Structured Overlay Topologies

The specific details of the topologies is crucial to the exposition of how these topologies can be achieved. We briefly look into some of the important structured overlay topologies, particularly the *ring* and *tree* topologies. These are not necessarily optimal in the sense of achieving the smallest routing table size or smallest diameter, however have proven to be practical because of their overall characteristics. They have moderately small average routing table sizes which provide good resilience at reasonable maintenance cost, small diameter, good degree-distribution (necessary for congestion-free and load-balanced routing) and flexibility to deal with different kind of workloads, and last but not the least, they are also relatively simple. The complexity of the topology plays an important role in a peer-to-peer setting, where the topology invariants need to be established and maintained without global knowledge and coordination in presence of potentially large scale in terms of both peer population as well as high membership dynamics.

2.1 Ring

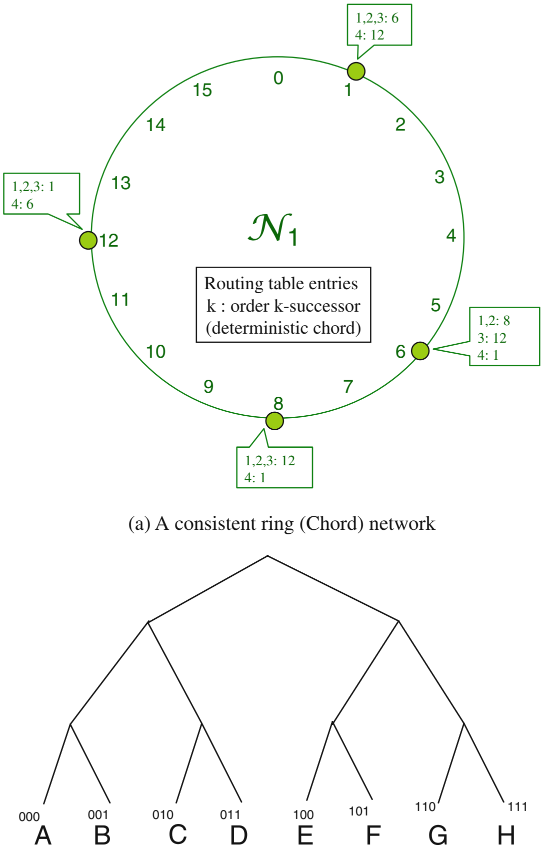
The ring based topology was pioneered in the context of overlays in the Chord [27] network. Chord uses SHA-1 based consistent hashing to generate an m-bit identifier for each peer p, which is mapped onto a circular identifier space (key-space).

Irrespective of how the peers' identifiers are generated in a ring based topology, what is essential is that the peer identifiers are distinct. Similarly, unique keys are generated corresponding to each resource. Each *key* on the key-space is mapped to the peer with the least identifier greater or equal to the *key*, and this peer is called the *key*'s successor. Thus to say, this peer is responsible for the corresponding resource.

What is relevant for our study is how keys from the key-space are associated with some peer(s) and how the peers are interconnected (in a ring) and communicate among themselves.

Definition 3. A ring network is (1) weakly stable if, for all nodes p, we have $predecessor(successor(p)) = p$; (2.a) strongly stable if, in addition, there exists no peer s on the identifier space where $p < s < q$ where $successor(p) = q$; and (2.b) loopy if it is weakly but not strongly stable.

Condition (2.a) that there exists no peer s on the identifier space where $p < s < q$ if p and q know each other as mutual successor and predecessor determines the correctness of the ring structure. Figure 1a shows one such consistent ring structure (peer's position in the ring and its routing table). The order-1 successor known also just as "successor" of each peer is the peer closest (clock-wise) on the key-space.



(a) A consistent ring (Chord) network

(b) A tree based (P-Grid) network. The actual topology has no hierarchy as shown in Figure 2.

Fig. 1 Some structured overlay topologies

If at any time such a s joins the system, the successor and predecessor information needs to be corrected at each of p, q and s. Maintaining the ring is basically to maintain the correctness of successors for all peers – this in turn provides the functional correctness of the overlay routing – i.e., successor peer for any identifier *key* can be reached from any other peer in the system (by traversing the ring). For redundancy, f_s consecutive successors of each peer are typically maintained, so that the ring invariant is violated only when all f_s consecutive peers of any peer depart the system even before a ring maintenance mechanism like Chord's self-stabilization algorithm can react and repopulate with the correct successor entries.

In addition to the successor/predecessor information, each peer maintains routing information to some other distant peers in order to reduce the communication cost and latency.

It is the way these long ranges are chosen which differ in many ring topology networks and has no critical impact on the functional correctness of the overlay. Distance in such ring based topologies is generally measured in terms of the absolute difference of the two concerned points on the key-space, but other metrics can as well be used. For the real topology, devoid of the artificial distance metrics, the long ranges are essentially to halve the number of peers (the "true" distance on a ring traversed sequentially) between the current peer and the destination peer [14].

Explicitly or implicitly, most variants of the ring topology exploit this fact and reduce the distance geometrically – either deterministically or probabilistically. The original Chord proposal advocated the deterministic use of the successor of the identifier $(p+2^{k-1})$ modulo 2^m as an order-k successor of peer p or a finger table entry. Many other variants for choosing the long range links exist – e.g., randomized choice from the interval $[p+2^{k-1}, p+2^k)$ or exploiting small-world [20] topology or emulating skip graphs.

The maintenance of the ring (strong stability) is critical for functional correctness of the routing process in ring based topologies. The self-stabilization mechanisms proposed in the original Chord proposal [27] exhaustively deals with the maintenance of the ring, and all other ring based topologies rely on similar mechanisms. It has been shown that the ring topology has better static resilience than other topologies because of the greater flexibility to choose both routing table entries to instantiate the overlay, as well as to choose from multiple routes to forward a query at run time.

2.1.1 Ring Self-Stabilization Highlights

The ring invariant is typically violated when new peers join the network, or existing ones leave it. If such events occur simultaneously at disjoint parts of the ring, the ring invariant can easily be reestablished using local interactions among the affected peers. Note that these events do not lead to a loopy state of the network.

Apart looking into the simple violations of the ring invariant which are relatively easily solved, the original Chord proposal (technical report version) also provides mechanisms to arrive at a strongly stable network starting from a loopy network (whichsoever reason such a loopy state is reached). We summarize the results of stabilizing a loopy network here.

Any connected ring network with N peers becomes strongly stable within $O(N^2)$ rounds of strong stabilization if no new membership changes occur in the system. Starting from an arbitrary connected state with successor lists of length $O(logN)$ if the failures rate is such that at most $N/2$ nodes fail in $\Omega(logN)$ steps then, whp, in $O(N^3)$ rounds, the network is strongly stable.

2.2 Tree

Arguably the earliest approach to locate objects in a distributed environment – the PRR scheme proposed by Plaxton et al. [25] used a tree structure where searches were forwarded based on longest prefix matching. Tapestry [28], Pastry [26] (also uses the ring as a fall-back mechanism) and P-Grid [1] shown in Fig. 1b among others uses similar prefix resolution in order to forward search operations, and has the tree topology. The leaf-nodes of the tree represent the key-space partitions (peers). The (maximum) distance between these partitions when the query is resolved based on prefix is then the height of the common subtree. Kademlia [22] resembles the tree structure and peers have the same routing choices as other tree-based networks. Despite having the same topology, Kademlia routing uses the XOR distance between the peer identifiers (essentially the binary string representing the node's path in the tree) instead of resolving common prefix.

Note that this also exemplifies the essential orthogonality of the topology itself from the routing strategy – the same graph connectivity may be explored based on different routing schemes, and thus defined as separate ingredients of a structured overlay network in Definition 1.

2.2.1 The P-Grid Overlay

Some of the concepts in this article will be illustrated using examples of the P-Grid network, thus we next provide a formal description of P-Grid. Figure 1b shows the tree abstraction and Fig. 2 shows one possible instance of peers' connections.

P-Grid divides the key-space in mutually exclusive partitions so that the partitions may be represented as a prefix-free set $\Pi \subseteq \{0,1\}^{\cdot}$. Stored data items are identified by keys in $\mathscr{K} \subseteq \{0,1\}^{\cdot}$. We assume that all keys have length that is at least the maximal length of the elements in Π, i.e.,

$$\min_{k \in \mathscr{K}} |k| \geq \max_{\pi \in \Pi} |\pi| = \pi_{max}$$

Each key belongs uniquely to one partition because of the fact that the partitions are mutually exclusive, that is, different elements in Π are not in a prefix relationship, and thus define a radix-exchange trie.

$$\pi, \pi' \in \Pi \Rightarrow \pi \nsubseteq \pi' \wedge \pi' \nsubseteq \pi$$

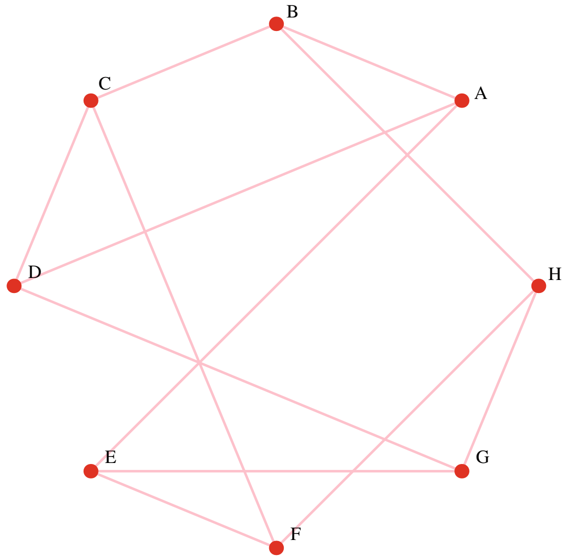


Fig. 2 The actual P-Grid connectivity graph does not have any hierarchy. The routes are randomly chosen from complimentary sub-trees of Fig. 1b. The basic P-Grid graph is directional, however since each link establishment and maintenance cost is the same, and from the symmetry of the routing choices, the actual P-Grid uses bidirectional routes

where $\pi \subseteq \pi'$ denotes the prefix relationship. These partitions also exhaust the key-space, so to say, the key-space is completely covered by these partitions so that each key belongs to one and only one (because of exclusivity) partition.

In P-Grid each peer $p \in P$ is associated with a leaf of the binary tree, and each leaf has at-least one peer associated to itself. Each leaf corresponds to a binary string $\pi \in \Pi$, also called the *key-space partition*. Thus each peer p is associated with a path $\pi(p)$. For search, the peer stores for each prefix $\pi(p,l)$ of $\pi(p)$ of length l a set of references $\rho(p,l)$ to peers q with property $\overline{\pi(p,l)} = \pi(q,l)$, where $\overline{\pi}$ is the binary string π with the last bit inverted. This means that at each level of the tree the peer has references to some other peers that do not pertain to the peer's subtree at that level which enables the implementation of prefix routing for efficient search. The whole routing table at peer p is then represented as $\rho(p)$ Moreover, the actual instance of the P-Grid is determined by the randomized choices made at each peer for each level out of a much larger combination of choices. The cost for storing the references and the associated maintenance cost scale as they are bounded by the depth of the underlying binary tree. This also bounds the search time and communication cost.

Each peer stores a set of data items $\delta(p)$. For $d \in \delta(p)$ the binary key $\kappa(d)$ is calculated using an order-preserving hash function. $\kappa(d)$ has $\pi(p)$ as prefix but it is not excluded that temporarily also other data items are stored at a peer, that is, the set $\delta(p,\pi(p))$ of data items whose key matches $\pi(p)$ can be a proper subset of $\delta(p)$. Moreover, for fault-tolerance, query load-balancing and hot-spot handling, multiple peers are associated with the same key-space partition (*structural replication*). $\Re(\kappa)$ represents the set of peers replicating the object corresponding to key κ. Peers additionally also maintain references to peers with the same path, i.e., their replicas $\Re(\pi(p))$, and use epidemic algorithms to maintain replica consistency. Routing in P-Grid is greedy, and based on matching the longest prefix, similar to the PRR scheme [25].

There are many other topologies derived from interconnection networks that have been adapted for P2P settings, but the rest of the chapter will focus on overlays based on the prevalent ring and tree topologies.

3 Quasi-Sequential Construction of Overlays

The construction of overlays, from the early days have focussed not only in establishing a logically correct topology, but also on how to ensure that load distribution across the peers is uniform.

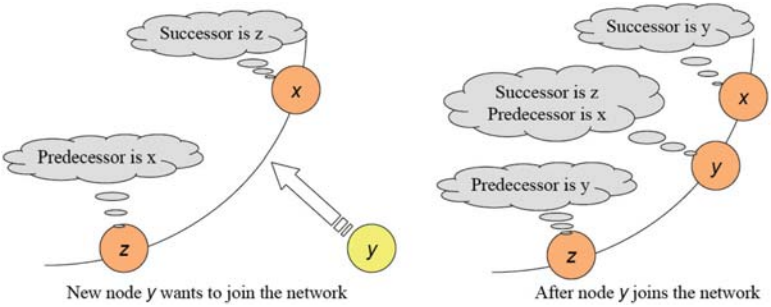
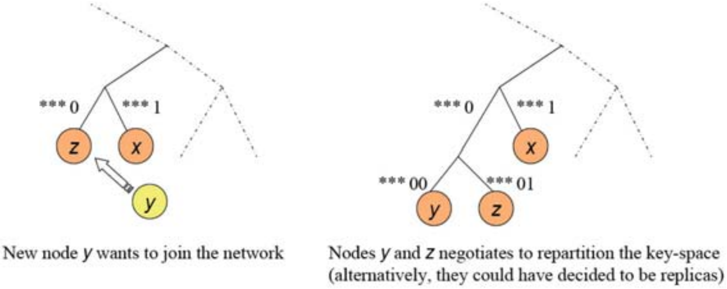


Fig. 3 A new node joining a Chord network. Nodes x and z which are originally neighbors (predecessor and successor of each other) need to update their local information to register that y is the new neighbor

The original approach to construct most structured overlays assumed an incremental approach, investigating how new nodes can continuously join an existing network, or how the network manages continuous but gradual departures of existing nodes – called churn in the network.

Churn in the network requires reallocation of the part of the key space that a peer is responsible for, as well as rewiring of the connections, for instance in the ring based networks, ensuring that the ring integrity is maintained is most important. Ring integrity is ensured by making sure that nodes maintain the right successor (and predecessor) lists despite churn. To ensure this, it is important to allow only one node to join between a node and its immediate successor at any given time. Figure 3 depicts such a scenario. Such a scheme makes an implicit assumption that nodes join in the same part of the key-space in a sequence. Of-course nodes can join mutually disjoint parts of the key-space simultaneously. Essentially such a model can thus support quasi-sequential join of peers in the network.



*** 0
*** 1
z
x
y
New node y wants to join the network
*** 0
*** 1
x
*** 00
*** 01
y
z
Nodes y and z negotiates to repartition the key-space
(alternatively, they could have decided to be replicas)

Fig. 4 A new node *y* joining a P-Grid network by redividing the responsibility with existing peer *z*. They also need to update their routing tables, so that, for example, if *z* receives a query with prefix $\cdots 00$ it can forward it to *y*. Other peers, for example *x* do not need to update any information, and can continue to forward all queries with prefix $\cdots 0$ still to *z*

In the case of an overlay like P-Grid abstracting the tree structure, the sequential overlay construction will require the newly joining peer to negotiate with an existing peer to redivide key-space partition responsibility (as shown in Fig. 4) or alternatively decide to become mutual replicas.[3]

3.1 Load-Balancing Considerations

In order to achieve load-balancing, randomized strategies are often used in such a setting. Each new node joining the system picks a random point uniformly on the key-space, and joins the network by splitting that part of the key-space with whichever peer has been originally responsible for that part of the key-space. Uniformly choosing the part of the key-space to join the network was assumed to evenly partition the key-space. While not highlighted as such, such a randomized strategy coincidentally makes sure that different peers joining at the same time join different parts of the network, doing so in an almost trivial way without any need for global knowledge or coordination.

Uniform random choice of peer's position in the key-space however leads to a relatively high variation in the size of key-space partitions for which individual

[3] Replication is necessary for both load-sharing and fault-tolerance, but has not been shown in this specific example. Structurally replicated P-Grid networks will be shown later in Fig. 12.

peers are responsible for. A lot of work has been done to reduce such variation, and to make the load distribution among peers more uniform. Included among those approaches is a simple but very effective randomized strategy "power of two choices", where each node would choose two (or several) random points on the key-space, and then join the network in the points with most load [12]. Such an approach is a variation of the power of two choices originally proposed by Byers et al. [7] in the context of peer-to-peer systems, where multiple hash values were used to generate multiple keys for a resource, and then stored at the least loaded peer.

For various reasons, explained in the next section, it may be desirable or even necessary to construct an overlay in a parallelized manner, a departure from the original quasi-sequential approach.

4 Parallelized Construction of Overlays

Fast construction of structured overlays has been motivated by several reasons. Building such a network rapidly from scratch enables fast recovery from catastrophic failures as well as easy deployment of such a network on demand, which can be used to perform tasks required by more complex distributed systems, as well as eliminate or at least complement complex and expensive overlay maintenance algorithms, allowing for recreation of the whole network instead. From the perspective of considering the structured overlay as a distributed index structure, building an overlay from scratch can be considered to be analogous to (re-)indexing content.

Further motivation: A data-management perspective

In standard database systems it is common practice to regularly (re-)index attributes to meet changing requirements and optimize search performance. Structured peer-to-peer overlay networks are increasingly used as an access structure for highly distributed data-oriented applications, such as relational query processing, metadata search or information retrieval [4, 24]. Structured overlays' use was motivated by the presence of certain features that are supported by their design such as scalability, decentralized maintenance, and robustness under network churn. Compared to unstructured overlay networks which are also being proposed for these applications [16, 21], structured overlay networks additionally exhibit much lower bandwidth consumption for search as well as guarantee completeness[4] for search results.

The standard maintenance model for peer-to-peer overlay networks assumes a dynamic group of peers forming a network where peers can join and leave, essentially in a sequential manner, as elaborated in the previous section. In addition proactive or reactive maintenance schemes are used to repair inconsistencies resulting from node and network failures or to re-balance load in order to react to data

[4] In terms of information retrieval terminology, recall = 1.

updates. These approaches to maintenance, that have been extensively studied in the literature, correspond essentially to updating database index structures in reaction to updates.

In data-oriented applications resources may be identified by dynamically changing predicates. Multiple overlay networks can be needed simultaneously, each of them supporting a specific addressing need. One can illustrate these requirements by a typical application case of peer-to-peer information retrieval.

The standard application of structured overlay networks in peer-to-peer information retrieval is the implementation of a distributed inverted file structure for efficient keyword based search. In this scenario, several situations occur, in which the overlay network has to be constructed from scratch:

- A set of documents that is distributed among (a potentially very large number) of peers is identified as holding information pertaining to a common topic. To support efficient retrieval for this specific document collection, a dedicated overlay network implementing inverted file access may have to be set up.
- A new indexing method, for example, a new text extraction function for identifying semantically relevant keywords or phrases, is being used to search a set of semantically related documents distributed among a large set of peers. Since the index keys change as a result of changing the indexing method a new overlay network needs to be constructed to support efficient access.
- Due to updates to a distributed document collection an existing distributed inverted file has become obsolete. This may either result from not maintaining the inverted file during document updates or due to changing characteristics of the global vocabulary and thus changing the indexing strategy (e.g., term selection based on inverse document frequency). Thus a complete reconstruction of the overlay network is required.
- Due to catastrophic network failures the standard maintenance mechanisms no longer can reconstruct a consistent overlay network. Thus the overlay networks needs to be constructed from scratch. Of course, this scenario applies generally in any application, but becomes more probable when multiple overlay networks are deployed in parallel.

In principle a (re-)construction of an overlay network in any of these scenarios can be achieved by the standard maintenance model of sequential node joins and leaves. However, this approach encounters two serious problems:

- The peer community will have to decide on a serialization of the process, e.g., electing a peer to initiate the process. Thus the peer community has to solve a leader election problem, which might turn out to be unsolvable for very large peer populations without making strong assumptions on coordination or limiting peer autonomy.
- Since the process is performed essentially in a serialized manner, it incurs a substantial latency. In particular it does not take any advantage of potential parallelization, which would be a natural approach.

These motivate a fundamentally different approach, that of parallelized overlay construction. Several such mechanisms including [2, 5, 18] have been proposed in the literature, which we summarize next.

4.1 Sorting Peer-IDs as a Mechanism to Build a Ring

To achieve a strongly stable ring (Definition 3) it is necessary that all nodes know the correct immediate neighbors (successor and predecessor). This leads to a sorting of the peers according to their identifiers or responsibility of the parts of the key-space. Achieving this without the global knowledge of which all peers are in the network, and without any central coordination of peers to choose the correct neighbors is a crucial challenge in constructing an overlay. So the problem of constructing a ring based structured overlay essentially boils down to the problem of decentralized sorting of the peers according to their identifiers. Now we discuss one rigorous and one heuristic mechanism to sort peer-IDs.

4.1.1 Pairing and Merging Virtual Trees

Angluin et al. [5] considers the problem of constructing a structured overlay as that of creating a linked list of nodes sorted according to their identifiers, which can then be used as the basis for constructing the essential ring of the system. Long range links useful for optimization can be wired subsequently to make the network routing efficient.

They assume that any node in the system can communicate with any other node, once the nodes become aware of each other, which defines a “knowledge graph”. Initially, this knowledge graph is assumed to be a weakly connected degree bounded directed (random) graph. That is to say, each node knows a fixed number of random nodes.

The first step in this approach is to pair nodes using a randomized mechanism. Based on the original knowledge graph connectivity, every node probes all potential successors. The recipient of such a probe either accepts (the first received probe) or rejects to be paired. Ideally, after a round of such pairing is finished, the paired nodes can behave as a virtual “supernode” to conduct further pairing. Note that such a pairing of virtual supernodes which are composite of previously paired nodes, require a merging mechanism. The virtual supernodes are maintained as Patricia trees, which facilitates the use of tree merging algorithms, and finally provides a single supernode comprising of all the nodes sorted also as a list.

Figure 5 shows a step of their mechanism as a toy example. Nodes 1 and 7 pair up, and act as a virtual node, as do nodes 5 and 9. Subsequently these two virtual nodes (comprising two nodes each), merge to form a single virtual node. The implicit tree representation allows both efficient mergers, as well as provides readily a tree structure, apart from sorting the peer identifiers.

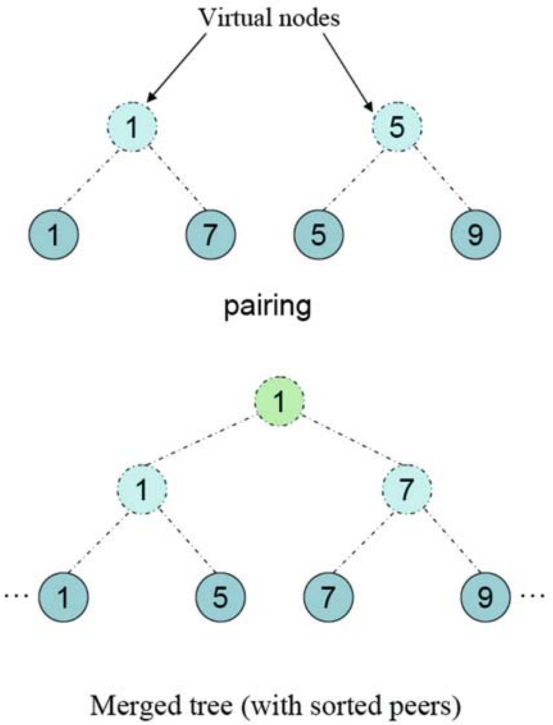


Fig. 5 Sorting based on iterations of pairing and merging [5]. Once the peer identifiers are sorted, a ring structure can readily be obtained. Essentially, what is necessary in a decentralized setting to achieve such a sorted "linked list" is that all peers know the correct predecessors and successors (multiple entries are useful for fault tolerance)

The main complexities of this approach in a real life scenario (asynchronous setting) are in determining and phasing the pairing rounds and the merging process, and avoiding live-locks. In its original form, the algorithm also could not tolerate any node departure during the overlay construction.

4.1.2 Gossip Based Mechanism

Jelasity et al. [18, 19, 23] proposed a gossip based approach to bootstrap overlays. They assume that each node can obtain a random subset of the participating peers. This assumption is similar to the idea of knowledge graph assumed by Angluin et al. [5].

Each node maintains a constant sized leaf-set, comprising of the nearest nodes to itself in both direction – termed as predecessors and successors. Ideally, equal

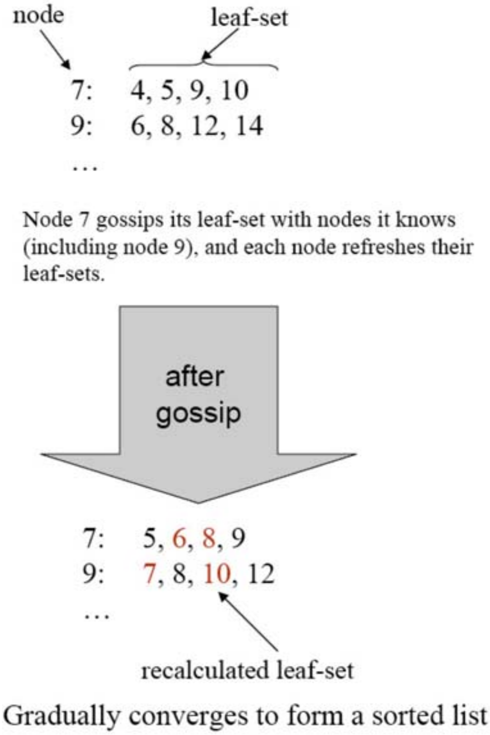

node
leaf-set
7: 4, 5, 9, 10
9: 6, 8, 12, 14
...
Node 7 gossips its leaf-set with nodes it knows (including node 9), and each node refreshes their leaf-sets.
after gossip
7: 5, 6, 8, 9
9: 7, 8, 10, 12
...
recalculated leaf-set
Gradually converges to form a sorted list

Fig. 6 Sorting based on iterations of gossips [18, 19, 23].

number of predecessors and successors are maintained, but if a node knows fewer of one kind, then it fills the space with nodes of the other category. Such information originally comes from the random subset of participating peers that each node gets, and eventually is derived from the information that nodes gossip among each other.

The essential idea is to gossip the leaf-set information, and refine it based on the information obtained from other nodes. In the toy example shown in Fig. 6, node 7 originally knows nodes $\{4,5,9,10\}$. Likewise 9 knows $\{6,8,12,14\}$. By gossiping with all the nodes 7 knows, which includes 9, 7 gets to know about 6 and 8, its immediate neighbors in this example. In contrast to the rigorous approach of Angluin et al., the gossip based approach is more a heuristic, in that while there is no formal proof of convergence, simulation based studies show it performs pretty well, and nodes get to establish the ring information with few iterations of the gossip mechanism. This mechanism can also be expected to be robust against node departures and arrivals during the process, and hence may be more practical.

4.2 Recursive Proportional Partitioning

The approach to sort peer identifiers can be considered to be a bottom-up approach, where the peer identifiers are already chosen (often randomly) and thus the way the key-space responsibility should be distributed is pre-determined. A logically top-down approach introduced by Aberer et al. [2] proposes to autonomously and recursively partition the key-space, but in a granularity adaptive to the load-distribution on the key-space (which can be arbitrarily skewed for data-oriented applications).

During the overlay construction process, two types of load balancing problems are dealt with simultaneously – the balancing of storage load among peers under skewed key distributions (i.e., number of keys per partition is balanced) and the balancing of the number of replica peers across key space partitions. The first problem is important to balance workload among peers and is solved by adapting the overlay network structure to the key distribution. The second one is important to guarantee approximately uniform availability of keys in unreliable networks where peers have potentially low availability. This is similar to a classical "balls into bins" scenario, where the key-space partitions are the bins and the peers (replicas) the balls. The extra twist, why existing solutions for balls into bins problems can't however be directly used is that the number of bins (key-space partitions) itself is dynamic.

Similar to Angluin et al. [5] and Jelasity et al. [18, 19, 23], this approach also assumes a (loosely) connected network, so that any node can communicate with any other node, and particularly uses random walkers on this network to find random peers with whom to interact. However in contrast to those approaches, where nodes already have a particular identifier, which determines the part of the key-space partition that node is responsible for, and the task is to sort the nodes to establish a sorted list, necessary to construct a ring, Aberer et al. [2] instead allows the nodes to negotiate among each other to refine the part of the key-space they will be responsible for. It leads to construction of an overlay (P-Grid) which can support prefix based routing.

The bilateral negotiations among the peers is illustrated in Fig. 7. Such interactions can lead to three possible course of actions. Two peers may decide to repartition the part of the key-space they are responsible for, or decide to become mutual replicas, or refer each other to other suitable peers to interact with. Notice that the "repartition or replicate" actions are similar in spirit to the sequential approach described earlier, but now these actions are happening in parallel for different peers, and are iterated several times thanks to the "refer" actions.

The essential idea is that all peers are originally responsible for the whole key-space. Then, they need to find a partner to decide and split responsibility – say, for the prefixes 0 and 1 respectively. The peers also need to keep track to such a peer which is responsible for the other half, so that in future a node responsible for partition 1 can forward all queries for partition 0 to a relevant peer. Thus all peers responsible for the partition 0 can repartition it for 00 and 01, and so on.

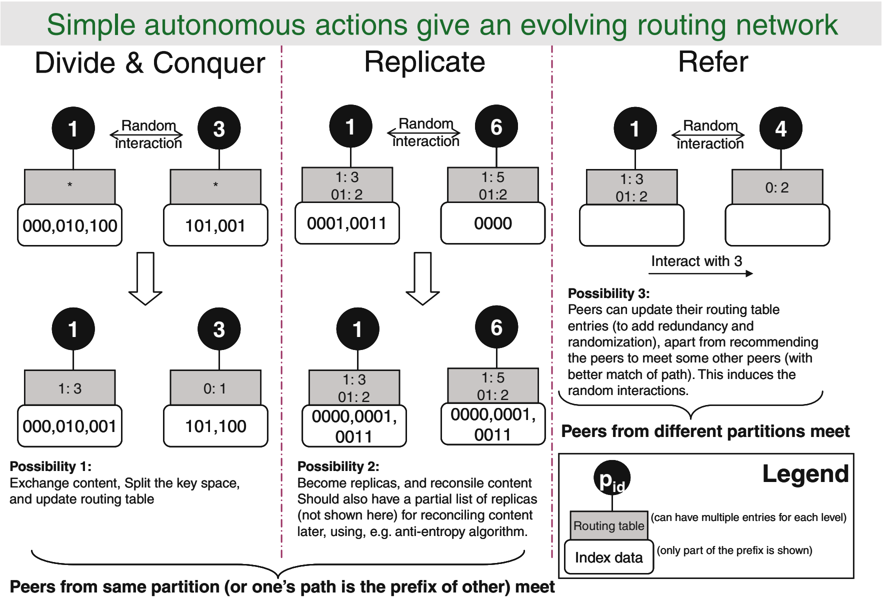

Simple autonomous actions give an evolving routing network
Divide & Conquer
Replicate
Refer
1
Random interaction
3
*
*
000,010,100
101,001
1
3
1: 3
0: 1
000,010,001
101,100
Possibility 1:
Exchange content, Split the key space, and update routing table
1
Random interaction
6
1: 3
01: 2
1: 5
01:2
0001,0011
0000
1
6
1: 3
01: 2
1: 5
01: 2
0000,0001, 0011
0000,0001, 0011
Possibility 2:
Become replicas, and reconsile content
Should also have a partial list of replicas (not shown here) for reconciling content later, using, e.g. anti-entropy algorithm.
Peers from same partition (or one's path is the prefix of other) meet
1
Random interaction
4
1: 3
01: 2
0: 2
Interact with 3
Possibility 3:
Peers can update their routing table entries (to add redundancy and randomization), apart from recommending the peers to meet some other peers (with better match of path). This induces the random interactions.
Peers from different partitions meet
Legend
p_id
Routing table
(can have multiple entries for each level)
Index data
(only part of the prefix is shown)

Fig. 7 Network evolution based on pairwise peer interactions

There are several practical problems in realizing the scheme. This includes the following.
What fraction of peers should choose a specific partition? Ideally this depends on the load in the specific half of the key-space.

For example, for the example in Fig. 8, there is three times more load for the the prefix 0 than the prefix 1, and twice the load for prefix 01 than for 00. So ideally, the peers should evolve into a network, with the partitioning of the key-space and its replication, as shown in the figure.

There are however two practical problems in realizing such an ideal partitioning. Its unrealistic to assume global knowledge like the load-skew. Even if this information were available, without global coordination, it is not possible to achieve partitioning of the network among the peers according to the granularity of the load-distribution.

Aberer et al. [2] provides some heuristics to address these problems. They propose random sampling of a subset of other peers to estimate the load-imbalance at each level of repartitioning, and using this estimate to determine parameters of a randomized algorithm to split the key-space which in expectation follows the desired (estimated) load-skew. Experiments show, that while far from perfect, the heuristics

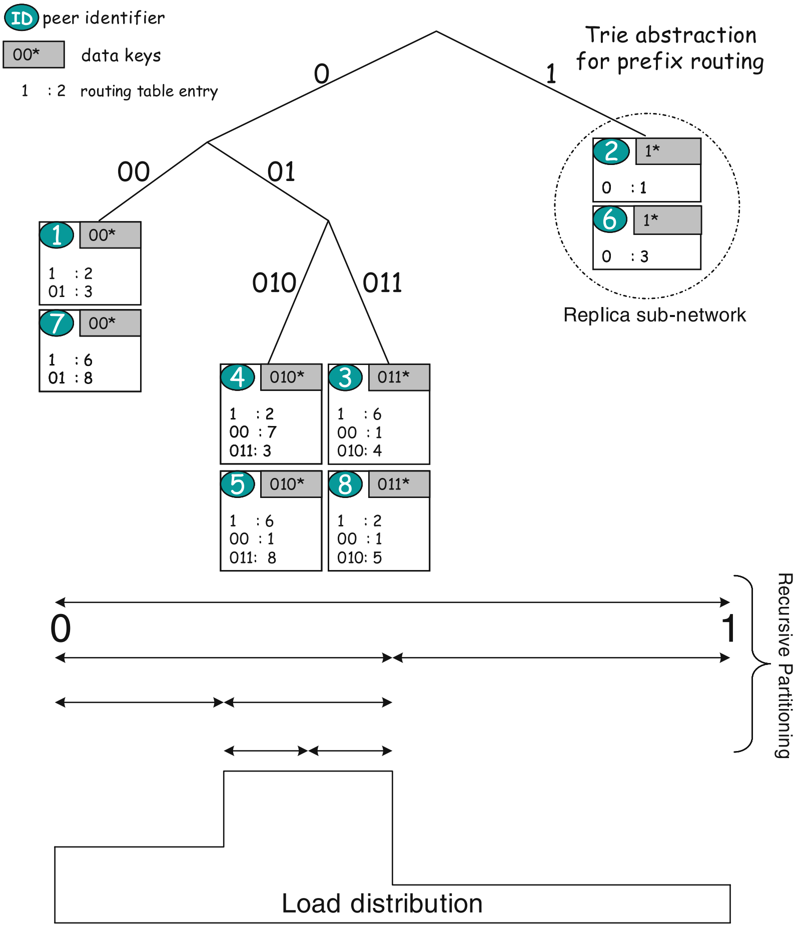


Fig. 8 P-Grid structure: Key-space is partitioned in a granularity adaptive to load-skew. In this example peers p_1 and p_7 are structural replicas for the partition for prefix 00. Peer p_1 has reference to peer p_2 for prefix 1, and to peer p_3 for prefix 01. Peer p_7 stores the same keys as peer p_1 (replicas), however they can and do have different routing table entries. In practice, for each level, each peer will also maintain multiple references primarily in order to have some fault-tolerance. Thus peer p_1 would also refer to some of p_3, p_5 or p_8 for the prefix 01

lead to a reasonably well load-balanced network construction. Other maintenance mechanisms [3] are deployed to further improve the quality of the load-balancing, but overall, this approach again achieves the primary objective, that of constructing an overlay from scratch in a parallelized manner, more or less conforming to load-skews to achieve moderate load-balance.

5 The Need and Challenges of Merging Two Similar Structured Overlays

All the parallelized construction approaches described above assumed (i) an originally connected network, and that (ii) there is one specific network to join.

Such assumptions implicitly introduces some degree of centralization. What happens if multiple isolated networks are originally constructed?

In the somewhat frantic rush to identify new cool topologies, or more practical problems like dealing with load-balancing and churn, the peer-to-peer research community has until recently ignored a fundamental and realistic problem for structured overlays that any distributed system needs to deal with – that of making two partitions of such a system merge to become one. One can speculate several reasons for such omissions in early structured overlay network research. (i) Merger of isolated overlays is trivially resolved in unstructured overlays, which is where most of the empirical information of P2P research so far has been derived from. (ii) Until recently, there has not been any real structured overlay implementations deployed and hence the problem not identified. (iii) The recent deployments and experiments have typically been under a controlled setting, where some central coordination like the use of a common set of bootstrap nodes has been used with the intention and sufficient coordination to construct only one overlay, making sure that independent and distinct overlays are not created. Moreover, none of these experiments with real implementations looked specifically for, or even accidentally, encounter network partitioning problems.

Apart network partitioning which can lead to the creation of two disjoint overlay networks there is a more likely scenario. It may so happen that disjoint overlay networks (using the same protocols) are formed over time by disjoint group of users. One may imagine that an overlay P2P network caters to a specific interest group from a particular geographic area who participate in an overlay network. At a later time, upon discovering a hitherto unknown group of like-mind users from a different part of the world, who use their own "private" network (using same protocols), these two groups may want to merge their networks in order to benefit from their mutual resources (like content or knowledge). In fact, such isolated overlay networks may result because of initial isolation of groups because of various reasons including geographic, social or administrative – a large company or country, which may originally restrict their users from interacting with outsiders in the overlay, and changes the policy at a later time – or purely because of partitioning of the physical infrastructure.

Structured overlays have often been touted as a generic substrate for other applications and services. Ideally, there will be one or few such universal overlays [8] which will be used by a plethora of other P2P applications. Realizing such an universal service too will need the possibility to merge originally isolated networks. Small overlays can be built independently, which may later be all merged together incrementally into a single overlay network.

One can thus imagine isolated islands of functional overlays catering to their individual participants. Someday, some member from one of these overlays may discover a member from another overlay. The natural thing to do then would be to merge the two originally isolated overlays into a single overlay network. In simple file-sharing networks, the motivation of doing so will be to make accessible content from both the networks to all the users. Similar conclusions can be drawn for various other conceivable applications of overlay networks.

In unstructured overlay networks (like Gnutella), merger of two originally isolated overlays happens trivially. Whichever peers from two originally isolated networks come in contact with each other need to establish mutual neighborhood relationship, and then onwards just need to forward/route messages to each other as they do with all other neighbors (Fig. 9). That's all! Likewise, hierarchical (super-peer based) unstructured overlays also merge together trivially. This is because no peer has any specific responsibility and can potentially be responsible for any and all resources in the network.

One approach to deal with multiple structured overlay networks is to let them continue to exist and operate autonomously, while allowing cross-network communication. This is typically achieved in a hierarchical fashion, where the original

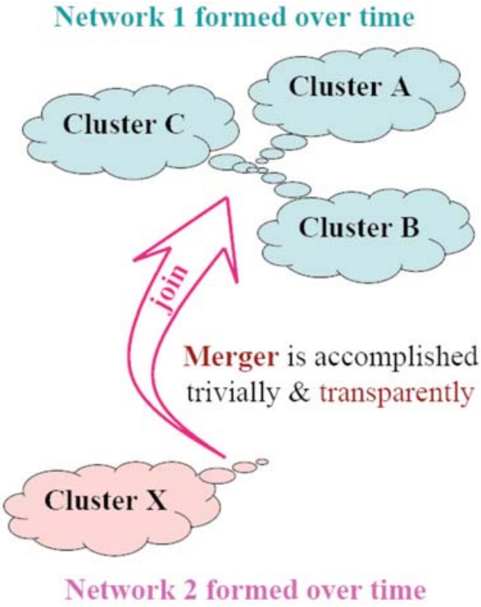
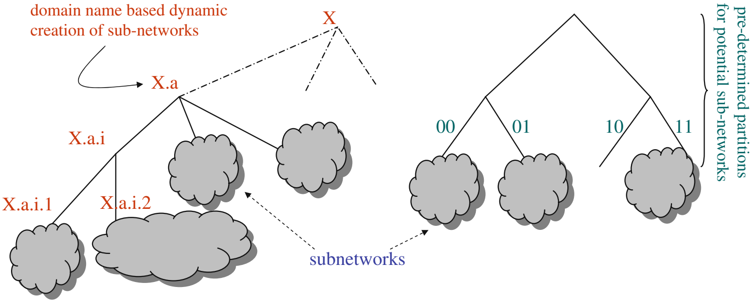

Network 1 formed over time
Cluster A
Cluster C
Cluster B
join
Merger is accomplished trivially & transparently
Cluster X
Network 2 formed over time

Fig. 9 Merger of two unstructured overlay networks (like Gnutella) is trivial. As soon as some peers from each of the two originally isolated networks establish connections to each other, a merged network is formed

networks act as “sub-networks” which are glued together to form one integrated network. Two addressing components are used to uniquely determine a peer’s role in the integrated network – one to determine which autonomous sub-network it belongs to, and the other to identify it within the sub-network. The sub-networks can be distinguished based on a domain name space [13] or by simply allocating a predetermined part of the key space for each potential sub-network [6]. Figure 10 shows example instances of such hierarchically integrated overlays. Such hierarchy allows nodes to retain communication traffic within a particular domain, or look for keys available within specific sub-networks instead of the whole integrated network. On the downside, each peer needs to keep track of more information, including keeping track of which sub-network it belongs to, at different levels of the hierarchy. For example, in a hierarchical Chord using the Canon approach each peer also needs to maintain a list of successors (and hence a ring) at every level of the hierarchy. Thus the simplicity of a completely flat address space which was a design goal in many original distributed hash tables is lost.


domain name based dynamic
creation of sub-networks
X
X.a
X.a.i
X.a.i.1
X.a.i.2
subnetworks
00
01
10
11
pre-determined partitions
for potential sub-networks

The `Canon’approach where sub-networks are distinguished by domain names. A parent domain emulates a merged network of the children sub-networks. Each individual subnetworkcan, in theory, use its own routing mechanism.

The `Cyclone’approach where inter sub-network routing is based on XOR distance between subnetworkidentifiers, while intra-subnetwork traffic uses proprietary routing mechanism of the subnetwork.

Fig. 10 Hierarchically integrated networks based on Canon [13] and Cyclone [6] approaches

An alternative to the hierarchical approaches discussed above is to device mechanisms to merge the originally isolated networks, so that a single resulting network is formed. That way peers do not need to keep track of the hierarchy, and the original overlay (DHT) design of completely flat identifier space is achieved. Some of the challenges of merging similar structured overlays, which is essential for decentralized bootstrapping of overlays [11], has been studied by Datta et al. [10], which we summarize next along with tentative solutions and shortcomings. The discussion below is also pertinent, at least in part, to hierarchical approach like Canon [13], where the authors overlook the key management issues.

5.1 Merger of Two Ring Based Networks

Consider two originally isolated Chord networks $\mathscr{N}_1$ and $\mathscr{N}_2$ with N_1 and N_2 peers respectively. An example of two such networks (superimposed) is shown in Fig. 11a. Next we will explain how such isolated networks can merge into one, since such an organic network growth is not only essential for decentralized bootstrapping, but also such a merger is necessary to ensure that the overlays retain functionality. When peers from the two different overlays meet each other (by whatsoever reason – accidentally or deliberately), in a decentralized setting there is no way for them to ascertain that they belong to two completely different systems. This is so because overlay construction always relies on such peer meetings to start with. As a consequence, if the peer pair that meets have identifiers such that they would replace their respective successor and predecessor, then they will indeed do that.

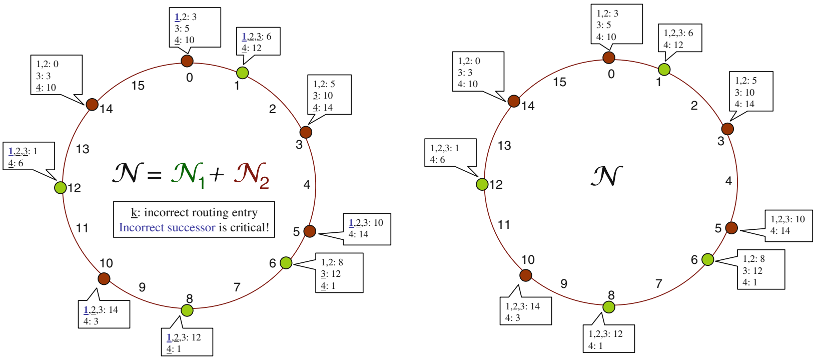


Fig. 11 When (peers from) two ring-based overlays meet

For our example from Fig. 11a lets say peer 1 from $\mathscr{N}_1$ meets peer 0 from $\mathscr{N}_2$. Then peer 1 will treat 0 as its new predecessor, and 0 will treat 1 as its new successor, instead of 12 and 3 respectively. However, if they only change their local information, then the ring network will no more be strongly stable (may in-fact not even be weakly stable). In-fact such a reconfiguration will need and lead to a cascading effect, so that all members of both the original network try to discover the appropriate immediate neighbors (successor/predecessor) – requiring coordination among all the peers.

Estimation of the probability that a peer's predecessor changes:

From the perspective of any peer in $\mathscr{N}_1$, the successor will change, if at least 1 out of the N_2 peers have identifier within the next $1/N_1$ stretch of the key-space (for which its present successor is responsible, on an average). Any particular peer from $\mathscr{N}_2$ has an identifier for this stretch with probability $1/N_1$. The number of peers falling in this stretch is thus distributed as $Binomial(N_2, 1/N_1)$, which approaches to a Poisson distribution with expectation N_2/N_1 as $N_2 \rightarrow \infty$. Hence, a peer from $\mathcal{N}_1$ will have its successor unchanged with probability $e^{-\lambda_1}$ where $\lambda_1 = N_2/N_1$. Thus each of the N_1 peers will have their successor node changed with a probability $1-e^{-\lambda_1}$ i.i.d. Peers in $\mathcal{N}_2$ will be affected similarly with a parameter $\lambda_2 = N_1/N_2$ (symmetry).

Estimate of the number of peer pairs which will have their immediate neighbors (either successor and/or predecessor) changed:

The number of peers whose successor will change in $\mathcal{N}_i$ is then distributed binomially $Binomial(N_i, 1-e^{-\lambda_i})$ for i =1,2. Hence the expected number of nodes which will need to correct their successor nodes (and predecessor nodes) is $N_1(1-e^{-\lambda_1}) + N_2(1-e^{-\lambda_2})$.

The basic idea of how the ring can be reestablished is that when two peers from different networks meet so that they replace each other's successor and predecessor (immediate neighbor), then this information needs to be communicated to the original immediate neighbors, and the process continues.

There are several combinations of how the neighborhoods of the peers are affected after their interaction, each of which needs to be accounted for the actual network merger algorithm. Moreover, different combinations of faults (single or multiple peers crashing or leaving) can happen during the ring merger, and these too need to be dealt with. The specifics of such algorithms, and evaluation of the actual ring network merger algorithm is currently underway.

Without proper and exhaustive evaluation of the exact algorithms for merging two ring networks, it is difficult to see whether a strongly stable ring can be directly achieved, or whether a sequence of faults during the merger of two rings can even lead to a loopy network, which would then require even more effort to converge to a strongly stable state using Chord's already existing self-stabilizing mechanisms.

Thus the back of the envelope analysis above just provides the expected lower-bound of the ring reestablishment process in terms of correction of successor/predecessors. The latency of such a process started because of two peers from the two networks will be $O(N_1 + N_2)$ even if there is no membership changes during the whole merger process – this is the time required to percolate the information that the ring neighborhood has changed and to discover the correct neighbor when peers from both the original networks are considered together.

5.1.1 Ring Loses Bearing During the Merge Process

Above we provided a sketch of how to only reestablish the ring topology – which only guarantees the functional correctness of the routing process – i.e., the query will be routed to the peer which is supposed to be responsible for the key-space to which the queried key belongs.

Reestablishing the ring will be necessary in order to be able to query and locate even the objects which were accessible in the original network of any individual network. Hence, such a merger operation of ring topology based overlay will typically cause a complete interruption of the overlay's functioning.[5]

5.1.2 Managing Keys on the Merged Ring

Establishing the ring in itself is however not sufficient in an overlay network based index. In order to really find all keys (which originally existed in at least one of the two networks) from any peer in the merged network, it will still be necessary to transfer the corresponding key/value data to the "possibly" different peer which has become responsible for the new overlay. To make things worse, in a ring based network the queries will be routed to the new peer which is responsible for a key, so that even after the reestablishment of the ring itself, some keys that could be found in the original networks may also not be immediately accessible, and will need to wait until the keys are moved to the new corresponding peer.

Lets consider that before the networks started merging, network $\mathcal{N}_i$ had key set $\mathcal{D}_i$ such that $|\mathcal{D}_i| = D_i$. Furthermore, if we consider that α fraction of the keys in the two networks is exclusive, that is $|\mathcal{D}_1 \cap \mathcal{D}_2| = \alpha|\mathcal{D}_1 \cup \mathcal{D}_2|$, then on an average, if a $\mathcal{N}_i$ node's successor changes, it will be necessary to transfer on an average α fraction of the data from network $\mathcal{N}_j$'s $\frac{1}{N_1+N_2}$ stretch of the key-space. Thus, on an average, the minimum[6] required transfer of unique data from members of original networks $\mathcal{N}_j$ to $\mathcal{N}_i$ will be $D_{tx}^{j \to i} = N_1(1-e^{-\lambda_1})\alpha\frac{D_2}{N_1+N_2}$.

Apart from assigning the data corresponding to a key on the key-space to the peer which is the successor for that key, ring based topologies provide fault-tolerance by replicating the same data at f_s consecutive peers on the ring.[7] Given the strict choice of f_s as neighborhood changes, the transferred data will in-fact have to be replicated at the precise f_s consecutive peers of the merged network, determining the actual minimal bandwidth consumption. Similarly, some of the original f_s replicas will need to discard the originally replicated content.

[5] Note that such a vulnerability may expose ring topologies to a new kind of "throwing rings into the ring" distributed denial of service (DDoS) attack, though the implications of such an attack and the amount of resources an adversary will require to make such a DDoS attack needs to be studied in greater detail.

[6] The actual implementation of such a data transfer will need to identify the distinct data in the two networks and transfer only the non-intersecting one, in order to achieve this minimal effort. This is an orthogonal but important practical issue that any implementation will need to look into.

[7] The parameter f_s is a predetermined global constant determined by the system designer.

5.2 Merger of Two Structurally Replicated P-Grid Networks

In the P-Grid network, replication is achieved in a very different manner than in a ring based overlay. Multiple peers are responsible for (by replicating) precisely the same exclusive key-space partition. This is called structural replication, and is in contrast to the ring based approach, where the replication is done along something like a sliding window, at the next f_s peers on the ring. Effectively structural replication can be thought of as multiple virtual instances of the same key-space partitioned network superimposed with each other. For example, in Fig. 12a the network $\mathscr{N}_2$ can be thought of composed itself of superimposition of two networks, the first say comprising of nodes U, W and X, while the other nodes belonging to the other virtual network. The routing reference maintained by the peers are intertwined, and does not need to discern the virtual networks, and is indeed desirable to achieve greater interconnection of the routing network.

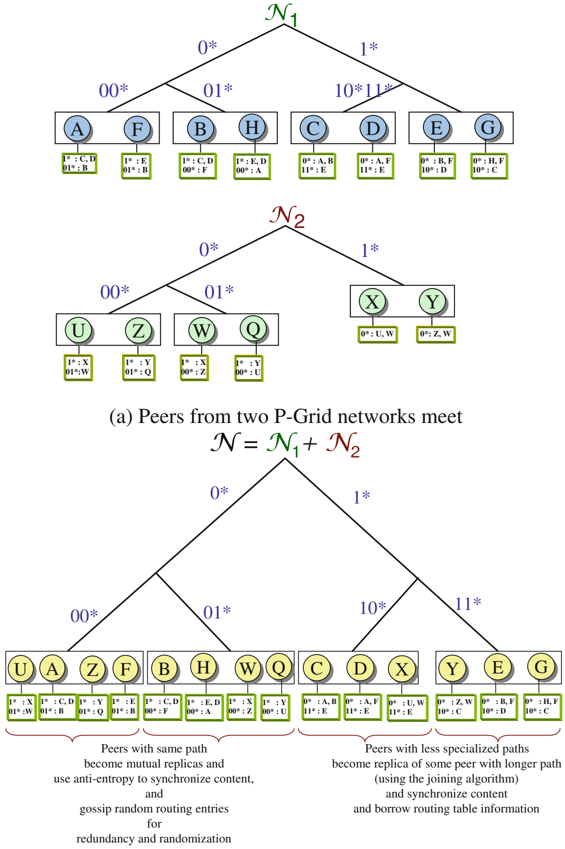
When peers from different overlays meet, similar to the situation in the ring networks, it will be desirable to achieve a merger of the two into a single intertwined network. Figure 12 shows an example of such originally isolated overlays, and the subsequent single merged overlay.

Notice that the originally isolated overlays $\mathscr{N}_1$ and $\mathscr{N}_2$ do not have identical partitioning of the key-spaces, but the eventual merged overlay shown in Fig. 12b should. Next we sketch how such a merger process will happen.

If peers from the two different networks meet, so that their paths are exactly the same (for example peers A and U from networks $\mathscr{N}_1$ and $\mathscr{N}_2$ respectively in Fig. 12a), then they will execute an anti-entropy algorithm to reconcile their content and become mutual structural replicas. In fact, such an anti-entropy algorithm will have to be run among all the other structural replicas of that part of the key-space too, and eventually of the other parts as well. However, since the original members of each network still retain the original routing links, routing functionality is not affected – and whichever keys were originally accessible will continue to be accessible. So to say, peer C will always be able to access all the keys/content available at A before the merger process. The keys from the same key-space which were present in the other network would however be available only after the background replication synchronization has completed. That is to say, a resource available originally only in $\mathscr{N}_2$ at U and Z (but with the same prefix 00 as A) will be visible to C only when A has synchronized its content with any one of U or Z.

Use of structural replication has an additional downside – by not limiting the number of replicas nor having a proper structure among the replicas, it is difficult to have knowledge of the full replica subnetwork at each peer, and hence updates and replica synchronization is typically probabilistic. In contrast, once the ring is reestablished, replica positions are deterministic and hence locating replica is trivial in ring based topologies. Having discovered a replica, the anti-entropy algorithm itself (is an orthogonal issue) and hence the cost of synchronization of a pair of peers will be the same.

When two peers from $\mathscr{N}_1$ and $\mathscr{N}_2$ meet so that one's path is strictly a prefix of the other peer' path, then the peer with shorter path can execute a normal network



(a) Peers from two P-Grid networks meet

$\mathcal{N} = \mathcal{N}_1 + \mathcal{N}_2$
0*
1*
00*
01*
10*
11*
U
A
Z
F
B
H
W
Q
C
D
X
Y
E
G
1* : X
01* : W
1* : C, D
01* : B
1* : Y
01* : Q
1* : E
01* : B
1* : C, D
00* : F
1* : E, D
00* : A
1* : X
00* : Z
1* : Y
00* : U
0* : A, B
11* : E
0* : A, F
11* : E
0* : U, W
11* : E
0* : Z, W
10* : C
0* : B, F
10* : D
0* : H, F
10* : C
Peers with same path
become mutual replicas and
use anti-entropy to synchronize content,
and
gossip random routing entries
for
redundancy and randomization
Peers with less specialized paths
become replica of some peer with longer path
(using the joining algorithm)
and synchronize content
and borrow routing table information

(b) Ideal P-Grid network comprising peers from both networks

Fig. 12 When (peers from) two structurally replicated overlays meet

joining algorithm [2] – extending its path to replicate either the peer it met, or a peer this peer refers it to. For example, *Y* may extend its path from 1 to 11. In order to do so, *Y* will need to synchronize its content with one of the peers which originally had the path 11, say *G*. Moreover *Y* will need to obtain routing reference to a peer responsible for the path 10 (e.g., peer *C*) – information it can obtain from *G* itself.

Since new peers join as structural replica or existing peers, no other existing peer need necessarily to update their routing table for routing functionality (unlike in a ring based topology). Thus, peer *Q* referring to *Y* for prefix 1 continues to refer to it as such, and any query 10 from *Q* is routed first to *Y*, which then forwards it to – say *C*. Peers may however, over time add more routing entries, for instance, *Q* adding a reference to *D* for redundancy in its routing table for the prefix 1. Such changes however is a normal process in the P-Grid network and can be carried on in the background, again without interrupting the functioning of the overlay (and in fact instead making it more resilient to faults and churn).

Consequently, neither joining peers, nor merger of two existing overlay network disrupt the available functionality of the network members.[8]

If peers with different paths meet each other, they need to do nothing, though they can refer each other to peers which are most likely to have the same path (similar to ring based topologies which can forward the peers closer to their respective key-spaces).

5.2.1 Managing Keys in the Merged Network

The amount of data that needs to be transferred from each system to the other is essentially the non-intersecting data. However, there is no need to transfer data from one peer to another merely because the key-space partition a peer is responsible for changes – because with structural replication, new peer joins or network mergers do not in itself automatically change the network's structure.[9]

The important thing to reemphasize is that a peer always finds the keys it could find before the merger process began, irrespective of the state of the merger process. Hence the replica synchronization can be done as a slow back-ground process – hence the performance and network usage is also graceful – that is, merger of two overlays does not suddenly overburden the physical network's resources nor disrupt the functioning of the overlay networks. Such a graceful merger of existing networks also facilitates highly parallelized overlay construction [2] in contrast to the traditional sequential overlay construction approaches.

6 Summary and Conclusion

Research and development in structured overlays now spanning almost a decade, has focused on diverse issues. Adaptation of different topologies in a peer-to-peer environment characterized by large scale, peer autonomy and membership dynamics, maintenance of these topologies – both in terms of ensuring that the topology invariants such as the ring invariant are continuously satisfied, as well as that other performance related concerns like load-balancing are addressed. These activities account for the first five-six years o structured overlay research, bringing it from the drawing board of theoretical results and simulation based validations starting around 2000–2001 to the actual prototyping and benchmark experiments in moderate scale in a controlled environment and with adequate coordination around 2005–2006. The final essential ingredient for a large scale deployment of structured overlays is to allow merger of smaller overlays to form a larger one organically. These smaller over-

[8] Note that the above discussion is true only for write once and then onwards read-only data, since for read/write, it will be necessary to maintain the replicas more pro-actively.

[9] Local view of the structure however changes when a peer with shorter path meets a peer with longer path, and extends its own path according to the network construction algorithm [2], as explained above.

lays can be bootstrapped using any of all the possible manners – quasi-sequential, parallelized, or by merger of even smaller networks. This chapter provides a high level summary and survey of the various kinds of bootstrapping mechanisms for some of the predominant classes of structured overlays.

Acknowledgement

The work presented in this article is partially supported by A*Star SERC Grant No. 0721340055.

Disclaimer: This article summarizes different bootstrapping mechanisms for structured overlay networks, including approaches designed by third parties as well as myself along with various collaborators. The different approaches have accordingly been cited so that each mechanism can be attributed to their original designers.

References

1. K. Aberer (Conference on Cooperative Information Systems (CoopIS 2001)) P-Grid: A self-organizing access structure for P2P information systems.
2. K. Aberer, A. Datta, M. Hauswirth and R. Schmidt (VLDB 2005) Indexing data-oriented overlay networks.
3. K. Aberer, A. Datta and M. Hauswirth (Self-* Properties in Complex Information Systems, ”Hot Topics” series, LNCS, 2005) Multifaceted Simultaneous Load Balancing in DHT-based P2P systems: A new game with old balls and bins.
4. S. Abiteboul and I. Manolescu and N. Preda (SWDB 2004) Constructing and Querying Peer-to-Peer Warehouses of XML Resources.
5. D. Angluin, J. Aspnes, J. Chen, Y. Wu and Y. Yin (SPAA 2005) Fast construction of overlay networks.
6. M.S. Artigas, P.G. Lopez, J.P. Ahullo and A.F. Gomez-Skarmeta Cyclone: A Novel Design Schema for Hierarchical DHTs, (P2P 2005).
7. J. Byers, J. Considine and M. Mitzenmacher (IPTPS 2003) Simple Load Balancing for Distributed Hash Tables.
8. M. Castro and P. Druschel and A-M Kermarrec and A. Rowstron (ACM SIGOPS European Workshop 2002) One ring to rule them all: service discovery and binding in structured peer-to-peer overlay networks.
9. I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, B. Wiley (IEEE Internet Computing, vol.6 no.1, 2002) Protecting Free Expression Online with Freenet.
10. A. Datta and K. Aberer (IWSOS 2006) The challenges of merging two similar structured overlays: A tale of two networks.
11. A. Datta (SASO 2007) Merging Intra-Planetary Index Structures: Decentralized Bootstrapping of Overlays.
12. A. Datta EPFL Phd. Thesis 3615 (2006) SoS: Self-organizing Substrates.
13. P. Ganesan, K. Gummadi and H. Garcia-Molina (ICDCS 2004) Canon in G Major: Designing DHTs with Hierarchical Structure.
14. S. Girdzijauskas, A. Datta and K. Aberer (International Workshop on Networking Meets Databases, NetDB 2005) On Small-World Graphs in Non-uniformly Distributed Key Spaces.

15. S. Girdzijauskas, W. Galuba, V. Darlagiannis, A. Datta and K. Aberer (accepted for publication in Springer's Peer-to-Peer Networking and Applications Journal) Fuzzynet: Zero-maintenance Ringless Overlay.
16. A. Y. Halevy, Z. G. Ives, J. Madhavan, P. Mork, D. Suciu and I. Tatarinov (TKDE vol.16 no.7, 2004) The Piazza Peer Data Management System.
17. IETF-RFC:3174 (http://www.ietf.org/rfc/rfc3174.txt, 2001) Secure Hash Algorithm 1 (SHA1).
18. M. Jelasity and O. Babaoglu (ESOA 2005) T-Man: Gossip-based overlay topology management.
19. M. Jelasity, A. Montresor and O. Babaoglu (IEEE International Conference on Distributed Computing Systems Workshops, 2006) The Bootstrapping Service.
20. J. Kleinberg (STOC 2000) The Small-World Phenomenon: An Algorithmic Perspective.
21. G. Koloniari and E. Pitoura (EDBT 2004) Content-Based Routing of Path Queries in Peer-to-Peer Systems.
22. P. Maymounkov and D. Mazieres (IPTPS 2002) Kademlia: A peer-to-peer information system based on the XOR metric.
23. A. Montresor, M. Jelasity and O. Babaoglu (P2P 2005) Chord on Demand.
24. W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst and A. Löser (Journal of Semantic Web, vol.1 no.2, 2004) Super-peer-based routing strategies for RDF-based peer-to-peer networks.
25. C. G. Plaxton, R. Rajaraman and A. W. Richa (SPAA 1997) Accessing Nearby Copies of Replicated Objects in a Distributed Environment.
26. A. Rowstron and P. Druschel (Middleware 2001) Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems.
27. I. Stoica, R. Morris, D. Karger, F. Kaashoek and H. Balakrishnan (SIGCOMM 2001) Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications.
28. B.Y. Zhao, J.D. Kubiatowicz and A. D. Joseph (2001 UC Berkeley Technical Report UCB/CSD-01-1141) Tapestry: An infrastructure for fault-tolerant wide-are location and routing.

Network-Aware DHT-Based P2P Systems

Marguerite Fayçal and Ahmed Serhrouchni

Abstract P2P networks lay over existing IP networks and infrastructure. This chapter investigates the relation between both layers, details the motivations for network awareness in P2P systems, and elucidates the requirements P2P systems have to meet for efficient network awareness. Since new P2P systems are mostly based on DHTs, we also present and analyse DHT-based architectures. And after a brief presentation of different existing network-awareness solutions, the chapter goes on effective cooperation between P2P traffic and network providers' business agreements, and introduces emerging DHT-based P2P systems that are network aware through a semantic defined for resource sharing. These new systems ensure also a certain context-awareness. So, they are analyzed and compared before an open end on prospects of network awareness in P2P systems.

1 Motivations

Peer-to-peer (P2P) networks are an evolution of the Internet network, and lay over it. However, the Internet network is composed of tens of thousands smaller independent networks called Autonomous Systems (AS) and belonging to various administrative entities (e.g., network providers, universities, companies, etc.), each having its own routing policies. Bilateral connections between ASs are governed by business agreements negotiated between the entities they belong to [1]. And routing information between ASs is exchanged via an exterior gateway protocol such as BGP

Marguerite Fayçal
Institut Telecom, TELECOM-ParisTech, 46 rue Barrault, 75013 Paris, France,
e-mail: Marguerite.Faycal@telecom-paristech.fr

Ahmed Serhrouchni
Institut Telecom, TELECOM-ParisTech, 46 rue Barrault, 75013 Paris, France,
e-mail: Ahmed.Serhrouchni@telecom-paristech.fr

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_12, © Springer Science+Business Media, LLC 2010

[2]. But in P2P networks, routing tasks are distributed across all system peers in an autonomous and spontaneous way without taking the IP network structure or routing into account, and consequent traffic often cross network boundaries multiple times [3], causing redundant traffic and extra delay, and overloading links, increasing the risk of their congestion.

Last years, many researches went on to investigate and understand the relation between routing in the overlay and underlay IP network. Reference [4] shows how current ISP traffic engineering techniques are inadequate to deal with emerging overlay network services, as overlays do not follow ISP's policies. It also illustrate how an uncoordinated effort of the two layers to recover from failures may cause performance degradation for both overlay and non-overlay traffic. The problem of rerouting around failed links was also tackled in [5], which finds out that tuning underlay routing parameters improves overlay performance. Using game theoretic models, [6] studies the interaction between overlay routing and traffic engineering within an AS, and shows that the selfish behavior of an overlay can cause huge cost increases to the whole network. Also [7], which deals with voice over IP as a successful application of P2P, shows that the quality of the relay paths could be improved when the underlying network AS topology is considered.

Moreover, focusing on P2P platforms and applications, their overall performance also depends on the performance of their background P2P routing protocol. So, new systems are mostly based on distributed hash tables (DHT), which are algorithms that provide efficient mechanisms for resource location. However, DHTs assume the system is uniform in available resources and that every node participating in the DHT is within the same transport domain.

Otherwise, despite various hurdles, as copyright issues, legality matters, and security concerns, P2P networks and systems still gain in popularity. The research community continues also to develop interesting applications of P2P technology, together with new platforms for application development. P2P is then used for increasing scalability and decreasing the cost of management and deployment. The consequently continually growth of the P2P traffic appears thus somehow daunting for network providers. An effective cooperation between both parts is then challenging, so that according to the underlying network, peers can find the best instance of the resource they target. But underlay awareness at the P2P level becomes therefore necessary and unavoidable.

2 DHT-Based Architectures

Mathematically, a *DHT* is a distributed injective hash function that associates keys with values, both in the same logical key space *K*. In DHT-based structured P2P systems, the ownership of this key space is split among the active peers of the network. A global unique identifier is thus assigned to every node; it is known as the *nodeID*. Similarly every object has a global unique *objectID*. An object can be any kind of resource.

DHT algorithms map then *objectIDs* to the node responsible for that object. They implements lookup and retrieval functionalities providing two main functions: a *put(key, data)* function that fills the table with couples of corresponding keys and values, and a *get(key)* function that permits to locate an object, taking the object's *key* (which is the *objectID*) and returning the *nodeID* of the peer responsible for that object [8]. The responsible node then either supplies the object directly or indicates where (or how) it can be acquired.

So, given a hash function *h* that should balance the distribution of keys throughout the logical space *K*, and a resource identifier *r* that the peer *p* of address IP *@IP* wants to share on the network, we have then:

$$h(r) = k \in K \text{ ; where } k \text{ is the } objectID$$
$$\text{and}$$
$$h(@IP) = i \in K\text{; where } i \text{ is the } nodeID$$

Then, the resource *r*, or a link towards this resource, is stored on the peer *p* of address *@IP*, so that *dist (k, i)* is minimal; *dist (k, i)* being the distance between *k* and *i*, according to the distance definition in the logical space *K*. In other words, each peer owns all the resources (or links towards the resources) whose key is, in the logical key space, closest to the identifier of that peer. Thus, in the DHT, *k* is mapped to *i*, and the peer *p* can retrieve *r*. Likewise, each peer identified by *j* can retrieve, thanks to his DHT, any resource whose identifier *m* is closest to *j*. If *j* doesn't have *m* in its DHT, it has at least an *n* closest to *m*. Consequently, each key lookup is resolved by iteration, in multiple steps, resulting in a multihop path to be taken in the overlay [9]. DHTs can thus effectively route messages to the unique owner of any given key, and they are typically designed to scale to large numbers of nodes. Typically, the worst case cost to locate an object, in terms of number of messages exchanged, is logarithmic with the number of peers.

The key feature of DHTs used to build P2P architectures is that they use consistent hashing [10], which means that changes to the location where data items are stored are minimal when new locations are added or old locations are removed. In fact, as peers enter and leave the network, messages are exchanged between the peers in the DHT to preserve the structure of the DHT and exchange stored entries. DHTs provide thus data with high degree of availability across a set of peers with dynamic membership. Various DHT implementations may visualize the hash space as a line, a ring, a tree-like structure, a grid, etc. Reference [11] discusses basic geometry underlying DHT routing algorithms and how it impacts their performance in two important areas: static resilience and proximity routing.

All DHT-based P2P architectures share the following four main properties [12]:

1. *Low degree*: each peer keeps only a small number of active connections to other peers.
2. *Low diameter*: the maximal number of necessary hops to reach any peer of the network is minimized.
3. *Greedy routing*: peers independently calculate a short path to the destination.
4. *Robustness*: even if links or peers fail, a path to the destination can be found.

In DHT-based P2P systems, nodes organize themselves in accordance with the DHT's implementation to form a communication graph with an optimal diameter / degree trade-off. Each peer maintains addresses of other peers participating in the same DHT in a routing table, sorted according to specific criteria. And in order to reduce path latency of distributed queries, DHT algorithms include round trip time estimations among such criteria [13].

Nevertheless, DHTs suffer some limitations. First of all, the partitioning of the key space in a DHT is directly related to the number of nodes residing in the system at any given time. As such, whenever a new node joins the system or an existing node leaves it or fails, the partitioning changes, and data must be moved so that the system still works properly and live nodes still be able to determine through the DHT substrate the current live node responsible for the key they look for. Second, DHTs are not designed to answer interval queries, since they link only one object to each key. Lastly, the problem we address in this chapter is the fact that DHTs leave aside the structural aspect of the underlay IP network, assuming that every node participating in the DHT is within the same transport domain. In fact, a single hop in the DHT is likely to involve multiple routing hops in the Internet, and successive hops may lead a message travel back and forth several times. Thus, the physical path so travelled is often less than optimal. DHTs assume also that the system is uniform in resources, since they leave also aside the available resources at each node, such as network bandwidth, free processor, or storage capacity available at each active peer.

More details on DHTs' evaluation can be found in the following. Reference [14] addresses problems in DHT P2P applications. Reference [15] discusses churn. And [16, 17] present performance studies of various DHT algorithms with and without churn.

DHT-based P2P routing protocols are commonly called DHTs. They were first introduced to the research community through four different architectures: Chord [18], Pastry [19], CAN [20], Tapestry [21]. Since that time, a plethora of other DHTs emerged. Some of the most well-known ones are Kademlia [22], Viceroy [23], Bamboo [15], D2B [24]. And the list still grows longer, although very few are publicly released with robust implementations. However, Kademlia is already integrated in two of the most popular P2P filesharing applications: BitTorrent [25], Emule [26].

3 Requirements

As a direct consequence of the total distribution of the Internet, and as DHT-based networks lay over the Internet, it is quite likely that topology information would be distributed at an ISP, a network provider, or an AS or domain level. Thus, beside scalability and robustness, network-aware P2P systems aim to satisfy both P2P users and ISPs. The former are interested in an enhanced time of data retrieval, with better performance; the latter are interested in avoiding congestion on critical links, and look forward to both the optimization of network resources usage, such as bandwidth, and the reduction of operation costs.

Effectively using the underlay network information implies two essential parts:

- firstly, discovery techniques, generating and providing underlay network proximity information,
- secondly, techniques exploiting such information in both query routing and data retrieval processes.

Three techniques of generating proximity information may be identified [27]:

- *Expanding ring search* is a flooding technique for measuring the round-trip time (RTT) between a node and all the others within a defined radius (in terms of network hops).
- *Heuristic based approaches* consist in measuring the RTT between a node and its close neighbours only.
- *Landmark clustering measures* RTTs to selected landmark nodes, and sorts the landmarks in terms of increasing RTTs. Intuitively, nodes having similar distances to these landmarks are close to each other in terms of network latency. This scheme is used in [28].

Vivaldi [29] assigns synthetic coordinates to Internet hosts, so that the Euclidean distance between two hosts' coordinates predicts the network latency between them, with no need of landmark nodes.

Most works on estimating topology information focus on predicting network distance in terms of latency. But for many P2P applications, e.g., video-based ones, throughput is often a more important quality metric. The *iPlane* service [30] aims then to generate and maintain an "atlas" of the Internet using active measurements that contains information about latency, bandwidth, capacity and loss rates between arbitrary Internet hosts.

However, available bandwidth and lossrate estimation from end hosts may be somewhat obscured by lastmile bottlenecks. The notion of network tomography summarizes techniques of active network probing and passive traffic monitoring to infer information about the network topology and link-level characteristics [31].

Lastly, the topology awareness of overlay networks can be modelled, involving a mathematical metric for the degree of topology matching between an overlay network and its underlying physical network [32]. The model is based on an optimization problem, a NP hard problem but solvable in polynomial time for some particular inputs [32].

4 State of the Art

First DHT-based P2P protocols were developed agnostic of the underlay topology. However, Pastry [19] uses certain heuristics to exploit physical network proximity in its overlay routing tables. But decisions are only made when there is a choice between nodes, and locality is not exploited for the underlying routing strategy.

Existing techniques that permit to exploit the network proximity information are subdivided into three categories [33]: proximity routing, geographic layout or topology-based *nodeID* assignment, and proximity neighbour selection [34].

1. With Proximity routing, the overlay network is built independently of the physical topology. During the routing process, if a peer has the opportunity to choose among k possible next hops, it chooses to route the message towards the closest node in the underlay network, among this set of k nodes. An alternative is to choose to route towards the node that represents the best compromise between proximity and progress in the overlay key space. The overall performance of this technique depends thus on k that is proportional to the size of the routing table. Moreover, small hops may be thwarted by an increase of the number of hops.

2. Topology-based *nodeID* assignment aims to map the overlay key space onto the underlay topology, and biases the *nodeID* assignment. Consequently, delays decrease certainly, but uniform distribution of *nodeIDs* in the overlay key space is violated, which leads to loadbalancing problems. Neighbouring nodes are also likely to suffer correlated failures, and thus both robustness and security of the system are likely to be weakened. However, this technique has been successfully used to create a topologically sensitive CAN [28] but it cannot be applied in a one-dimensional id space (e.g., Chord [18], Pastry [19]).

3. Proximity neighbor selection: Like the previous one, this technique builds a topology-aware overlay. But instead of biasing the *nodeID* assignment, it chooses the routing table entries to refer to the closest nodes at the underlay level among all the ones that satisfy the algorithm. This technique is likely to have success only when applied to an overlay protocol that allows certain freedom in constructing routing tables without affecting the diameter. Thus, it can be used with Pastry [19] when constructing the neighbourhood set, but not with CAN [20] or Chord [18], where each routing table entry refers to a very specific point in the overlay key space. Consequently, proximity neighbour selection introduces only low overhead when implemented, and facilitates the cache management since overlay neighbours are likely to be also neighbours at the underlay layer. It can also be viewed as a good compromise decreasing the network's diameter and preserving the load balance, but neighbours' discovery is still tightly related to the protocol.

These three techniques of using generated network proximity information have their limits and are enormously linked in the overlay protocol. Thus, new structural solutions emerged afterwards, and the list still continues to lengthen.

Hereafter, we give a brief presentation of different well-known DHT-based topology-aware P2P systems, namely: Brocade [35], the *expressway* [36], Hieras [37], Toplus [38] and Plethora [39].

Brocade [35] adds a secondary overlay on top of a primary DHT-based P2P network that exploits knowledge of the underlying network characteristics. The secondary layer consists of nodes having high network capacities (bandwidth, processing power), and situated near the network access points such as gateways and routers. These nodes are called supernodes and act as landmarks for each network domain. Each supernode manages a set of local nodes to reduce the traffic in the network, but has to keep information about all overlay nodes inside its domain. They may thus become bottleneck points. To route a message each node first connects to the nearby supernode, then uses the second layer as a shortcut to the destination, reducing both network bandwidth usage and overall physical hops. But this layer still uses a logical routing, which involves for each logical hop several ones at the underlying IP level. Thus, to a certain extent, Brocade pushes the problem back to a secondary network of smaller size.

The *expressway* [36] is an auxiliary network constructed on top of any primary DHT-based P2P overlay, in order to take advantage of the inherent heterogeneity of the underlay network (node connectivity, physical proximity, forwarding capacity, node availability). Two generic approaches exist for such a construction. The first approach uses the AS-level topology derived from BGP [2] reports. The second approach uses a landmark numbering technique that enables proximity neighbour selection and can deal with changing network conditions. The constructed auxiliary network is called an expressway as it is intended to speed up routing. It is formed by nodes with high network capacity, situated near gateways or routers, and called expressway nodes. The first approach requires that every node knows all the nodes in its AS. And to route a message, a node contacts first its local expressway node: the one in its AS or its landmark cluster. In the second one, routing is similar to Distance Vector Routing.

Hieras [37] is a multilayer hierarchical system intending to relieve the problem of distributed overlay routing tasks without awareness of underlay network link latency. At the highest level, peers are grouped in one big ring, and at the lower ones, topologically adjacent peers are grouped into several disjointed rings. But each peer belongs to all the levels simultaneously and manages thus more than one successor list. Each ring is a subset of the overall P2P network and is created in such a strategy that the average link latency between two peers in lower level rings is much smaller than higher level rings. To estimate such proximity for each ring, Hieras employs distributed binning [28], requiring thus the existence of well-defined landmarks nodes. In Hieras, routing tasks are first executed in the smallest ring first. If it fails, it moves up to higher level rings, until eventually reaching the highest level. But a majority of routing hops previously executed in the global P2P ring is so replaced by hops in lower level rings, reducing thus routing latencies.

Toplus [38] is a hierarchical lookup service for structured P2P networks. It organizes peers in groups according to their network IP prefixes, and groups into a new higher order group, and so on following the Internet hierarchical topology it gets from BGP tables [2]. Thus, an AS is subdivided into an IP hierarchy. The routing mechanism of Toplus is based on a generalization of longest prefix matching of IP addresses, using the XOR metric, like Kademlia [22]. Both structure and look up performance of Toplus follow those of the Internet network. However, Toplus suffers a number of drawbacks. Obviously, it is sensitive to correlated node failures that may bring down an entire set of related IP addresses. And another matter is the unbalanced load among the nodes, since the population of the id space is not uniform, because the number of active nodes in an inner group is not necessarily proportional to the IP addresses it covers.

Plethora [39] is a two-layer wide area read-write repository, where peers are expected to have a partially persistent network state and good Internet connectivity in terms of bandwidth. On the top of the global overlay that contains all the peers, the Plethora routing core organizes peers into several local overlays according to ASs and associated proximity. Local overlays serve as locality-aware caches for the global overlay to improve access time to data items. Queries are thus routed first in the local overlay. The size of the local overlay impacts the system's performance and is thus defined by two system parameters: a maximum and a minimum numbers of peers. The size of each local overlay is controlled by the one of its peers with the smallest identifier. Two distributed algorithms are also active in the system. One for merging local overlays when necessary and another one for splitting them with a high probability guarantee that nodes in the same AS stay in the same local overlay after a split operation.

Many other systems still emerge to enhance P2P system's performance by exploiting underlay topology information. And recently, new architectures emerged for topology estimation through layer cooperation, allowing cooperation between P2P traffic and network providers' policies.

A simple scheme [40] can let P2P overlay interacts with underlying ISP infrastructure. It relies on a server, called the oracle, hosted by the ISP and that helps P2P users choose optimal neighbours. More precisely, a P2P user sends the list of potential neighbouring peers to the oracle, which ranks this list based on a number of factors that each ISP can decide individually, like their proximity to the user or higher bandwidth links, or according to its routing policies or its agreements signed with other ISPs. The oracle acts then like an abstract routing underlay to the overlay network but as it is a service offered by the ISP. It has thus direct access to the relevant information (e.g., the one concerning the topology) and does not have to infer or measure it. Although the oracle can be deployed with DHTs, reference [40] focuses mostly on unstructured P2P systems.

Provider Portal for P2P (P4P) [41] is a light-weight architecture enabling explicit communication between P2P (independently of DHTs) and network providers, in order to reduce backbone traffic and lower operation costs. The proposal leverages the fact that the ISP is best-positioned to determine locality and to direct clients not only to nearby peers but also to peers that are accessible over well-provisioned and lightly loaded links. P4P framework consists of a data-plane component and a control-plane component with an *iTracker* providing three kinds of information regarding the network provider: network status/topology, provider guidelines/policies, and network capabilities. However, with the *iTracker*, the P4P framework seems to be a kind of centralized architecture applying CDN (Content Delivery Network) architecture to a filesharing P2P network.

5 Semantics for Resource Sharing

Semantics in P2P systems is an active area research, but it focuses on the problem of mapping or retrieving semantically close data shared by different peers. This paragraph does not deal with the resources' semantic, but tries to define semantics for the identifiers in a DHT, based on the underlay topology in order to lead to a win–win situation where both P2P users and network providers meet their requirements. Hereafter we present two such systems, namely CAP (a Context-Aware P2P system) [42] and NETPOPPS (a NETwork Provider Oriented P2P System) [43]. Both proposals are multi-layer systems exploiting the injective cryptographic hash function of the DHT, which is also used to build the different identifiers of the P2P system. Following proposals are thus independent of the P2P algorithm implemented at the primary global overlay.

5.1 A Context-Aware P2P System

CAP [42] aims to build context-aware *nodeIDs* and *objectIDs*. Therefore it proposes to compute the different identifiers of the system at the secondary levels using HMAC (Hash based Message Authentication Codes) [44], a keyedhashing function, originally aimed for message authentication, thanks to a secret key it uses.

HMAC is a message authentication code that uses a cryptographic key in conjunction with a hash function in order to guarantee integrity between a sender and a receiver. It is computed as follows.

$$HMAC(h,k,m) = h(k \oplus opad||h(k \oplus ipad||m))$$

Here are details of the equation.

- *h* is a commonly used cryptographic hash function; in CAP it is the one used by the DHT of the global P2P overlay.
- *k* is initially defined as a secret key, but in CAP it is a configurable system parameter, called HKey, and which value is known to all the peers of one DHT. Thus, a HKey labels each DHT, which is then known as VDHT (for Virtual DHT). A detailed definition is given in the next paragraph.
- *m* represents the text to be hashed; it is thus either the object name or the address IP of the peer.
- *ipad* and *opad* are the bytes 0x36 and 0x5C respectively, each repeated 64 times.
- $\oplus$ denotes the bitwise XOR operation.
- $||$ denotes the concatenation of two bit strings.

Coming back to the HKey, it could be a simple or compound one. A simple HKey is based on one parameter or a single criterion: it could be a defined characteristic (network metric or policy). A compound HKey is based on the combination of any two or more characteristics (parameters or criteria) and is computed as the output of the cryptographic hash function used by HMAC and applied to the concatenation of those characteristics; e.g., if we are interested in searching a resource according to three metrics $m1$, $m2$ and $m3$, then $HKey = h(m1||m2||m3)$.

A compound HKey could be derived in two or more, simple or compound HKeys: these derived HKeys are then typically based on one or more different characteristics from those the initial compound HKey is based on. But derived HKeys could also be defined with simple HKeys; e.g., if a certain simple HKey could take four values for the same peer (e.g., if the peer resides in four zones), derived HKeys could be defined as a combination of two or more of those values.

The type of the HKey is defined in a global system profile. (The way the system profile is loaded in the system does not affect node operations or routing mechanism; e.g., it could be loaded on a kind of bootstrap node managed by the system operator or service provider.) So, in case of a derived HKey, the profile defines which combination of two or more values or characteristics to take into account, and eventually in which order. In fact, the profile defines a set of priority criteria and identifies the parameters to consider in the routing protocol, defining thus the HKey structure. These criteria and parameters are quantified as the peer arrives in the system, according to an external procedure that doesn't affect the routing mechanism and that could be based on some database reports of the system (e.g., the BGP reports [2] for the identifier of the AS the peer resides in).

Examples of what a simple HKey could be are: the AS identifier, the administrative domain name or the group name or identifier of a specific group communication or a specific secure group, a QoS parameter (e.g., minimum available bandwidth, minimum battery power for mobile systems, minimum storage capacity, etc.), a location parameter (e.g., network or real distance, GPS location, country, etc.), a type of shared files (e.g., a specific movie or audio file type), a language or a topic of shared files (based on metadata), a secret key, a keyword, etc. A compound HKey could be based on two or more such simple HKeys. Derived HKeys could then be defined depending on the routing policy of the system, defined by the above mentioned profile.

Consequently, a heterogeneous P2P network can be layered in a set of different uniform overlays, each one called a zone and characterized by a specific HKey. In order to have homogeneous identifiers in the whole system, each zone is identified by $h(HKey)$. The HKey will then also label both an overlay and the DHT in its corresponding key space, above named VDHT.

Each node can participate in one, two or more of these overlays, depending on its properties (available resources, locality, group membership, etc.) and according to the semantics of the HKey that is taken into account, but resides in at least one overlay, the global one, which could for example be at the lowest overlay level. The other overlays are called local ones, and at one level there could be many zones. In case the system profile does not define derived HKeys, there will be only a single secondary overlay at the top of the global one; else, there will be L more local overlays, where L will be equal to the maximum number of the different possible defined derived HKeys per peer.

The routing mechanism is the same at each level according to the P2P routing protocol defined at the global overlay; the only difference is the HKey characterising the level and thus the usage of the consequent different *nodeIDs*, *objectIDs*, and DHTs.

When a peer needs a data item, it first searches the local overlay it resides in. If the query fails, the peer will then searches the global overlay where the identifiers remain unchanged. When derived HKeys are defined, before searching the global overlay, a peer searches each zone it resides in. To improve the data retrieval performance in the system, a data item is automatically cached in the requester's local overlay if it is retrieved from the global overlay (and of course, its objectID will be then computed using *HMAC* and based on the HKey corresponding to that local overlay).

The joining and failure/departure operations are the same at each level according to the routing protocol implemented in the system; but after a node joins the system (at the global overlay) and before it joins any local overlay at any secondary level, some operations must be taken in order to join the secondary overlay if it exists or to build it if possible.

Otherwise, each existing zone is associated with a data table, called zone table, having the same identifier, and stored on a rendezvous point of the global overlay. The zone table contains up to four *nodeIDs* in the global overlay of nodes already participating in its corresponding local overlay. Thus, a new node in the global overlay has first to look up $h(HKey)$ to be able to join a zone with its corresponding $nodeId$.

If a new node is the first one asking to join its local overlay, then it creates a zone table with the identifier of that local overlay and its own global *nodeID*, it stores the created data table at the rendezvous node, and it starts the local overlay.

CAP guarantees that a data item will be retrieved from the zone where it has been found. In fact, at each secondary layer, when an overlay is populated, all nodes and objects have their identifiers based on the HKey characterising the overlay; so if a query initiated by a node residing in the overlay does not fail, that means that the response is present in this overlay. And even if the response is a pointer to the requested object, the object is necessary in the overlay, since every data presented or represented in a secondary overlay has its identifier necessarily computed based on the HKey characterising that overlay.

Consequently, CAP ensures through a specific semantics that the result of a context-oriented query is context-oriented.

5.2 A Network Provider Oriented P2P System

To enable tight cooperation between P2P traffic and network providers' routing policies and business agreements, NETPOPPS [43] layers P2P overlays in a set of different uniform overlays, and applies the principle of key derivation to $nodeIDs$. But an object keeps a unique identifier at all levels, namely its primary objectID, the identifier it has at the global P2P layer. In fact, objects are managed by nodes.

The principle of key derivation is an efficient key management technique that was proposed in [45] to satisfy the needs of private hierarchical group communication requiring specific confidentiality upward and downward the hierarchy. It is illustrated by Fig. 1, where h is a distributed injective hash function. Considering any node of *Level 1* identified by $Id_1 = k$, it will have at *Level 2* an identifier $Id_2 = h(Id_1)$. This computational relation applies for the identifiers of a same node between any two successive layers, i.e., $Id_i = h(Id_{i-1}) = h^{i-1}(Id_1)$ at any level i. So, any node of level i can compute its identifier Id_j at a level j, where j is greater than i, by deriving $Id_i(j-i)$ times; in other words: $Id_j = h^{j-i}(Id_i)$.

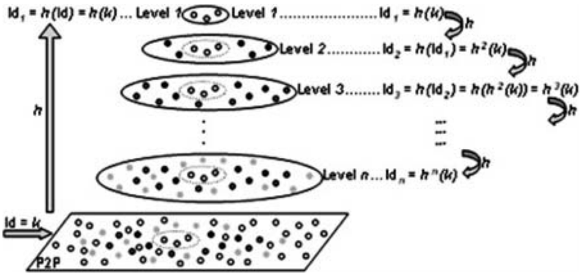

Id₁ = h(Id) = h(k)... Level 1
Level 1.......................Id₁ = h(k)
h
Level 2............Id₂ = h(Id₁) = h²(k)
h
Level 3........Id₃ = h(Id₂) = h(h²(k)) = h³(k)
h
h
Level n...Id_n = h^n(k)
h
Id = k
P2P

Fig. 1 Principle of key derivation for key management

Regarding the system architecture, as illustrated by Fig. 2, secondary overlays are characterized by the width of their constituent entities in term of number of basic entities (BE) merged together. A BE is any kind of routing domain managed by a Network Provider (NP). NETPOPPS proposes that a P2P path connection emerging from any peer of an entity remains entirely inside the same entity. It also proposes that according to NP's business agreements two BEs merge in an intermediate entity (IE) to allow a P2P path connection emerging from one of them to end in the other one. Two IEs can also merge in a new IE, and so on. Besides, secondary overlays lay on the top of the primary P2P overlay in descending order of the width of their entities, in terms of number of merged BEs. This makes the system looks like a collection of several hierarchical subsystems composed of a BE on the top of as many IEs and managed by NPs; each NP managing as many subsystems as BEs it provides. The identifier of an entity is called a *vector* and labels both the virtual overlay in the corresponding entity and the DHT in its corresponding key space. The *vector* of an IE is in fact built in a vector format, by listing in an ascending order the identifiers of all its constituent BEs.

The routing mechanism in each overlay is the same as in the global P2P network; only the *nodeIDs* and DHTs in use are different. When a peer needs a data item, it first searches the local overlay (of the BE) it resides in. If the query fails, the peer will then similarly searches for the data item in the overlay corresponding to

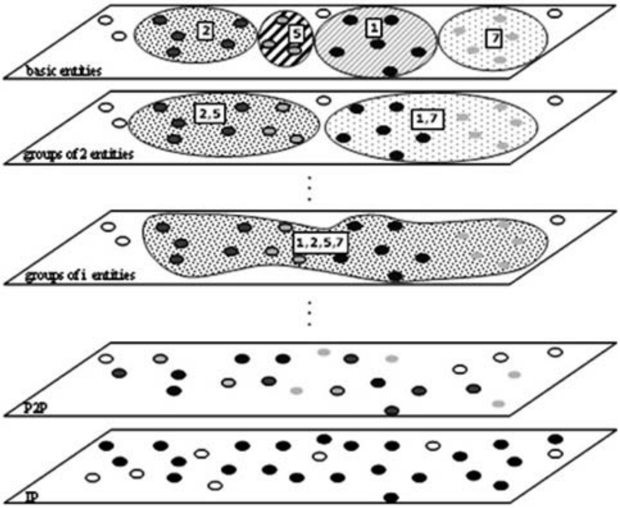


Fig. 2 Overview of NETPOPPS

the first IE; and so on, the peer continues searching for the desired data item in each intermediate overlay going down the hierarchical subsystem, until success. But during a single routing process, only the originator of a query can switch from an overlay to another. The node requester also automatically caches in its local overlay a data item it retrieves from the global or an intermediate overlay.

The joining and failure/departure operations of nodes are the same at each level according to the routing protocol implemented in the system. However, after joining the global overlay and before it joins any other secondary overlay, a new node has to identify the secondary overlays it has to join. NETPOPPS proposes thus that each NP manages a single control node (CN) per BE. The CN does not participate in the routing procedure. It holds a profile that memorizes the vectors of the different IEs of the hierarchy going down from its BE. The NP can also decide on other features to deploy on the CN and other data to save in the profile.

Like any new joining node, the CN joins first the global overlay. Then, as illustrated by Fig. 3, the CN computes its different identifiers according to the principle of key derivation, except an additional derivation: the first derivation of its primary *nodeID* gives its *nodeID* in the BE's overlay. Consecutively, the CN initiates the dif-

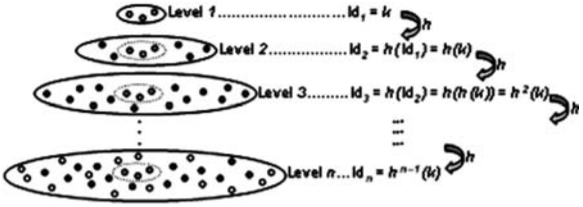

Level 1..........................$Id_1 = k$
h
Level 2................$Id_2 = h(Id_1) = h(k)$
h
Level 3.........$Id_3 = h(Id_2) = h(h(k)) = h^2(k)$
h
h
Level n...$Id_n = h^{n-1}(k)$

Fig. 3 Computation of the different identifiers of a new joining node

Table 1 Structure of the reference table

vector of the BE	primary ID of CN	vector of the 1st IE	...	vector of the Nth IE

ferent overlays. Afterwards, it shares at the global overlay a Reference Table (RT), represented by Table 1, and identified by $h(vector\ of\ BE)$.

So, to correctly join the different secondary levels, a new node searches first for the RT in the global overlay. It computes then its different identifiers as illustrated by Fig. 3, and at each derivation, it matches its new identifier with the vector of the next entity, according to what it has just learned from the RT.

NPs may also operate a control access or enable any other own mechanism (e.g., a bill service) to the nodes joining the different overlays they manage. In this case, after retrieving the RT, new nodes have to send at the global level a join intention request to the CN to know the CN's different *nodeIDs* from the CN itself.

Now, to share a resource identified by an *objectID*, the new node inserts this *objectID* in each overlay it participates in (including the global P2P layer). The system ensures thus that the result of a query found in an entity can be retrieved through a path connection that does not pass by a node of another larger entity.

Consequently, NETPOPPS uses a key management technique to optimize data retrieval according to NP's criteria through a specific semantics.

6 Prospects

CAP [42] and NETPOPPS [43] are two different DHT-based topology aware P2P systems guaranteeing that a data item will be retrieved from the zone or entity where it has been found and the entire P2P path connexion will remain in that zone or entity. Both promise also alleviated message latency and enhanced lookup time. However, the former is more userfriendly and the latter is more network provider compliant.

In fact, CAP is a configurable and extensible context-aware system, where zones are independent from each others, and any node can create and initiate a zone according to any desired performance metric. Guarantee for context-awareness is then given by the semantic of the *objectID* through the HKey. The system can also be adapted for semantic queries or serve as a core routing for any overlay application aiming improved security issues or quality of service. CAP is thus service oriented, and possible applications to implement over it are as numerous as values a HKey can have. However each node or object manages multiple different independent identifiers.

As for NETPOPPS, the key management is greatly simplified, as each object has only one identifier in the whole system and the identifiers of each peer are in a simple relational computation between each other. However, the system is managed by the network provider in a fixed well structured architecture where it is impossible for users to create deliberately their own entity or group of communication, especially if they are clients of different network providers. But the control node opens wide the door to any commercial service, access control, statistics, etc. Nevertheless, NET-POPPS assume that each node knows its BE through an external procedure (it could be based on some database reports like BGP tables [2]).

Consequently, and as shown by in the oracle service [40] and P4P [41], network operators can play an important role in addressing P2P topology-awareness, traffic and costs optimization, and application performance. But operators generally consider topology of the networks they control to be confidential information [13]. Thus, in order to succeed and achieve wide adoption, any solution should provide a method to help P2P applications in peer selection without explicitly disclosing topology of the underlying network [13]. In this perspective, CAP [42], as a generic flexible userfriendly system, is promising.

However, like in any applicative research field, wide adoption will probably never happen without an agreement on a common solution based on open standards [13].

Logical routing, like the one in use in DHT-based P2P networks, will continue to evolve in the coming years, and several various new applications based on it will emerge. New claims will thus be required for corresponding architectures, e.g., security, clustering, quality of service, etc.

DHT-based structures with simple hash function are likely to be unable to afford such requirements, and existing emerging solutions (e.g., P4P [41]) are designed for dedicated infrastructures.

Our contributions aim resource lookup and retrieval within specific domains in the spirit of DHTs and with specific semantics. Large-scale evaluation of these solutions is currently in progress with the OverSim framework [46].

Future interesting research directions will then be explored to distinguish *nodeIDs* and *objectIDs*, i.e., to create one single overlay for the nodes and several overlays for the resources, each having its own semantics.

7 Conclusion

This chapter, dedicated to network-aware DHT-based P2P systems, began with elucidation of the network-aware problem. It detailed DHTs, their strengths and weakness and cleared up the incentives to focus on this specific class of P2P algorithms. The general requirements for network-awareness followed with a quick overview on available systems that explicitly provide underlay information or awareness evaluation.

Then a few well-known DHT-based topology-aware systems were briefly presented, followed by two other systems for topology estimation through layer co-operation. And as the latter allow cooperation between P2P traffic and network providers' policies, the chapter went less briefly on two more emerging architectures to translate such cooperation in semantics within the different identifiers of a DHT system. These two proposals were then compared, concluding with what could be the trend of network-aware P2P systems, followed by a brief discussion on future research directions.

References

1. Meulle M.(2007): Interference of AS business relationships and routing policies in the Internet. PhD Thesis. Université Blaise Pascal Clermont-Ferrand, France
2. Rekhter Y., Li T.(1995): A Border Gateway Protocol 4 (BGP-4). IETF, RFC 1771
3. Karagiannis T., Rodriguez P., Papagiannaki K.(2005): Should ISPs fear Peer-Assisted Content Distribution? In: ACM SIGCOMM Proceedings of IMC. USENIX Association
4. Keralapura R., Taft N., Chuah C.N., Iannaccone G. (2004): Can ISPs Take the Heat from Overlay Networks? In: Proceedings of HotNets-III. ACM Press, New York
5. Seetharaman S., Ammar M. (2006): On the interaction between dynamic routing in the overlay and native layers. In: Proceedings of INFOCOM. IEEE Communications Society
6. Liu Y., Zhang H., Gong W., Towsley D. (2005): On the interaction between overlay routing and traffic engineering. In: Proceedings of INFOCOM. IEEE Communications Society
7. Ren S., Guo L., Zhang X. (2006): ASAP: An AS-aware peer-relay protocol for high quality VoIP. In: Proceedings of ICDCS. IEEE Computer Society
8. Rhea S. (2005): OpenDHT: A Public DHT Service. Ph.D. Thesis., University of California, Berkeley
9. Viennot L. (2005) : Pair-A-Pair: routage dans les réseaux de pair à pair. In: Panorama des Recherches Incitatives en STIC (PaRISTIC)
10. Karger D., Lehman E., Leighton F., Levine M., Lewin D., Panigrahy R. (1997): Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In: Proceedings of STOC. ACM Press, New York
11. Gummadi P.K., Gummadi R., Gribble S., Ratnasamy S., Shenker S., Stoica I. (2003): The Impact of DHT Routing Geometry on Resilience and Proximity. In: Proceedings of SIGCOMM. ACM Press, New York
12. Risson J., Moors T. (2006): Survey of Research towards Robust Peer-to-Peer Networks: Search Methods. Computer Networks 50(17):3485–3521
13. Marocco E., Gurbani V. (2008): Application-Layer Traffic Optimization (ALTO) Problem Statement. draft-marocco-alto-problem-statement-01 (work in progress)
14. Sit E., Morris R.T. (2002): Security Considerations for Peer-to-Peer Distributed Hash Tables. In: Peer-to-Peer Systems (IPTPS), number 2429 in LNCS. Springer, Berlin/Heidelberg
15. Rhea S., Geels D., Roscoe T., Kubiatowicz J. (2004): Handling Churn in a DHT. USENIX Annual Technical Conference
16. Jain S., Mahajan R., Wetherall D. (2003): A Study of the Performance Potential of DHT-based Overlays. In: Proceedings of USITS. USENIX Association
17. Li J., Stribling J., Gil T.M., Morris R., Kaashoek F.M. (2004): Comparing the performance of distributed hash tables under churn. In: Peer-to-Peer Systems-III (IPTPS), number 3279 in LNCS. Springer, Berlin, Heidelberg
18. Stoica I., Morris R., Liben-Nowell D., Karger D.R., Kaashoek M.F., Dabek F., Balakrishnan H. (2001): Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. In: Proceedings of SIGCOMM. ACM Press, New York
19. Rowstron A., Druschel P. (2001): Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM Middleware 2001, number 2218 in LNCS. Springer, Berlin/Heidelberg
20. Ratnasamy S.: (2002) A Scalable Content-Addressable Network. Ph.D. Thesis, University of California at Berkeley
21. Zhao B.Y., Huang L., Stribling J., Rhea S.C., Joseph A.D., Kubiatowicz J.D. (2004): Tapestry: A Resilient Global-Scale Overlay for Service Deployment. IEEE J-SAC 22(1):41–53

22. Maymounkov P., Mazières D. (2002): Kademlia: A peer-to-peer information system based on the XOR metric. In: Peer-to-Peer Systems (IPTPS), number 2429 in LNCS. Springer, Berlin/Heidelberg
23. Malkhi D., Naor M., Ratajczak D. (2002): Viceroy: A Scalable and Dynamic Emulation of the Butterfly. In: Proceedings of PODC. ACM Press, New York
24. Fraigniaud P., Gauron P. (2006): D2B: a de Bruijn Based Content-Addressable Network. Theoretical Computer Science 355(1):65–79
25. Loewenstern A. (2008): DHT Protocol. BitTorrent.org
`http://www.bittorrent.org/beps/bep0005.html.`
`Accessed20August2008`
26. John (2004): eMule v.40f is available. eMule-Project.net
`http://www.emule-project.net.Accessed20August2008`
27. Xu Z., Tang C., Zhang Z.(2002): Building Topology-Aware Overlays using Global Soft-State. HPL-2002-281. Hewlett-Packard Laboratories, Palo Alto
28. Ratnasamy S., Handley M., Karp R., Shenker S. (2002): Topologically aware overlay construction and server selection. In: Proceedings of INFOCOM. IEEE Communications Society
29. Cox R., Dabek F., Kaashoek F., Li J., Morris R. (2003): Practical, distributed network coordinates. In: Proceedings of HotNets-II. ACM Press, New York
30. Madhyastha H.V., Isdal T., Piatek M., Dixon C., Anderson T., Krishnamurthy A., Venkataramani A. (2006): iPlane: an information plane for distributed services. In: Proceedings of OSDI. USENIX Association
31. Castro R., Coates M.J., Liang G., Nowak R., Yu B. (2004): Network Tomography: Recent Developments. Statistical Science 19(3):499–517
32. Rostami H., Habibi J. (2007): Topology awareness of overlay P2P networks. Concurrency and Computation: Practice & Experience 19(7):999–1021
33. Castro M., Druschel P., Hu Y.C., Rowstron A. (2002): Topology-Aware Routing in Structured Peer-to-Peer Overlay Networks. MSR-TR-2002-82. Microsoft Research, Redmond
34. Ratnasamy S., Shenker S., Stoica I. (2002): Routing Algorithms for DHTs: Some Open Questions. In: Peer-to-Peer Systems (IPTPS), number 2429 in LNCS. Springer, Berlin/Heidelberg
35. Joseph A.D., Zhao B.Y., Duan Y., Huang L., Kubiatowicz J.D. (2002): Brocade: Landmark Routing on Overlay Networks. In: Peer-to-Peer Systems (IPTPS), number 2429 in LNCS. Springer, Berlin/Heidelberg
36. Xu Z., Mahalingam M., Karlsson M. (2003): Turning Heterogeneity into an Advantage in Overlay Routing. In: Proceedings of INFOCOM. IEEE Communications Society
37. Xu Z., Min R., Hu Y. (2003): Hieras: A DHT Based Hierarchical P2P Routing Algorithm. In: Proceedings of ICPP. IEEE Computer Society
38. Garcés-Erice L., Ross K.W., Biersack E., Felber P.A., Urvoy-Keller G. (2003): TOPLUS: Topology Centric Lookup Service. In: Group Communications and Charges (NGC), number 2816 in LNCS. Springer, Berlin/Heidelberg
39. Ferreira R.A., Grama A., Jagannathan S. (2004): Plethora: An Efficient Wide-Area Storage System. In: Bougé L., Prasanna VK (eds) High Performance Computing (HiPC), number 3296 in LNCS. Springer, Berlin/Heidelberg
40. Aggrawal V., Feldmann A., Scheideler C. (2007): Can ISPs and P2P Users Cooperate for Improved Performance. In: ACM SIGCOMM Computer Communications Review 37(3): 31–40
41. Xie H., Krishnamurthy A., Silberschatz A., Yang Y.R. (2007): P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers. P4PWG Whitepaper
42. Fayçal M., Serhrouchni A. (2007): CAP: A Context-Aware Peer-to-Peer System. In: Meersman R., Tari Z., Herrero P (eds) On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, number 4806 in LNCS. Springer, Berlin/Heidelberg
43. Fayçal M., Serhrouchni A. (2008): NETPOPPS: A Network Provider Oriented Peer-to-Peer System. In: IFIP Proceedings of NTMS. IEEE Communications Society

44. Krawczyk H., Bellare M., Canetti R. (1997): HMAC: Keyed-Hashing for Message Authentication. IETF, RFC 2104
45. Hassan H.R., Bouabdallah A., Bettahar H., Challal Y. (2005): An Efficient Key Management Algorithm for Hierarchical Group Communication. In: IEEE, CreateNet Proceedings of SecureComm. IEEE Computer Society
46. Baumgart I., Heep B., Krause S. (2007): OverSim: A Flexible Overlay Network Simulation Framework. In: Proceedings of Global Internet Symposium (GI) in conjunction with Infocom 2007. IEEE Communications Society

On Adding Structure to Unstructured Overlay Networks

João Leitão, Nuno A. Carvalho, José Pereira, Rui Oliveira, and Luís Rodrigues

Abstract Unstructured peer-to-peer overlay networks are very resilient to churn and topology changes, while requiring little maintenance cost. Therefore, they are an infrastructure to build highly scalable large-scale services in dynamic networks. Typically, the overlay topology is defined by a peer sampling service that aims at maintaining, in each process, a random partial view of peers in the system. The resulting random unstructured topology is suboptimal when a specific performance metric is considered. On the other hand, structured approaches (for instance, a spanning tree) may optimize a given target performance metric but are highly fragile. In fact, the cost for maintaining structures with strong constraints may easily become prohibitive in highly dynamic networks. This chapter discusses different techniques that aim at combining the advantages of unstructured and structured networks. Namely we focus on two distinct approaches, one based on optimizing the overlay and another based on optimizing the gossip mechanism itself.

João Leitão
INESC-ID / IST, Lisboa, Portugal, e-mail: jleitao@gsd.inesc-id.pt

Nuno A. Carvalho
University of Minho, Braga, Portugal, e-mail: nuno.carvalho@di.uminho.pt

José Pereira
University of Minho, Braga, Portugal, e-mail: jop@di.uminho.pt

Rui Oliveira
University of Minho, Braga, Portugal, e-mail: rco@di.uminho.pt

Luís Rodrigues
INESC-ID / IST, Lisboa, Portugal, e-mail: ler@ist.utl.pt

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_13, © Springer Science+Business Media, LLC 2010

1 Introduction

Gossip, or epidemic, protocols have emerged as a highly scalable and resilient peer-to-peer approach to implement several application level services such as reliable multicast [1, 8, 12, 21, 25, 28, 34], data aggregation [17], publish-subscribe [7], among others [22, 27, 36]. This chapter addresses peer-to-peer communication support for reliable and scalable information dissemination. A gossip-based broadcast protocol usually operates as follows: to broadcast a message, a node selects t nodes at random from the system (t is a configuration parameter called *fanout*) and sends the message to them. Upon the reception of a message for the first time, each node simply repeats this procedure.

The gossip approach to data dissemination has several advantages: *(i)* it is simple to implement, *(ii)* it shares the load evenly across all nodes in the system, making gossip protocols highly scalable, in fact the load imposed by the process in each node of the systems only has to grow logarithmically with the size of the system in order to ensure atomic broadcast with a high probability [1, 6], and finally, *(iii)* its inherent redundancy makes gossip protocols highly resilient to node and link failures (for instance, [25] proposes a gossip-based broadcast protocol that can maintain high resilience even in scenarios where 80% of the nodes in the system fail simultaneously).

Gossip-based protocols were originally designed to operate with full membership information [1, 5], by maintaining locally at each node a list with the identifiers of every other node in the system (typically, an identifier is a tuple $(ip, port)$ that allows a node to be reached). However, such approach is not scalable, not only due to the large size of the membership but also (and mainly) due to the cost of maintaining such information up-to-date in dynamic systems. For scalability, nodes may rely on a *peer sampling service* [11, 16, 25, 41], provided by a membership protocol that operates with the goal of maintaining locally, at each node, a small random subset (called a *partial view*) of the full membership list. In this case, nodes use their local partial views to select peers for exchanging messages.

Partial views establish *neighboring* associations among nodes that define an overlay network which can be used for gossiping data. Ideally, the selection of peers from local partial views should be equivalent to a random selection of peers across the full membership. Therefore, the resulting overlay has a random (unstructured) topology.

Although this randomness has some desirable features, it also raises two distinct problems that may impair the efficiency of applications and protocols that operate on top of these unstructured overlay networks. First, it prevents the underlying network topology to be taken into consideration by the peer sampling service. This problem is known as *topology mismatch* [29]: it usually leads to scenarios where many overlay links are suboptimal with regard to a given network efficiency criteria such as bandwidth or latency. Second, because the overlay structure is random, it fails to exploit the *natural heterogeneity* [33] of large-scale peer-to-peer systems, and does not take advantage of nodes and links that have a higher capacity.

Node heterogeneity is easier to take into account in structured multicast protocols, by explicitly building dissemination structures according to a predefined efficiency criteria [10, 35, 37, 43], and then use these structures (such as spanning trees [3, 24]) to disseminate multiple messages. In a structured approach, nodes with higher resource availability can offer a bigger contribution to the global dissemination effort by having larger degrees or by being placed closer to the root of the tree (the reader should notice that nodes located at the leaves of the tree are not required to contribute to the message dissemination effort).

The trade-off between gossip-based and structured approaches is clear: By avoiding the need to build and maintain a spanning tree, epidemic multicast provides extreme simplicity. Moreover the balanced load across all nodes in the system, is a key factor to achieve resilience and scalability. On the other hand, structured multicast provides better resource usage (and thus higher performance when the network is stable) by optimizing the cost of the spanning tree according to efficiency criteria such as network bandwidth and latency. However, structured approaches have to deal with the complexity of rebuilding the structure when faults or network reconfiguration occurs.

In this chapter, we address techniques that aim at combining the best of both approaches, namely, the simplicity, scalability and resilience of unstructured overlay networks with the performance of structured approaches. In order to achieve this, some degree of structure is added to low-cost unstructured overlay networks to improve their performance without impairing the relevant properties of unstructured approaches. We start by presenting a survey of several existing works that aim at improving the topology of unstructured overlay networks. This is followed by a description of key properties of unstructured overlay networks that should be preserved when introducing structure. Then we introduce two approaches that can be used to introduce structure in unstructured overlay networks. The first approach bias the topology of an unstructured overlay according to some performance metric without compromising the resilience of the overlay. The second is based on an emergent behavior, approximating the operation of a structured overlay on top of an unstructured overlay. We present a performance evaluation of both approaches.

2 Adding Structure to Unstructured Overlay Networks

In this section we survey several existing protocols that can be used to add structure to, or improve the locality properties of, unstructured overlay networks. Then we list some key properties of unstructured overlay networks We also enumerate some relevant metrics that can be used to evaluate the benefits of adding structure to unstructured overlays. Finally we identify two distinct methodologies that allow to add some degree of structure to such overlays.

2.1 Existing Protocols

2.1.1 Narada

Narada [3] is a protocol designed to support application-level multicast. Narada aims at minimizing the overhead introduced by implementing multicast at the application layer (as opposed to IP multicast). Namely, Narada aims at minimizing both the stress induced on physical links (due to duplicate packets that transverse the same links) and the end-to-end latency of the multicast process. To address these issues, Narada is based on an unstructured overlay network, whose topology is adapted for improved performance. The goal of the protocol is to build an overlay that is: self-organizing, efficient, self-improving, and adaptive to network dynamics. We now briefly describe how the topology of the unstructured overlay is adapted.

Since Narada is targeted at small and medium sized systems, it is assumed that each node has access to a full membership list containing node identifiers for all participants in their algorithm. Using this information, Narada builds a limited degree unstructured overlay network, named a *richly connected mesh*. The overlay network topology is biased to obtain a majority of low cost links. On top of the resulting unstructured overlay network, a distance vector routing algorithm is executed to build, and maintain, a spanning tree routed at each sender for each multicast group. Each node will therefore maintain a local routing table which is used to disseminate multicast messages.

When a new element joins the system, it contacts a peer already present in the network to obtain the current full membership list. The node then randomly selects a few group members to whom it sends a join message, requesting to be added as their neighbor in the overlay. Nodes rely in the resulting unstructured overlay network to exchange periodic messages which are used to update global membership information, and to detect failed nodes and partitions.

After the execution of the steps described above, nodes form a fully connected unstructured overlay network. However, links in the network have a high probability to be suboptimal for a given set of target efficiency criteria. To improve the overlay, nodes capture information about their execution environment. For instance, in video conferencing applications, the overlay is biased to improve both point-to-point latency and bandwidth. Passive monitoring techniques are used to obtain available bandwidth values for peers in the system. Active monitoring techniques based in the exchange of ping messages are used to extrapolate values for latency. This information is then used in heuristics which bias the overlay topology as follows:

Add links Periodically, each node *n* selects another random non-neighbor node *p* and performs measurements to assess the efficiency of the communication with *p*. Also, *p* sends back to *n* a copy of its local multicast routing table. Node *n* uses both the received information and the expected efficiency of the link between *n* and *p* to locally compute a utility function that evaluates the gain of adding such link to the overlay. If the expected gain is above a given threshold value, n will add the link between himself and p to the overlay.

Remove links Periodically, each node selects, and removes, the (local) link with the lowest utility value. The computation of these utility values is done in such a way that the resulting value is an overestimate of the real utility of the link. Moreover, the link is only removed if its utility falls bellow a given threshold value. This is done to ensure some stability in the overlay. Notice that because the network is dynamic, the utility of a link may also be dynamic. It would not be efficient to allow situations where one is constantly removing and adding the same link to the overlay.

Using this methodology, the unstructured overlay topology can be biased, increasing the efficiency of applications that operate above it.

2.1.2 Localiser

The Localiser algorithm [30] aims at solving the *network mismatch* problem while ensuring that the overlay network remains connected despite failure of large percentage of nodes. It also ensures a fair degree distribution among every peer in the system. The localiser algorithm is fully decentralized and only relies in local knowledge. In [30] the authors show the impact of the algorithm on the operation of the unstructured overlay network maintained by the Scamp protocol [11].

The goal of Localiser is to bias the topology of the overlay network such that the majority of neighbors kept by each node are "close" peers (given a "network distance" criterion). The protocol also aims at biasing the original overlay such that every node has the same amount of neighbors, which also contributes to increase the failure resilience of the overlay.

Localiser was designed based on a metropolis model [32]. This is an iterative model in which an utility function f is minimized. In order to do this, on each iteration, the utility of the current overlay configuration c is compared with the utility of a possible alternative configuration c'. The algorithm is probabilistic given that the acceptance of an adaptation of the overlay configuration from c to c' is determined by a decreasing probability function in $f(c') - f(c)$. The reader should notice that this approach allows to perform adaptations to the overlay topology which increase the value of the function f. This however is required by the algorithm to avoid local minima configurations.

The specific algorithm is based on a periodic operation executed by every node in the system. In each iteration, each node n executes the following steps:

1. Node n selects at random 2 overlay neighbors p_1 and p_2 and computes for each one a local cost function.
2. Node n obtains the node degree of p_1 and p_2. Furthermore it also obtains from p_1 the cost of the link between p_1 and p_2.
3. Node n locally computes the global benefit of exchanging its link with p_1 for a link between p_1 and p_2.

4. Finally, node *n* uses a probabilistic function, which takes into account the benefit, the expected cost of the adaptation, and node degree, to make the decision of applying, or not, the link exchange. If the exchange is accepted, *n* coordinates with p_1 and p_2 the steps required to perform the adaptation.

The algorithm can be parameterized to give more weight, in the probability function, to the balance of node degrees or to the proximity of neighbors (notice that this proximity notion is encoded in the link cost function). The probability function can also be tuned to promote maintenance of low cost configurations or to increase the probability of a faster convergence.

This scheme allows an unstructured overlay to self adapt to reach a configuration where most neighbors are "local" (i.e., with a small link cost) while at the same time, improving the degree distribution. This leads to more efficient overlay configurations with increased resilience to faults.

2.1.3 Araneola

Araneola [31] is a protocol for reliable and efficient multicast based on unstructured overlay networks. The protocol is able to build, and maintain, a bounded degree overlay. Moreover, Araneola, includes a mechanism for exploiting network proximity in the overlay.

This mechanism operates independently of the main task of the Araneola protocol. It operates by adding new links to the overlay to promote communication between close peers. The proposed extension to the original protocol is based on two distinct components, namely: a task to locate nearby peers, and another task which establish connections with discovered nearby peers. These tasks operate as follows:

Locating nearby nodes The task operates by capturing network performance values from peers selected from a local partial view. The performance values are used to sort nodes into a candidate list which is then used by the second task. Several techniques can be employed to capture performance values (different metrics require different techniques). For instance, the authors of [31] rely on a network-level hop-count between peers which is extracted using the UNIX *tracepath* utility (aiming at lowering point-to-point latency in the network).

Connecting to nearby nodes This task tries to maintain a number of nearby neighbors equal to a target value *NB* (*NB* is a protocol parameter). Periodically, if the number of nearby neighbors of a node falls bellow the target value, the node issues a CONNECT_NEARBY request to the first peer in its candidate list (the list generated by the previously described component). A node which receives a CONNECT_NEARBY message will accept the connection, and add the issuing node to its nearby neighbors set, if it has a number of nearby neighbors below *NB*. If the node accepts the request it replies with a CONNECT_OK_NEARBY. Upon the reception of a CONNECT_OK_NEARBY the receiving node adds the sender to its nearby neighbors set, unless the number of its nearby neighbors has reached the target value of *NB*. In the later case, the node will send a LEAVE_NEARBY message, which will result in the removal of the newly established connection.

This extension to the original Araneola protocol is able to correlate the topology of the overlay and the topology of the underlying network. As a result, better links are used in the overlay and the latency of the dissemination process is decreased.

2.1.4 GoCast

GoCast [39] is a protocol for reliable group communication that operates by building a multicast tree on top of an unstructured overlay. This overlay network is biased to promote low latency links and a constant degree for all nodes in the system. GoCast operates by maintaining both near and random neighbors. The protocol also relies in a peer sampling service which is used as a bootstrap overlay, and also as a source for random peers for the protocol operation. The protocol tries to select a sample of C_{rand} and C_{near} nodes such that the sum of these numbers converges to a given value *D* (all these values are protocol parameters). TCP connections are maintained for every neighbor of each node, and all communication made between such peers is done by relying in these connections. UDP is used for communication for all remaining nodes (for instance, to obtain latency measurements).

Periodically every node in the system performs two operations; the first to maintain random neighbors and the second to maintain a nearby neighbors. We now describe these operations.

Maintaining random neighbors To this purpose, each node *p* compares its current number of random neighbors with the target value: C_{rand}. If these values are equal, no operation is required. If the number of random neighbors is below the target value, then the node adds a random node (obtained from the peer sampling service) to its neighbors set, and establishes a TCP connection to it. Finally, if the number of random neighbors is above C_{rand}, the node *p* might take one of the following corrective measures:

- If the current number of *p*s random neighbors is equal or above $C_{rand}+2$, *p* selects two random neighbors, *q* and *r*, and asks them to replace their links with *p* for a link between *q* and *r*. This allows to reduce by 2 the number of random neighbors of node *p*, while preserving the number of random links for all remaining nodes in the system.
- If one of *p*s random neighbors, *q*, has a number of random neighbors above C_{rand}, *p* simply asks *q* to remove the link between them. This allows for two nodes, in a single step, to approximate their number of random neighbors to the target value of C_{rand}.

Maintaining nearby neighbors GoCast mechanism to maintain nearby neighbors is composed of three sub protocols. The first serves to replace nearby neighbors for other nearby neighbors with a lower latency. The second is used to add nearby neighbors to the partial view of the node, when the number of nearby neighbors is bellow the target value of C_{near}. Finally, there is a protocol to remove nearby neighbors when their number is equal or above $C_{near}+2$.

- Periodically, a node p, measures its latency to a random peer, say r. If the estimated latency to r is lower than an existing nearby neighbor n, p might exchange its link with n for a link with r if and only if the following four conditions are true: *(i)* the number of nearby neighbors of n must at least $C_{near}-1$; *(ii)* the number of nearby neighbors of r must be below $C_{near}+5$; *(iii)* if the number of nearby neighbors of r is above C_{near}, then r must have a nearby neighbor with a higher latency than the estimated latency value between p and r and finally, *iv)* to ensure that there is a relevant gain in the link exchange, the latency between p and r must be at least, half of the latency between p and n.
- In order to add new nearby neighbors, p selects a random peer r and simply adds it as his neighbor if, and only if, the conditions *ii* and *iii* depicted above, are true.
- If node p has a number of nearby neighbors equal or above $C_{near}+2$ it drops the connection to a nearby neighbor which does not have a number of nearby neighbors below the threshold of $C_{near}-1$.

GoCast can successfully bias an unstructured overlay network to improve its performance, reducing the overall latency, and also converge to a configuration where all nodes have a degree value between $D-2$ and $D+2$.

2.1.5 T-Man

T-Man [15] is a generic topology management scheme for unstructured overlay networks. The goal of the protocol is to reach a given target topology from a pure random overlay. Examples of target topologies are torus, ring, or some user defined topology. The topology is defined by fixed size partial views that are maintained at each node (the size c of these partial views is a protocol parameter).

The protocol relies on a ranking function that, at any give node, is able to sort a set of peers accordingly to some preference. The ranking function must be able to encode, somehow, the desired topology, in the sense that it must be able to provide clues, for every node, concerning the most relevant peers that they should keep as neighbors, in order to generate the desired topology. The operation of the protocol is based on a periodic exchange of information performed by every node, which works as follows:

1. A given node n starts by using the ranking function to select the neighbor p that is closer to itself;
2. then n sends to p a set of peers containing n's identifier, n's partial view, and a random sample of other peers in the system[1];
3. when p receives this information from n it replies with a similar set of peers: p's identifier, p's partial view, and a random sample of other nodes in the system;

[1] This random sample can usually be extracted from an out-of-band peer sampling service such as Cyclon [41] or HyParView [25].

4. after this exchanged is performed, both nodes use a merge function which also relies in the ranking function to return the *c* best peers from the union of each node partial view and the received set of peers;
5. each node partial view is then updated to contain the *c* nodes returned by the local execution of the merge function.

This protocol allows the overlay network to converge for the desired topology. Because nodes exchange information with their closest peers, the probability of receiving information concerning other peers which are good candidates to improve the overlay topology is increased, as there is a high probability that nearby nodes will be trying to converge their partial views to contain similar peers.

2.1.6 Plumtree

The Plumtree protocol [24] is a dissemination scheme which relies on a reactive unstructured overlay network to embed a highly resilient low cost spanning tree. The protocol uses this spanning tree to bias the communication pattern of a gossip-based broadcast protocol, in order to lower the inherent overhead of the gossip protocol, without impairing its reliability. To do this, eager push is used in overlay links which belong to the spanning tree while, for both fault-tolerance and support the healing mechanism of the spanning tree, lazy push is used on the remaining overlay links.

The protocol has two main components. The first builds the spanning tree structure by removing redundant links when they exist. The second component is able to heal the spanning tree structure whenever a node fails or leaves the system, and also recover from message loses due to membership dynamics. We now describe the behavior of each component:

Building the spanning tree Initially, Plumtree assumes that every link in the overlay belongs to the spanning tree. The same is true whenever a new link is added to the overlay due to natural dynamics in the peer-to-peer system. When a link is used to transmit a redundant gossip message, the protocol removes that link from the spanning tree. Therefore, when the first broadcast message is disseminated, it is eagerly flooded through the overlay. However, when the dissemination process is concluded, the spanning tree has been completely established, and following broadcast messages are only eagerly transmitted in the links which belong to the tree.

Healing the spanning tree In the presence of node failures the spanning tree may become disconnected. This results in poor reliability, as disconnected nodes will miss broadcast messages. To address this, nodes also send lazy push messages through the remaining links of the overlay (e.g., links which are not part of the spanning tree). Messages transmitted by lazy push only carry an unique identifier for broadcast messages and omit the original payload, therefore these messages are in fact announcement messages that a new broadcast message (or messages, as more than one message identifier can be carried in a single IP packet) is available.

When a node receives an announcement for a given broadcast message that it has not received yet, it starts a timer. When the timer expires, if the payload is not yet locally available, the node request the payload to the neighbor who sent the announcement. This message implicitly adds the link to that neighbor to the spanning tree, effectively healing the tree.

After a failure, several nodes may concurrently add links to the spanning tree; this may result in the creation of redundant links. However, the mechanism for building the tree will detect such redundancy during the dissemination of the next broadcast message and, as result, will prune existing redundant links (if any).

This protocol is completely decentralized, and by using two distinct transmission modes (eager and lazy push) it can lower the overhead of disseminating broadcast messages to a value comparable to some multicast structured solutions. The use of lazy push ensures, at a low cost, that the natural resilience of gossip protocols is maintained. Also, as a result of the strategy used to select links, links that form spanning tree are those with lower latency.

2.2 Key Properties to Preserve

In the previous paragraphs, we have surveyed a number of protocols to optimize the overlay network to achieve better performance. There are however a number of key topology properties[2] that should be preserved during the optimization, as listed below:

Connectivity: The overlay is connected if there is at least one path that allows every node to reach every other node in the overlay. The overlay should remain connected despite failures that might occur. If this requirement is not meet, isolated nodes will not receive broadcast messages.

Degree Distribution: The degree of a node is the number of edges of a node, or in other words, the number of neighbors that a given node has.[3] The degree of a node is both a measure of its reachability on the overlay and also a measure of its contribution to maintain the overlay connected. If the probability of failure is uniformly distributed in the node space, for improved fault-tolerance, all nodes should have the same degree value. Nodes that have a small degree will more easily become disconnected from the overlay as the number of faults increases. On the other hand, the failure of nodes with high degree may have an undesired impact in the overall connectivity of the overlay.

[2] Some of these properties are intrinsically related with graph properties, as it is, an overlay network can be seen as a graph, where nodes are represented by vertex, and links, or neighboring relations, are represented by edges. Depending on the nature of these relations, graphs can be directed or undirected.

[3] To be precise, usually partial views establish asymmetric neighboring relations, therefore the degree is viewed as two distinct components: in-degree and out-degree. However, in this chapter we will mostly focus on systems which use a gossip-based membership protocol which offers to nodes access to symmetric partial views therefore, we do not consider these components as being distinct.

Average Path Length: A path between two nodes in the overlay is a set of edges that connect one node to the other. We define the average path length as the average of all shortest paths between all pair of nodes in the overlay. The average path length is closely related to the overlay diameter. To promote the overlay efficiency when broadcasting messages, the average path length between nodes should be as small as possible. Large values of average path length have two negative implications: *(i)* The number of hops required for messages to reach all nodes increases, with a negative impact in the broadcast latency and, *(ii)* the broadcast process becomes more prone to failures, as the time window for failures increases (e.g., the number of steps required to fully disseminate a message increases).

Clustering Coefficient: The clustering coefficient of a node is the number of edges between that node's neighbors divided by the maximum possible number of edges across those neighbors. The clustering coefficient captures a density of neighbor relations across the neighbors of a given node, having it's value between 0 and 1. The clustering coefficient of a graph is the average of clustering coefficients across all nodes. The clustering coefficient of an unstructured overlay should be as small as possible, and failure to meet this requirement has the following negative implications: *(i)* the number of redundant messages received by nodes when disseminating data increases, especially in the first steps of the dissemination process; *(ii)* the diameter of the overlay increases, which in turn will make the average path length increase, and finally *(iii)* it decreases the fault resilience of the overlay, as areas of the overlay which exhibit a high value of clustering can more easily became disconnected.

The interested reader can find a more detailed discussion of these and other properties of random overlays in [23, 25].

2.3 Performance Metrics

Several metrics can be used to measure the performance of a gossip-based broadcast protocol operating on top of a random overlay network. In this chapter we focus mainly on offering high dependability for applications requiring reliable broadcast. Therefore, the metrics that we present here are mostly related with the operation of broadcast protocols, as we specifically aim at biasing the overlay topology to minimize the message dissemination overhead, while preserving the typical reliability of gossip-based broadcast protocols.

Average Link Cost: We assume that each link of the overlay may be tagged with a *cost*. Costs may be associated to a concrete (underlay) network metric such as link latency. However, the link cost may also be associated to higher level utility metrics; for instance, in a file sharing peer-to-peer system it could be a measure of the semantic similarity between the data stored at the edges of a link.

Reliability: Gossip reliability is defined as the percentage of correct nodes that deliver a given broadcast message. A reliability of 100% means that the protocol was able to deliver a given message to all active nodes or, in other words, that the message resulted in an atomic broadcast as defined in [21].

Latency: We define latency of a gossip-based broadcast protocol as the time between the instant when a message is transmitted by its original sender, to the moment when the last peer, which receives the broadcast message, delivers it to the application layer. The reader should notice that one can have good latency values by failing to deliver the message to a large number of nodes. Therefore the goal of a gossip-based broadcast protocol should be to achieve a low latency value while ensuring a high reliability. Moreover, latency values are only comparable between broadcast protocols that exhibit a similar reliability value, for systems composed of the same number of nodes.

Last Delivery Hop: The Last Delivery Hop measures the number of hops required to deliver a broadcast message to all recipients. When a message is gossiped for the first time, its hop count is set to 1 and, each time it is relayed in the overlay, the hop count is increased in one unit. The last delivery hop is the hop count of the last delivery for a given broadcast message or, in other words, is the maximum number of hops that a message must be forwarded in the overlay before it is delivered to all participants. This metric is closely related with the diameter of the overlay and with the the latency of a gossip protocol. In other words, it can be seen as an efficiency metric.

2.4 Methodologies

We can distinguish two main methodologies that allow to introduce some degree of structure in unstructured overlay networks. These methodologies operate at distinct levels:

Overlay Optimization: This methodology consists in manipulating the neighboring relations among peers, effectively changing the unstructured overlay network topology and changing the communication patterns among peers (e.g., by changing the communication peers for each node). This methodology aims at improving the overall overlay network, which is the support for several gossip protocols, by replacing existing links between peers for alternative links which present a better performance given an efficiency criteria (e.g., such as latency). In Section 3 we describe with some detail a protocol based on this approach.

Gossip Optimization: This methodology consists in selecting different communication modes for transmitting messages between different peers. The possible modes to transmit messages are: eager push, lazy push and pull [24]. This methodology supports the emergence of structure, from the unstructured overlay, by establishing patterns in the communication modes used among peers. Protocols based on this approach are able to: *(i)* make a better usage of network resources; *(ii)* reduce the communication overhead of gossip protocols and also, *(iii)* address heterogeneity of nodes. In Section 4 we describe a protocol which employs this technique. Another protocol which also uses this methodology can be found in [24].

The first technique is used in a larger number of proposed solutions. Protocols such as Narada, Localiser, Araneola, GoCast and T-Man use variants of overlay optimization to bias, or adapt, the topology of random overlay networks. The second technique has only recently been proposed (the protocol presented in Section 4 and Plumtree are two of the few protocols employing this technique).

3 Overlay Optimization

3.1 Overview

In this section, we describe a protocol to Bias the Overlay Topology according to some target efficiency criteria *X*, or simply X-BOT. A target efficiency criteria can be, for instance, to better match the topology of the underlying network. However, in X-BOT, biasing the overlay is done without compromising key properties of random overlay networks (such as the node degree, small diameter, and low clustering coefficient), which are essential to ensure the efficiency and reliability of some peer-to-peer applications such as, gossip-based broadcast protocols.

X-BOT relies on the combined use of two distinct partial views, inspired by HyParView [25], a gossip-based membership protocol that illustrated how to achieve a high resilience to faults (as high as 80% of simultaneous nodes failures) in a gossip-based broadcast protocol using a low fanout value. The architecture of this protocol is based in the combination of a small sized active view and a larger passive view. X-BOT relies on a similar architecture in order to optimize the overlay network used for message dissemination.

The goal of the protocol is to reduce, as much as possible, the average link cost of the overlay network defined by the active views. For that purpose, X-BOT actively bias the neighbors in the active view using random peers extracted from the larger passive view. This is feasible because only the active view is used for communication among peers. Moreover, the passive view is maintained by a cyclic strategy [25] which ensures that the contents of this view are periodically updated and therefore, gives access to an increasing number of potential neighbors over time to each node. X-BOT is flexible allowing to bias a topology for different criteria such as, link latency or content similarity as a result of being independent of the cost function. The protocol only requires costs to be comparable and totally ordered.

3.2 Architecture

X-BOT maintains two distinct, and disjoint, partial views: a small sized symmetric active view and a larger cyclic passive view. As in HyParView, the active view is used mainly for communication among peers and TCP connections are maintained to neighbors in this view.

X-BOT assumes that all nodes have access to a local *Oracle*. Oracles are components that export a `getLinkCost(Peer p)` interface, which returns the link cost between the invoking node and the given target node p in the system (since there is a single link to each neighbor, in this chapter we use interchangeably link cost or node cost when referring to the output of the Oracle). The implementation of such Oracles are not discussed in the chapter. However, for completeness, we provide a brief description of three simple Oracles.

3.2.1 Oracles

Latency Oracle This Oracle operates by measuring round trip times (RTT) to peers. This can be performed by exchanging probe messages with Oracles located at other nodes.[4] The Oracle must be aware of the peers which are known at the local host, and it slowly measures the RTT for each know node (this value can be directly used as the cost value).

Internet Service Provider Oracle In a setting where exchanging messages across different ISPs has an increased monetary cost, it might be useful to keep as many neighbors as possible that share the same ISP. Such Oracle can be built by maintaining information concerning the local ISP and a table of costs for each known ISP. When the Oracle becomes aware of a new peer, it simply exchanges local ISP information with the remote Oracle and asserts the cost for the link using the local cost table.

IP-based Oracle X-BOT can also leverage on previous work addressing the use of inexpensive Oracles that do not require the exchange of control information [19, 20]. Such Oracles are able to calculate neighbor proximity values, which can be used as cost, using IP aggregation information (for instance, using a match of common IP prefixes to calculate a measure of proximity between two peers).

Oracles are not required to be perfect for the operation of the protocol, in the sense that provided costs are not required to be 100% accurate. The interested reader can refer to [26] for experimental results that show the effect of unreliable Oracles in the protocol.

[4] Probe messages can also be piggybacked on application traffic, for instance, when measuring the cost for peers in the active view.

3.2.2 Rationale

The rationale of X-BOT is as follows. As in HyParView X-BOT maintains a small active view and a larger passive view. However, unlike HyParView, that strives to ensure the stability of the overlay, X-BOT relaxes stability to be able to continuously improve the overlay. This allows the topology of the unstructured overlay to self adapt to better match the requirements of the application executed on top of it. Periodically, each node starts an *optimization round* in which it attempts to switch one member of its active view for one (better) neighbor of its passive view. In the optimization protocol, a node uses its local Oracle to obtain an estimate of the link cost to some random selected peers of its passive view. The number of nodes π for which the cost is measured in each optimization round is a protocol parameter called *Passive Scan Length*. This parameter limits the maximum number of optimization exchanges started by each node each time it runs the optimization procedure. Similarly to the original HyParView protocol, the passive view is not biased.

X-BOT strives to preserve the connectivity of the overlay. This has two implications: *(i)* nodes only make an effort to optimize their active views when they have a full active view (i.e., no bias is applied to active views until connectivity of the nodes is ensured). Furthermore, each node attempts to maintain some unbiased neighbors, as we explain in the next section; *(ii)* we try to preserve the degree of nodes that participate in a optimization procedure, given that the node degree has a significant impact on the connectivity of the overlay. To ensure this, each optimization round involves typically 4 nodes in the system as we describe later in the text.

3.2.3 Unbiased Neighbors

By blindingly imposing a bias in the topology of the overlay, one may easily break some of the desirable key properties of a random overlay, such as the low clustering coefficient, low average path length, or connectivity [39]. The negative effect of such bias can be even more notorious in the architecture of X-BOT, that relies on small partial views. To avoid this flaw, X-BOT does not bias all members of the active view. Instead, each node maintains both “high-cost” (unbiased) and “low-cost” (biased) neighbors. The number of unbiased neighbors each node keeps is a protocol parameter called *Unbiased Neighbors* and simply denoted μ.

Unfortunately, it is not trivial to decide which peers have a “high-cost”, given that nodes are not expected to have global knowledge of the system, not only regarding membership information but also regarding global metrics, such as the average link cost in the overlay. To circumvent this limitation, X-BOT maintains the active views of each node sorted by link cost, where the first element of each active view is the neighbor with the largest link cost. Therefore, a node never applies any bias to the first μ members of its active view. Also, whenever a change occurs in the elements of a node’s active view, the active view is reordered using the same criterion. The same happens if the cost of a node changes due to modifications in the execution environment.

3.3 Algorithm

In this section we briefly describe the operation of the overlay optimization algorithm (a more detailed description can be found in [26]). The algorithm executed at each optimization round is depicted in Algorithm 12 and illustrated in Fig. 1. The algorithm listing has been simplified for clarity, for instance, we omitted some insertions of nodes into passive views and the mechanisms required to ensure the symmetry of active views.

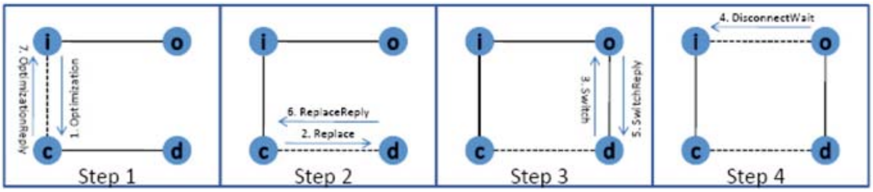


Fig. 1 Steps of the optimization protocol

Usually an optimization round involves 4 nodes of the system, and each round is composed of 4 steps, one for each node that participates in the optimization. The goal of these steps is to exchange two of the existing links in the overlay for other two links such that the cost of the two added links is lower than the cost of the original links. To ensure that the overall cost of the overlay is reduced in an optimization round, and because it is assumed that link costs are symmetric, the optimization scheme only requires that two of the four participating nodes in a round consult their local Oracles.

A complete optimization round requires the serial exchange of seven messages. However, in most cases, each node involved in the optimization only has to send and receive at most two messages. Given that the optimization of the overlay can be executed as a background activity, the cost of the adaptive mechanism can be easily tuned to become negligible when compared with the (application) data traffic.

Oracles are not required to be perfect, in the sense that they might provide information that is not fully accurate. Namely, two nodes may obtain different costs for the same link when they consult their local Oracle. For instance, [20] states that longest IP prefix and latency has an approximate correlation of -0.85. In cases where Oracles are not perfect, nodes have to make decisions with inaccurate information. Therefore, the *isBetter* evaluation function (depicted in Algorithm 12) includes some hysteresis, namely, a given link *new* is only considered to have a lower cost than another link *old*, if the difference between the cost (obtained through the Oracle) offers a gain above a given *Threshold*, which is a protocol parameter introduced to address the inaccuracy of Oracles. The threshold value can be calculated experimentally for each Oracle. This value should be related with medium precision and error drift of Oracles, which can be measured in a semi-controlled environment such as Planet-lab [4].

Algorithm 12: Optimization Procedure

```
Data:
    activeView //fixed size sorted list
    passiveView //fixes size list

every $\Delta\ T$ do
    if isFull(activeView) then
        candidates$\longleftarrow$ randomSample(passiveView, $\pi$)
        for $i := \mu$ ; $i < $ sizeOf(activeView) ; $i :=i + 1$
            $o \longleftarrow$ activeView[$i$]
            while candidates $\neq \{\}$ do
                $c \longleftarrow$ removeFirst(candidates)
                if isBetter($o$,$c$) then
                    Send(OPTIMIZATION($o$, myself),$c$)
                    break

upon Receive(OPTIMIZATION, $o$, $i$) do
    if $\neg$ isFull(activeView) then
        activeView $\longleftarrow$ activeView $\cup$ $\{i\}$
        Send(OPTIMIZATIONREPLY(true, $o$, $\perp$, myself),$i$)
    else
        $d \longleftarrow$ activeView[$\mu$]
        Send(REPLACE($o$, $i$, myself),$d$)

upon Receive(OPTIMIZATIONREPLY,answer,$o$,$d$,$c$) do
    if answer then
        if $o \in$ activeView do
            if $d \neq \perp$ then
                Send(DISCONNECTWAIT(myself),$o$)
            else
                Send(DISCONNECT(myself),$o$)
            activeView $\longleftarrow$ activeView $\setminus\{o\}$
        passiveView $\longleftarrow$ passiveView $\setminus\{c\}$
        activeView $\longleftarrow$ activeView $\cup\{c\}$

upon Receive(REPLACE, $o$, $i$, $c$) do
    if $\neg$ isBetter($c$,$o$) then
        Send(REPLACEREPLY(false, $i$, $o$, myself),$c$)
    else
        Send(SWITCH($i$, $c$, myself),$o$)

upon Receive(REPLACEREPLY,answer,$i$,$o$,$d$) do
    if answer then
        activeView $\longleftarrow$ activeView $\setminus\{d\}$
        activeView $\longleftarrow$ activeView $\cup\{i\}$
    Send(OPTIMIZATIONREPLY(answer,$o$,$d$,myself),$i$)

upon Receive(SWITCH,$i$,$c$,$d$) do
    if $i \in$ activeView or received(DISCONNECTWAIT from $i$) then
        Send(DISCONNECTWAIT(myself),$i$)
        activeView $\longleftarrow$ activeView $\setminus\{i\}$
        activeView $\longleftarrow$ activeView $\cup\{d\}$
    Send(SWITCHREPLY(answer,$i$,$c$,myself),$d$)

upon Receive(SWITCHREPLY,answer,$i$,$c$,$o$) do
    if answer then
        activeView $\longleftarrow$ activeView $\setminus\{c\}$
        activeView $\longleftarrow$ activeView $\cup\{o\}$
    Send(REPLACEREPLY(answer,$i$,$o$,myself),$c$)

isBetter($old$,$new$)
    cOld := Oracle.getLinkCost($old$)
    cNew := Oracle.getLinkCost($new$)
    return cOld > cNew $\wedge$ $\frac{(cOld\ -\ cNew)}{cOld} \geq$ $THRESHOLD$
```

3.4 Performance

In the following sections we show performance results of the X-BOT protocol. To better understand the impact of this approach, which affects the properties of the unstructured overlay network, we used simulation, which allows us to extract performance metrics in a system composed of a large (10.000) number of nodes.

3.4.1 Experimental Setting

The figures presented are the result of extensive experimental evaluation of this approach in the PeerSim simulator [18] using its cycle based engine. Each cycle is a virtual time slice where each node can execute periodic operations. Additionally, a single broadcast message is also disseminated in each cycle to enable the observation of the reliability of a gossip-based broadcast protocol operating on top of biased unstructured overlays. All experiments were conducted in a system composed of 10.000 nodes. Simulation were run on top of a network model composed of 13.037 routers generated by Inet-3.0 [42] with its default parameters. In order to calculate the cost between neighbors, we calculated the shortest paths in the underlying network among the 10.000 overlay nodes. The cost between two nodes is the sum of the pseudo geographical distance, generated by Inet-3.0, of links between the routers that form shortest paths. All results report an average extracted from 5 independent runs. Each one of these runs used one of 5 distinct random network topologies generated using the methodology described above.

Simulations use the configuration parameters for HyParView reported in [25]. The most relevant configuration parameters are the active view size, which was set to 5 and the passive view size, which was set to 30. Different *Unbiased Neighbors values* (μ) ranging from 0 to 5 (all) were tested; the last configuration corresponds to the operation of the original HyParView protocol (given that no bias is applied to any member). Moreover, in all simulations we used the following parameters:

The period between optimizations was set to 2 simulation cycles. In each simulation cycle each node initiates an exchange with a random peer in the overlay, which results in the update of both nodes passive views. Moreover, in average, a node will also participate in an exchange initiated by other peer. Therefore, setting the period between optimization to two cycles ensures that, between executions of optimization steps the passive view of nodes is updated, increasing the possibility of selecting new nodes that can be used to improve the active view of the node.

Passive Scan Length (π) was set to 2, so each time a node executes the step 1 of the optimization algorithm, it measures, at most, 2 nodes from its passive view. This also limits the number of nodes which are replaced in the active view of any given node in a single round as 2. Setting π to a small value allows to achieve two goals: (i) It promotes some stability in the overlay, as we avoid to exchange the majority of nodes in the active view of a single node in the context of a single optimization execution.; (ii) It lowers the cost of the overall optimization process.

3.4.2 Stable Environment

First, X-BOT performance results are shown for a stable environment, where no failures were induced. Simulations were run for 250 cycles. Nodes were added to the overlay using the *Join* mechanism provided by the HyParView protocol as described in [23] and had access to local perfect Oracles (e.g., their precision was of 100%).

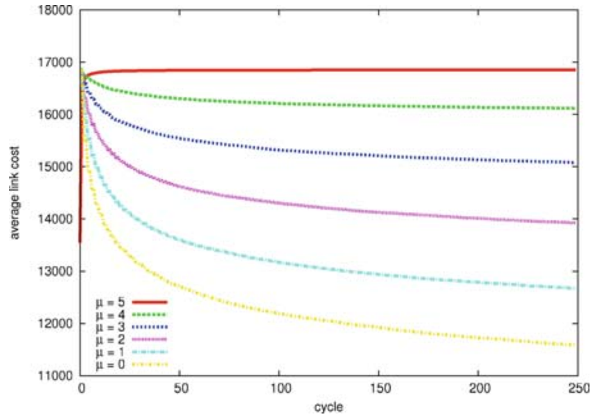


Fig. 2 Average Link Cost

Overlay Properties. Figure 2 shows the average link cost of the overlay as the system evolves. As expected, while the original HyParView protocol ($\mu = 5$) shows a constant link cost in steady state, X-BOT, is able to lower its average link cost from approximately 5% while maintaining 4 unbiased neighbors to 25% when keeping 1 unbiased neighbor or even 32% when no unbiased member is maintained in the active view. Notice also that, although the optimization process works continuously, 50 simulation cycles is enough to obtain a visible improvement in the average link cost.

Figure 3 depicts results for clustering coefficient. As expected, if the bias is applied to all members of the active view the clustering coefficient of the overlay increases. On the other hand, maintaining a single unbiased neighbor is enough to partially mitigate this effect. However, as it can be observed after 250 simulation cycles, the clustering coefficient of the network with a single unbiased member is still above that of the original HyParView protocol. Interestingly, when 2 to 4 unbiased neighbors are maintained in the active view, the clustering coefficient drops to values below those obtained with 5 random neighbors. This phenomenon can be explained as follows: By maintaining active views sorted by cost, the selected unbiased neighbors are those with a larger cost known by each node during the lifetime of the system. In other words, X-BOT with no extra cost, promotes the maintenance of "long distance" links in each active view. The same effect is also visible in Fig. 4 where we show the average path length values.

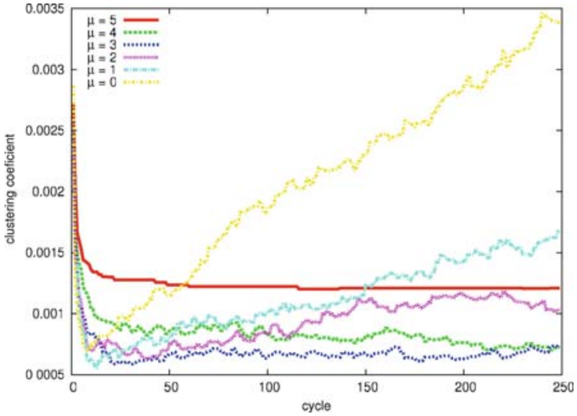


Fig. 3 Clustering coefficient

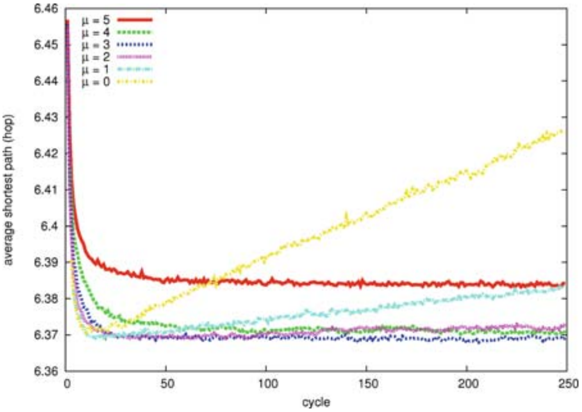
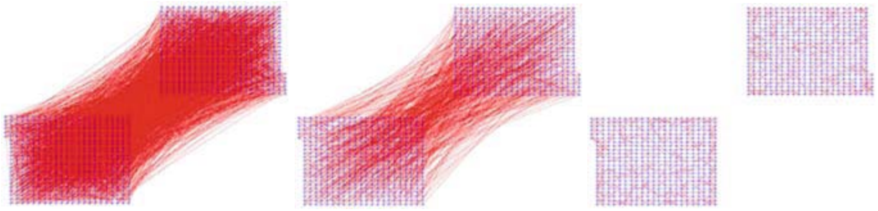


Fig. 4 Average path length

Broadcast Reliability. Experiments were also conducted to assert the impact in the reliability of a gossip-based broadcast protocol operating on top of a biased overlays resulting from the operation of X-BOT. Experiments were conducted as follows: in each simulation cycle, we select a random node in the system to broadcast a message. After the dissemination process is complete, we evaluate the reliability of the broadcast by observing the percentage of active nodes that receive that message. The reliability obtained for all configurations of the protocol was of 100% in steady state. This shows that the overlay maintained by the protocol did not became disconnect due to the operation of X-BOT.

Effect of Node Clustering. In order to illustrate some of the benefits of X-BOT, the algorithm was compared with T-Man in a scenario where nodes are highly clustered in the cost function space. Notice that in these scenarios the optimization of the overlay can lead to the creation of disconnected clusters.



(a) Initial State (b) Out protocol Result (c) T-Man Result

Fig. 5 Our protocol vs T-MAN highly clustered system

The experiment was conducted using a version of T-Man [14] for the PeerSim simulator, which can use the same Oracles as X-BOT. The selection of this version of T-Man was motivated by an additional parameter k present in the protocol which is similar to the μ parameter of X-BOT. The parameter limits the biasing the protocol performs over nodes partial views to $c-k$ neighbors. The remaining k neighbors of each node are selected at random.

The experiment was designed as follows: It starts by positioning 1024 nodes in two spatial clusters. Then HyParView is used to build an overlay connecting these nodes. The resulting overlay is depicted in Fig. 5a where each node is represented by a cross, and each overlay link is represent by a line. As the reader can observe, the overlay is highly connected.

Then X-BOT, with μ set to 1, and T-Man, with k set also to 1, are executed for 250 simulation cycles to optimize the distance between neighbors, resulting in the topologies depicted, respectively, in Fig. 5b, c. As expected, X-BOT replaces a large number of links between the two clusters by better links inside each cluster. However both clusters are still highly connected. The same is not true for T-Man, which breaks the connectivity between the two clusters. This happens because the random selection of nodes in T-Man is implicitly biased to nodes that are, at most, at two hops of distance whereas X-BOT extracts distant neighbors from an unbiased low cost passive view. Moreover, as described earlier, X-BOT (unlike T-Man) promoted the maintenance of long cost links in the overlay which, is such scenarios, is essential to ensure the global connectivity of the overlay.

3.4.3 Massive Failures

In this section we provide results concerning the reliability of the gossip-based broadcast protocol on top of the optimized overlay network. Specifically, after the induction of massive node failures in the system that range from 10% to 95% of all nodes. These faults were induced in the system after 250 cycles of simulation to ensure that the overlay had time to converge to a biased version. After the induction of failures, simulations were conducted for an extra 250 cycles and in each cycle a random correct node was selected to initiate the dissemination of a broadcast message. After the completion of the dissemination process, the reliability of the broadcast protocol was measured. Figure 6 shows the average reliability obtained while broadcasting 250 messages after massive failures. The reader should notice that all protocol instances present similar values for reliability. This happens due to the passive view maintained by HyParView. Notice that, although we use passive views across nodes to bias the topology of the overlay, the properties of the passive view are not affected, thus the healing properties of passive views are not affected by the operation of X-BOT.

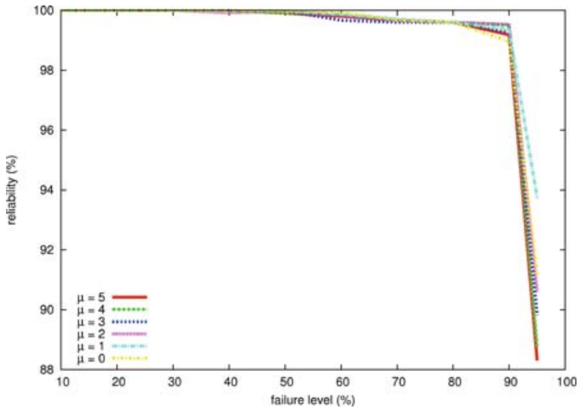


Fig. 6 Average reliability of 250 messages after failures

One could expect that maintaining a single unbiased neighbor would decrease the resilience of the overlay to node failures. In fact, one could imagine that if the unbiased member fails, it would be difficult for this member to be replaced by another unbiased member. However, X-BOT only attempts to apply some bias when the active view is complete. Therefore, when a node crashes and needs to be replaced, the replacement is picked at random from the passive view. This policy, combined with the use of a sorted active view explains the effect that we depicted in Section 3.4.2 in which unbiased neighbors become implicitly biased for high costs.

4 Gossip Optimization

4.1 Overview

In this section we describe a protocol which employs the *gossip optimization* methodology. This allows the structure to emerge from the natural operation of a gossip-based broadcast protocol. The approach leverages the fact that two distinct communications modes between peers may be used, namely: eager push and lazy push. A main goal of the protocol is to approximate the efficiency of a gossip-based broadcast protocol to that of a structured broadcast protocol, more specifically, to the efficiency of a dissemination strategy which employs a spanning tree that covers all peers in the system. Moreover the protocol aims at building a solution that exhibits the following characteristics:

- Does not impair the natural resilience of gossip protocols.
- Avoids the necessary overhead to explicitly build and maintain the spanning tree structure. Namely, it avoids the maintenance of additional state for supporting the dissemination structure.
- Uses a decentralized approach, with low requirements in node coordination.
- Optimizes the dissemination process to take node heterogeneity into account in such a way that nodes with higher capacity can contribute more to the dissemination of messages.

In the following sections we describe, in some detail, the operation of the protocol and how it allows to achieve the characteristics listed above.

4.1.1 Background

Gossip-based multicast protocols are often based on an *eager push gossip* approach [8, 22, 34]: A gossip round is initiated by a node that has received a message, relaying it to a number of targets. However, it is well known that this strategy consumes a lot of bandwidth, as the fanout required for atomic delivery leads to multiple copies of each message being delivered to each destination.

A different trade-off can be achieved by using a *lazy push* strategy, which defers the transmission of the payload. In detail, during a gossip round a node will send only an advertisement of the new message. Transmission of the payload is initiated only if the message is unknown to the recipient. This allows the message payload to be transmitted only once to each destination, at the expense of an additional round-trip. Lazy transmission has also an impact on the reliability, as the additional round-trip and resulting increased latency widens the window of vulnerability to network faults. The impact is however small for realistic omission rates and can be compensated by a slight increase in the fanout [33]. Even with a larger fanout, one can significantly improve the use of the network resources namely, in terms of consumed bandwidth.

In fact, one can mix both approaches in a single gossiping round [33], thus providing different latency/bandwidth trade offs depending on how many messages are eagerly transmitted.

4.1.2 Approach

The architecture stems from the observation that, in an eager push gossip protocol, paths leading to deliveries of each message implicitly builds a distinct random spanning tree for each broadcasted message. This tree is composed of links which belong to the overlay network, and therefore, one can say that it is embedded in the underlying random overlay. If one knew beforehand which links are used to transmit messages that lead to deliveries, one could use eager push gossip for those links and lazy push gossip for all others. This would achieve exactly once transmission for each destination. Unfortunately, this is not possible, as one cannot predict which paths in the overlay will lead to message deliveries.

There is however an alternative strategy which is feasible: If one of the embedded trees is selected beforehand for eager push gossip, one increases the probability that the links which compose that tree lead to an increased amount of message deliveries. This happens because lazy push has additional latency, and paths that use it will be outrun by paths that solely rely in eager push. If one assigns nodes and links with higher capacity to support such spanning tree, the performance of the protocol should approach that of a structured approach, where the tree is build beforehand, in such a way that it improves one, or more, efficiency criteria. Note that keeping redundant lazy transmissions is essential to retain the gossip resilience properties. On the other hand, this strategy requires the explicit coordination among peers to maintain a tree structure which imposes additional overhead and complexity.

Instead of selecting a single embedded tree, gossip optimization aims at increasing the probability of implicitly creating spanning trees in a gossip protocol to include nodes and links with higher capacity. The resulting structures are therefore probabilistic in nature: Nodes and links are selected with different probabilities for payload transmission (or in other words, for eager push transmissions). Consequently, structure emerges naturally from the strategy used for scheduling message payloads in a combined eager/lazy push gossip protocol. One of the main challenges is to achieve an emergent structure without global coordination, while at the same time obtaining a meaningful performance improvement.

4.2 Architecture

The architecture that supports the operation of the approach described above, is depicted in Fig. 7. Notice that it relies in an additional layer which is located below a pure eager push gossip protocol. This layer, called the *Payload Scheduler*, selects when to transmit the message payload (by using a combination of eager push and lazy push) in a transparent manner for the gossip protocol above. The Payload Scheduler layer can be decomposed into three separate components, also depicted in Fig. 7:

Lazy Point-to-Point The lazy point-to-point module is in charge of intercepting the interaction between the gossip layer above and the transport protocol below. It queries the Transmission Strategy module to decide whether to send the

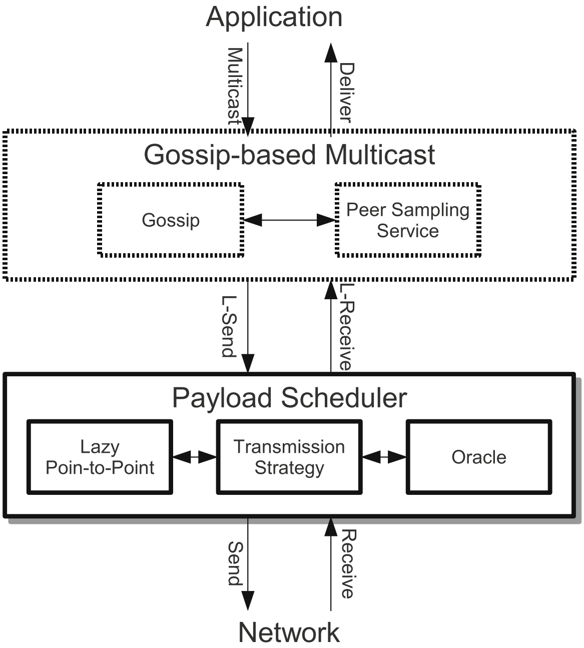


Fig. 7 Protocol architecture overview

payload immediately (in the case, the exchange is performed in a pure eager push mode) or to delay the payload transmission until a request is received. As we will later describe, this module is also in charge of generating and replying to payload requests.

Transmission Strategy The Transmission Strategy module is the core component of the Payload Scheduler. It defines the criteria that is used to defer payload transmission at the sender and, at the receiver, when to request a specific payload transmission (e.g., when to trigger lazy push requests by the receiver). Note that different strategies may be implemented, according to the efficiency criteria is targeted for optimization. Notice that the goal of each strategy is to generate a combined protocol (push gossip plus scheduler) that approximates the behavior of a structured multicast approach.

Oracle The last component of the Payload Scheduler is the Oracle. This component goal, similar to the Oracles described previously in this chapter, is to offer additional information to nodes. This information can be either configured in a static way or extracted, in run-time, concerning the performance data about the operation of the system, for instance, by computing round-trip delays between peers. This data is then used to feed the Transmission Strategy module.

In the remainder of this section, we describe each component of the architecture in detail as well as the existing interfaces among them.

Algorithm 13: Simple push gossip-based broadcast protocol

```
Data:
    $K$ //known messages

1:  initially
2:      $K \longleftarrow \emptyset$

3:  proc MULTICAST($d$) do
4:      FORWARD(MKID(), $d$, 0)

5:  proc FORWARD($i$,$d$,$r$) do
6:      DELIVER($d$)
7:      $K \longleftarrow K \cup \{i\}$
8:      if $r < t$ do
9:          $P \longleftarrow$ GETPEERS($f$)
10:         for each $p \in P$ do
11:             trigger $L\text{-}Send(i,d,r+1,p)$

12: upon $L\text{-}RECEIVE(i, d, r, s)$ do
13:     if $i \notin K$ then
14:         FORWARD($i$, $d$, $r$)
```

4.2.1 Gossip Protocol Layer

As noted before, a fundamental aspect of this approach is that the Payload Scheduler can operate in a manner that is transparent for the operation of a simple push gossip-based broadcast protocol. Therefore, it can be applied to different gossip protocols, such as [8, 22, 34].

Nevertheless, for self containment, we depict in Algorithm 13 a typical push gossip protocol. This implementation assumes the availability of a peer sampling service [16] providing an uniform sample of f other nodes with the GETPEERS(f) primitive. It assumes also an unreliable point-to-point communication service, such that a message m can be sent to a node p using the L-SEND(m,p) primitive. A message m is received from a peer p by handling the L-RECEIVE(m,p) up-call. The gossip protocol maintains a set K of known messages (line 2), initially empty. This set is used to detect and eliminate duplicates. In more detail, the algorithm works as follows.

- The application calls procedure MULTICAST(d) to multicast a message with payload d (line 3). This simply generates an unique identifier and forwards it (line 4). The identifier chosen must be unique with high probability, as conflicts will cause deliveries to be omitted. A simple way to implement this is to generate a random bit-string with sufficient length.
- Received messages are processed in a similar manner (line 12), with the difference that it is necessary to check for, and discard, duplicates using the set of known identifiers K (line 13) before proceeding.
- The forwarding procedure FORWARD(i,d,r) (line 5) uses the message identifier i, the payload d and the number of times, or rounds, the message has already been relayed r, which is initially 1. It starts by delivering the payload locally using the

DELIVER(d) up-call. Then the message identifier is added to the set of previously known messages K (line 7). This avoids multiple deliveries, as described before. Actual forwarding occurs only if the message has been forwarded less than t times (line 8) [22] and consists in querying the peer sampling service to obtain a set of f target nodes and then sending the message, as in lines 9 and 11. Constants t and f are the usual gossip configuration parameters [6].

For simplicity, we do not show how identifiers are removed from set K, preventing it from growing indefinitely. This problem has been studied before, and efficient solutions exist ensuring with high probability that no active messages are prematurely garbage collected [8, 22].

4.2.2 Payload Scheduler Layer

The Lazy Point-to-Point module is the entry point to the Payload Scheduler. It controls the transmission of message payload using a simple negative acknowledgment mechanism. The policy used for each individual message is obtained from the Transmission Strategy module using a pair of primitives:

- EAGER?(i,d,r,p) is used to determine if payload d for message with identification i on round r should be immediately sent to peer p. Note that if the method always returns true the protocol operates as a pure eager push protocol. Otherwise, if the method always returns false, the protocol operates as a pure lazy push protocol.
- $(i,s) =$ SCHEDULENEXT$()$ blocks until it is the time for some message i to be requested from a source s. From the correctness point of view, any scheduling policy is safe, and will maintain all gossip protocols properties, as long as it eventually schedules all lazy requests that have been queued.

The Lazy Point-to-Point module also informs the Transmission Strategy of known sources, for each message and also, when payload has been received using the following primitives:

- QUEUE(i,s) queues a message identifier i to be requested from source node s. The Transmission Strategy module must keep an internal queue of known sources for each message identifier, and eventually schedule them, unless payload is received first.
- CLEAR(i) clears all requests on message i. Note also that a queue eventually clears itself as requests on all known sources for a given message identifier i are scheduled.

The Lazy Point-to-Point module is depicted in Algorithm 14. It is based on two separate tasks. Task 1 is responsible for processing transmission requests from the gossip layer and message deliveries from the transport layer. Task 2 runs in background, and performs requests for messages that are known to exist (due to the reception of IHAVE messages), but whose payload has not yet been received. Furthermore, the module maintains the following data structures: a set R of messages

Algorithm 14: Point-to-point communication

```
Data:
      $C[\,]$ //cached data
      $R$ //known messages

1:  initially
2:        $\forall i : C[i] \longleftarrow \perp$
3:        $R \longleftarrow \emptyset$

4:  Task 1:
5:        proc L-Send($i$,$d$,$r$,$p$) do
6:              if Eager?($i$,$d$,$r$,$p$) then
7:                    Send(Msg($i$,$d$,$r$,$p$))
8:              else
9:                    $C[i] \longleftarrow (d,r)$
10:                   Send(IHave($i$),$p$)

11:       upon Receive(IHave($i$),$p$) do
12:             if $i \notin R$ then
13:                   Queue($i$,$s$)

14:       upon Receive(Msg($i$,$d$,$r$),$s$) do
15:             if $i \notin R$ then
15:                   $R \longleftarrow R \cup \{i\}$
16:                   Clear($i$)
17:             trigger L-Receive($i$,$d$,$r$,$s$)

18:       upon Receive (IWant($i$),$s$) do
19:             ($d$,$r$) $\longleftarrow C[i]$
20:             Send(Msg($i$,$d$,$r$),$p$)

21: Task 2:
22:       forever do
23:             ($i$,$s$) $\longleftarrow$ ScheduleNext()
24:             Send(IWant($i$),$s$)
```

whose payload has been received and; a map C, holding the payload and round number for the message (if known).

This module operates as follows: When a message is sent (line 5), the Transmission Strategy module is queried to test if the message should be immediately sent (line 7). If not, an advertisement without the payload is sent instead (line 10). Upon receiving a message advertisement for an unknown message, the Transmission Strategy module is notified (line 13). Upon receiving full message payload, the strategy module is informed (line 16) and the message is also handed over to the gossip layer (line 17). Finally, when a node receives a request (line 18) it looks it up in the cache and transmits the payload (line 20). Note that a retransmission request can only be received as a consequence of a previous advertisement and thus the message is guaranteed to be known and stored locally.

Task 2 executes the following loop. The Transmission Strategy module is invoked to select a message to be requested and a node to request the message from (pair (i,s) in line 24). This invocation blocks until a request is scheduled to be sent by the Transmission Strategy module. A request is then sent (line 25).

For simplicity, we again do not show how cached identifiers and payloads are removed from C and R, preventing them from growing indefinitely. This is however similar to the management of set K, discussed in the previous section, and thus the same techniques may be applied.

Finally, the goal of the Oracle module is to measure relevant performance metrics of the participant peers and to make this information available to the Transmission Strategy in an abstract manner. The exported interface of this module is simply composed by the $\textsc{Metric}(p)$ method, that returns the current metric value for a given peer p. This metric is used by the Transmission Strategy to select whether to immediately schedule an eager transmission or when to request lazy push transmissions from each source. Note that, the Oracle module may be required to exchange messages with its peers (for instance, to measure round trip delays). However, this communication does not affect the dissemination process and moreover, for optimization, can be piggy-backed in regular gossip messages.

In the following section we discuss different implementations of the Transmission Strategy and of Oracle modules, which aim at achieving different emergent dissemination structures.

4.3 Strategies and Oracles

The definition of a Transmission Strategy has two main objectives: (i) Avoid as much as possible redundant transmissions of the same payload to any given target node; (ii) Decrease the global latency for message dissemination and delivery. These goals are however conflicting. The first goal can be achieved by using a lazy push strategy in all peer exchanges. Since nodes only gossip IHAVE messages, the recipient can request the payload only once. Unfortunately, each lazy push exchange adds one additional round trip to the final delivery latency. On the other hand, a pure eager push strategy minimizes latency at the cost of generating a significant amount of redundancy.

The key to obtaining a better latency/bandwidth trade off in applying a clever and decentralized strategy to select which nodes (and therefore, links) should be preferred. To help the reader in assessing the goal of strategies, we start by describing a couple of strategies that do not take advantage of knowledge about the environment, which can also be used as a baseline for evaluating the benefits of more complex strategies.

4.3.1 Strategies

Flat The flat strategy is defined as $\textsc{Eager?}(i,d,r,p)$ returning true with some probability π or false with probability $1-\pi$. When π equal to 1, this strategy implements a fully eager push gossip. On the other hand, with π equal to 0, it provides a fully pure lazy push gossip. with π values between 0 and 1, it provides different latency/bandwidth trade offs, as a different share of gossip messages are handled in a lazy fashion.

When a lazy strategy is used (and IHAVE messages are sent), we need also to consider how retransmissions are scheduled by receivers within the SCHEDULENEXT() procedure. In the Flat strategy, the first retransmission request is scheduled immediately when queued, which means that an IWANT message is issued immediately upon receiving an IHAVE advertisement. Further requests are done periodically every T, while additional sources are known. Notice that, this is required to mask the failure of the node which sent the first received IHAVE advertisement.

The value of T is an estimate of maximum end-to-end latency. This avoids issuing explicit transmission requests until all eager transmissions have been performed, thus optimizing bandwidth. Note that, unless there is a network omission or an extreme transmission delay, there is usually no need to issue a second request. Thus the value of T has no practical impact in the final average latency, and can be set using only an approximation to the real end-to-end latency.

Although the T value is based only on the (current) end-to-end latency, the reader should notice that, as stated before, this transmission strategy logic has a pure probabilistic nature, as it does not rely in any knowledge about the execution environment (e.g., the transmission strategy does not rely in information provided by any Oracle).

Time-To-Live (TTL) This strategy uses eager push until some gossip round u and is thus defined as EAGER?(i,d,r,p) returning true if, and only if, $r < u$. When $u > t$, this strategy defaults to a simple lazy push gossip. With $u = 0$, it provides pure lazy push gossip. With a u value between 0 and t, it provides different latency/bandwidth trade offs, as it results, similar to the previous strategy, in a different share of gossip messages being handled in a lazy fashion. SCHEDULEDNEXT() is defined exactly as in the Flat strategy.

Notice that this strategy is intuitively useful: During the first rounds, the likelihood of a node being targeted by more than one copy of the payload is small and thus there is no point in using lazy push. Moreover, if the supporting unstructured overlay network presents a low clustering coefficient, the usefulness of this strategy is increased, as it will lower the number of redundant peers selected for gossip in the initial dissemination rounds.

Radius This strategy is defined as EAGER?(i,d,r,p) returning true if, and only if, METRIC$(p) < \rho$, for a given peer p, has a lower value than a given constant radius ρ. As described before, METRIC(p) is provided by an Oracle. module for node p. SCHEDULEDNEXT() delays the first retransmission by some time T_0, which is an estimate of the latency to nodes within radius ρ for the given metric. Further retransmissions are scheduled periodically with period T, as in the Flat strategy. However, if multiple sources are known, the nearest neighbor (according to the Oracle) is selected for requesting the payload.

This follows the intuition of gossiping first with close nodes to minimize hop latency. The expected emergent structure should approximate a mesh structure, with most of payload being carried by links between close neighboring nodes.

Ranked This strategy aims at achieving a hubs-and-spokes structure by selecting a set of *best nodes* to serve as hubs, bearing most of the load. Therefore, at some node q, EAGER?(i,d,r,p) returns true if, and only if, either q or p are considered to be best nodes, meaning that eager push is used whenever a best node is involved.

In such a way, more powerful nodes will have an increased probability to receive payload messages through eager push, as they only rely in this transmission mode to forward the message to all their selected peers. SCHEDULEDNEXT() is defined exactly as in the Flat strategy.

Although some nodes can be explicitly configured as *best nodes*, for instance, by an Internet Service Provider (ISP) that wants to improve performance to local users, a ranking can also be computed using local Oracles and a gossip based sorting protocol [13]. As shown later, this is greatly eased by the fact that the protocol still works even if ranking is approximate.

4.3.2 Oracles

In this section we provide a short description of two simple Oracles, which are used in the evaluation scheme (we will present experimental work further ahead in Section 4.4). Notice that, the implementation of these Oracles is orthogonal to the design of the architecture.

Other, and more complex, strategies may require different types of Oracles. Moreover, an Oracle may compute more complex metrics, for instance, that combine in some clever way more than one performance metric. These performance metrics can be both network efficiency metrics, such as latency, but also application efficiency metrics, such as content similarity in file sharing applications.

Latency Oracle Measures the latency to all neighbor nodes. Real-time monitoring of latency has been addressed a number of times, in fact, every TCP/IP connection implicitly estimates round-trip time to perform congestion control [9]. This estimate can be retrieved by applications and used for other purposes.

Distance Oracle Measures geographical distance to all neighbor nodes. This is useful mostly for demonstration purposes, as it allows to plot network usage graphs such that, the resulting emergent structure, is understandable by the reader. Otherwise, it is not useful in optimizing network parameters.

4.4 Performance

In this section we show performance values for the gossip optimization protocol. Unlike the simulation setting used in Section 3.4, here we use an experimental setting based on emulation.

4.4.1 Network Emulation

Experimental evaluation of the protocol for building emergent structures by optimizing gossip, was conducted using the ModelNet large-scale emulation infrastructure [40] with a realistic network model generated by Inet-3.0 [42]. In detail, ModelNet allows a large number of virtual nodes running unmodified programs to be configured in a smaller number of physical nodes in a LAN. Traffic is routed through a set of emulator nodes thus applying delay, bandwidth, and loss as specified in the network model. Inet-3.0 generates realistic Autonomous System level network topologies using a transit-stub model.

ModelNet was deployed in a cluster of 5 workstations connected by switched 100 Mbps Ethernet. Each workstation has a 2.4 GHz Intel Celeron CPU, 512 MB RAM, and a RealTek Ethernet controller. When hosting virtual nodes, they run Linux kernel 2.6.14 and IBM Java2 1.5 runtime. When running as an emulator, FreeBSD 4.11 is used.

The network model is generated using Inet-3.0 default of 3037 network nodes. Link latency is assigned by ModelNet according to pseudo-geographical distance. Client nodes are assigned to distinct stub nodes, also with the default 1 ms client-stub latency. A typical network graph has the following properties: average hop distance between client nodes is 5.54, with 74.28% of nodes within 5 and 6 hops; average end-to-end latency of 49.83 ms, with 50% of nodes within 39 ms and 60 ms.

4.4.2 Implementation and Configuration

The protocol was implemented by modifying an open source and lightweight implementation of the NeEM protocol [34] that uses the java.nio API for scalability and performance [38]. Briefly, NeEM uses TCP/IP connections between nodes to avoid network congestion. When a connection blocks, messages are buffered in user space, which then uses a custom purging strategy to improve reliability. The result is a virtual connection-less layer that provides improved guarantees for gossiping.

This implementation was selected as NeEM 0.5 already supports eager and lazy push, although the later is selected only based on a message size and in a round value threshold. The change required was to remove the hard-coded push strategy and insert the scheduler layer. Message identifiers are probabilistically unique 128 bit strings.

The protocol was configured with a gossip fanout of 11 and an overlay degree of 15. With 200 nodes, these correspond to a probability 0.995 of atomic delivery with 1% messages dropped, and a probability of 0.999 of keeping the overlay connected when 15% of nodes fail [6]. A retransmission period of 400 ms was used, which is the minimal value that results in approximately 1 payload received by each destination when using a fully lazy push strategy.

4.4.3 Traffic and Measurements

During each experiment, 400 messages are multicast, each carrying 256 bytes of application level payload. To each of them, a NeEM header of 24 bytes is added, besides TCP/IP overhead. Messages are multicast by virtual nodes in a round-robin fashion, with an uniform random interval with 500 ms average. All messages multicast and delivered are logged for later processing. Namely, end-to-end latency can be measured when source and destination share the same physical node, and thus a common clock.[5] Payload transmissions on each link are also recorded separately.

Results presented in the following section used 25 virtual nodes on each workstation, thus obtaining 100 virtual nodes. The reason for this limitation is that an epidemic multicast protocol produces a bursty load, in particular when using eager push gossip: Network and CPU load occurs only instants after each message is multicast. Using a larger number of virtual nodes was observed to induce additional latency which would falsify results. The configurations that result in lower bandwidth consumption, which are the key results and goals of this approach, were also simulated with 200 virtual nodes.

4.4.4 Statistics

Consider the following statistics of each experiment with 100 virtual nodes using an eager push strategy: 40000 messages delivered, 440000 individual packets transmitted. This amounts to 2200 Kpkts/s and thus approximately 6 MBps. As the overlay evolves, TCP/IP connections are created and tear down. During each run, approximately 550 simultaneous and 15000 different connections are used. The experiments presented in the next section, when automated, take almost 7 hours to run. The total amount of resulting logs is 1Gb, that has then to be processed and rendered in plots.

Special care was taken to consider variance of each measure taken. When in the following section we affirm that a performance difference is relevant, this was confirmed by checking that confidence intervals with 95% certainty do not intersect. In fact, the large number of samples used are sufficient to make such intervals very narrow.

4.4.5 Experimental Results

To strengthen the intuition behind the approach, we depict the impact of the described strategies with the Oracle that considers pseudo-geographical position of nodes, as generated by Inet-3.0. Although this cannot be used to assess the performance of the protocol, as geometrical distance does not directly map to end-to-end distance, it allows the resulting emergent structure to be plotted and understood.

[5] We do not present such results in this chapter however, the interested reader can refer to [2] for more information.

Figures 8–10 shows the result of running 100 node configurations with different strategies and then selecting the top 5% connections with highest throughput. The size of each red circle is proportional to the amount of payload transmitted by the node. Note that each connection is used for a brief period of time, as the membership management algorithm periodically shuffles peers with neighbors. This means that connections shown may have not existed simultaneously.

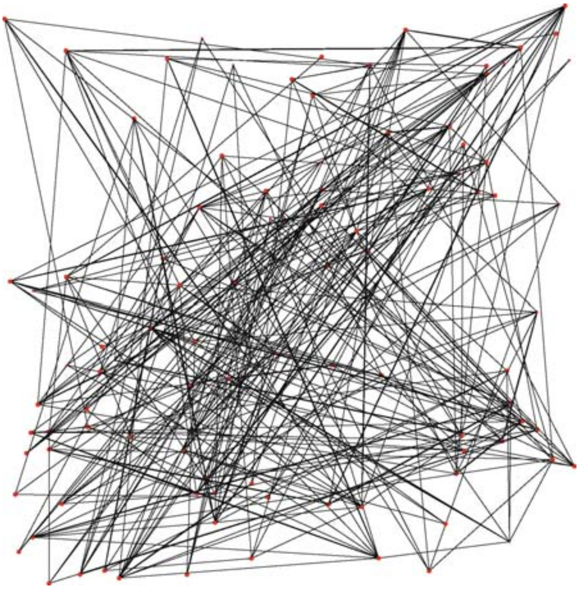


Fig. 8 Flat (7% of traffic)

As a baseline, Fig. 8 shows an eager push configuration, where no structure is apparent. A confirmation of this is given by the fact that the top 5% connections account for only 7% of all traffic, i.e., payload transmissions are evenly spread across all connections. In sharp contrast, Fig. 9 shows an obvious emergent mesh structure as a result of the Radius strategy, in which the 5% connections account for 37% of all payload transmissions. Finally, Fig. 10 shows a sub-set of nodes emerging as super-nodes, accounting for a large share of links and also transmitting a higher number of payloads. Again, the emergent structure is confirmed by the fact that 5% of the connections account for 30% of total payloads transmitted.

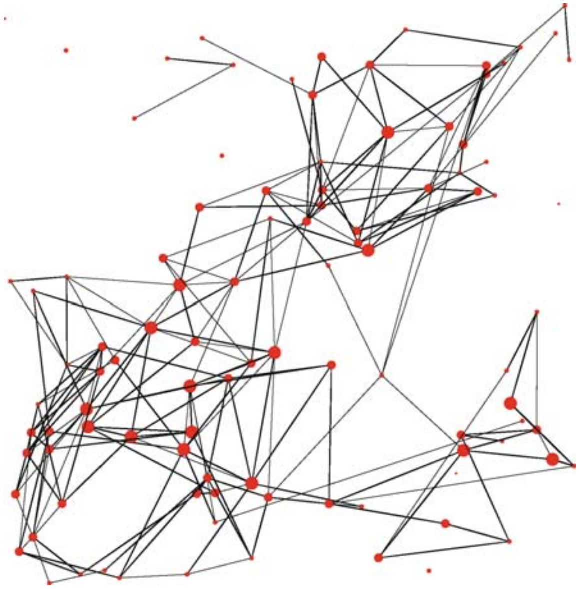


Fig. 9 Radius (37% of traffic)

5 Discussion and Future Directions

In this chapter we discussed how to add some structure to unstructured overlay networks. We have identified two distinct techniques that can be used for this end, namely: a technique based on optimizing the overlay and; a technique based on optimizing gossip itself. These techniques can be applied together as each of them operates at distinct levels in the typical architecture of gossip protocols. We have described in some detail one protocol for each of these techniques. The first protocol illustrates how to control the communication patterns to adapt the unstructured topology of the overlay, into a more structured and efficient infrastructure for peer-to-peer applications and protocols. The second protocol illustrates how the control of communication modes among peers can be used to create emergent communication structures in unstructured overlay networks. Moreover, we showed that by exploiting these emergent patterns, one can lower the typical communication overhead of gossip protocols, by applying according to different strategies, a combination of eager push and lazy push when gossiping with neighbors.

As future directions for research, we identify the following open issues that are raised when adding structure to unstructured overlay networks:

Unstructured overlay networks are a promising approach to develop high scale and highly resilient monitoring systems. Notice that one can map on top of the overlay links, monitoring relations. Therefore, one ensures that all components in the system (which are mapped on nodes) are constantly monitored by a given number of other components, and that every component shares the monitoring load.

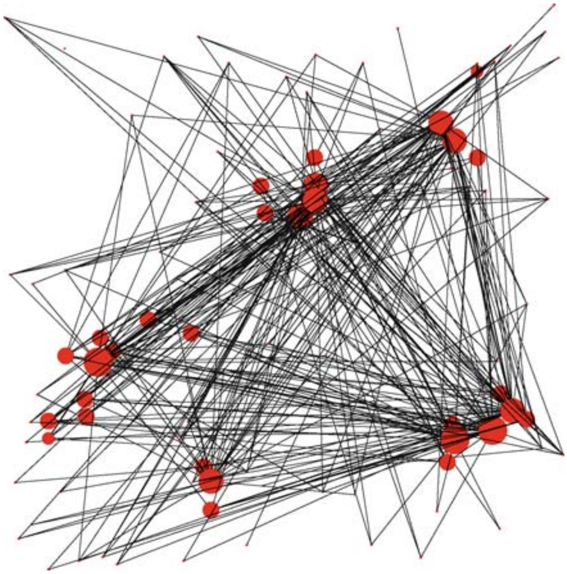


Fig. 10 Ranked (30% of traffic)

Moreover, adapting the overlay structure can be useful to promote efficient methods to disseminate monitoring information.

Location algorithms for unstructured overlay networks present a high overhead, specially compared with location algorithms for structured approaches such as DHTs. Usually, these algorithms degenerate in some kind of limited flooding in the overlay which presents a high communication overhead. On the other hand, unlike structured approaches, unstructured overlay networks can support more complex search semantics (as they do not operate based on hash keys) and can better tolerate churn scenarios. One can try to bias the topology of unstructured overlay networks to support more efficient, reliable and rich semantics resource location algorithms and protocols.

Finally, given that the methodologies illustrated in this chapter operate at distinct architectural levels, a promising direction in the peer-to-peer research field is to find efficient ways of combine both methodologies.

Acknowledgments

This work was partially supported by project "P-SON: Probabilistically Structured Overlay Networks" (POS_C/EIA/60941/2004).

References

1. Birman, K., Hayden, M., Ozkasap, O., Xiao, Z., Budiu, M., Minsky, Y.: Bimodal multicast. ACM Trans. Comput. Syst. **17**(2), 41–88 (1999)
2. Carvalho, N., Pereira, J., Oliveira, R., Rodrigues, L.: Emergent structure in unstructured epidemic multicast. In: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, p. (to appear). Edinburgh, UK (2007)
3. hua Chu, Y., Rao, S.G., Zhang, H.: A case for end system multicast (keynote address). In: SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pp. 1–12. ACM, New York, NY, USA (2000). DOI http://doi.acm.org/10.1145/339331.339337
4. Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., Bowman, M.: Planetlab: an overlay testbed for broad-coverage services. SIGCOMM Comput. Commun. Rev. **33**(3), 3–12 (2003). DOI http://doi.acm.org/10.1145/956993.956995
5. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., Terry, D.: Epidemic algorithms for replicated database maintenance. In: PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing, pp. 1–12. ACM, New York, NY, USA (1987). DOI http://doi.acm.org/10.1145/41840.41841
6. Eugster, P., Guerraoui, R., Kermarrec, A.M., Massoulié, L.: From Epidemics to Distributed Computing. IEEE Comput. **37**(5), 60–67 (2004). DOI NA
7. Eugster, P.T., Guerraoui, R.: Probabilistic multicast. In: DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks, pp. 313–324. IEEE Computer Society, Washington, DC, USA (2002)
8. Eugster, P.T., Guerraoui, R., Handurukande, S.B., Kouznetsov, P., Kermarrec, A.M.: Lightweight probabilistic broadcast. ACM Trans. Comput. Syst. **21**(4), 341–374 (2003). DOI http://doi.acm.org/10.1145/945506.945507
9. Floyd, S., Fall, K.: Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM Trans. Netw. **7**(4), 458-472 (1999)
10. Floyd, S., Jacobson, V., Liu, C.G., McCanne, S., Zhang, L.: A reliable multicast framework for light-weight sessions and application level framing. IEEE/ACM Trans. Netw. **5**(6), 784–803 (1997). DOI http://dx.doi.org/10.1109/90.650139
11. Ganesh, A., Kermarrec, A.M., Massoulié, L.: SCAMP: Peer-to-peer lightweight membership service for large-scale group communication. In: Networked Group Communication, pp. 44–55 (2001). URL `citeseer.ist.psu.edu/ganesh01scamp.html`
12. Hayden, M., Birman, K.: Probabilistic broadcast. Tech. rep., Cornell University, Ithaca, NY, USA (1996)
13. Jelasity, M.: A case study on gossip beyond gossip: Sorting. Ws. on Gossip Based Computer Networking, Lorent Center, Leiden (2006)
14. Jelasity, M., Babaoglu, O.: T-man: Fast gossip-based construction of large-scale overlay topologies. Tech. rep., University of Bologna (2004)
15. Jelasity, M., Babaoglu, O.: T-man: Gossip-based overlay topology management. In: The Fourth International Workshop on Engineering Self-Organizing Applications (ESOA'06). Hakodate, Japan (2006). URL `http://dx.doi.org/10.1007/11734697_1`
16. Jelasity, M., Guerraoui, R., Kermarrec, A.M., van Steen, M.: The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In: Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, pp. 79–98. Springer-Verlag New York, Inc., New York, NY, USA (2004)
17. Jelasity, M., Montresor, A.: Epidemic-style proactive aggregation in large overlay networks. In: Proceedings of The 24th International Conference on Distributed Computing Systems (ICDCS 2004), pp. 102–109. IEEE Computer Society, Tokyo, Japan (2004). URL `citeseer.ist.psu.edu/jelasity04epidemicstyle.html`
18. Jelasity, M., Montresor, A., Jesi, G.P., Voulgaris, S.: The Peersim simulator. `http://peersim.sf.net`

19. Karwaczynski, P.: Fabric: Synergistic proximity neighbour selection method. In: P2P '07: Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing (P2P 2007), pp. 229–230. IEEE Computer Society, Washington, DC, USA (2007)
20. Karwaczyński, P., Konieczny, D., Moçnik, J., Novak, M.: Dual proximity neighbour selection method for peer-to-peer-based discovery service. In: SAC '07: Proceedings of the 2007 ACM symposium on Applied computing, pp. 590–591. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1244002.1244137
21. Kermarrec, A.M., Massoulié, L., Ganesh, A.: Probabilistic reliable dissemination in large-scale systems. IEEE Trans. Parallel Distrib. Syst. **14**(3), 248–258 (2003). DOI http://dx.doi.org/10.1109/TPDS.2003.1189583
22. Koldehofe, B.: Buffer management in probabilistic peer-to-peer communication protocols. In: Proceedings of the 22th IEEE Symposium on Reliable Distributed Systems (SRDS'03), pp. 76–87. Florence, Italy (2003)
23. Leito, J.: Gossip-based broadcast protocols. Master's thesis, University of Lisbon (2007)
24. Leito, J., Pereira, J., Rodrigues, L.: Epidemic broadcast trees. In: Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems (SRDS'2007), pp. 301–310. Beijing, China (2007)
25. Leito, J., Pereira, J., Rodrigues, L.: HyParView: A membership protocol for reliable gossip-based broadcast. In: DSN '07: Proc. of the 37th Annual IEEE/IFIP Intl. Conf. on Dependable Systems and Networks, pp. 419–429. IEEE Computer Society, Edinburgh, UK (2007). DOI http://dx.doi.org/10.1109/DSN.2007.56
26. Leito, J., Pereira, J., Rodrigues, L.: Topology aware gossip overlays. Tech. Rep. 36, INESC-ID (2008)
27. Li, H., Clement, A., Wong, E., Napper, J., Roy, I., Alvisi, L., Dahlin, M.: BAR gossip. In: Proceedings of the 2006 USENIX Operating Systems Design and Implementation (OSDI) (2006)
28. Lin, M.J., Marzullo, K.: Directional gossip: Gossip in a wide area network. In: European Dependable Computing Conference, pp. 364–379 (1999). URL `citeseer.ist.psu.edu/237760.html`
29. Liu, Y., Xiao, L., Ni, L., Liu, Y.: Building efficient overlays. J. Grid Comput. **2**(2), 183–192 (2004)
30. Massoulié, L., Kermarrec, A.M., Ganesh, A.J.: Network awareness and failure resilience in self-organising overlays networks. In: Synmposium on Reliable Distributed Systems (SRDS). Florence, Italy (2003). URL `http://www.irisa.fr/paris/Biblio/Papers/Kermarrec/MasKerGan03SRDS.pdf`
31. Melamed, R., Keidar, I.: Araneola: A scalable reliable multicast system for dynamic environments. In: NCA '04: Proceedings of the Network Computing and Applications, Third IEEE International Symposium, pp. 5–14. IEEE Computer Society, Washington, DC, USA (2004)
32. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equations of state calculations by fast computing machine. J. Chem. Phys. **21**, 1087–1091 (1953)
33. Pereira, J., Oliveira, R., Rodrigues, L.: Efficient epidemic multicast in heterogeneous networks. In: Proceedings of the International Workshop on Reliability in Decentralized Distributed Systems, part of the OTM Federated Conferences and Workshops. Montpellier, France (2006)
34. Pereira, J., Rodrigues, L., Monteiro, M.J., Oliveira, R., Kermarrec, A.M.: NeEM: Network-friendly epidemic multicast. In: Proceedings of the 22th IEEE Symposium on Reliable Distributed Systems (SRDS'03), pp. 15–24. Florence, Italy (2003)
35. Ratnasamy, S., Handley, M., Karp, R., Shenker, S.: Application-level multicast using content-addressable networks. In: NGC '01: Proceedings of the Third International COST264 Workshop on Networked Group Communication, pp. 14–29. Springer-Verlag, London, UK (2001)
36. van Renesse, R., Minsky, Y., Hayden, M.: A gossip-style failure detection service. Tech. rep., Cornell University, Ithaca, NY, USA (1998)
37. Rowstron, A.I.T., Kermarrec, A.M., Castro, M., Druschel, P.: Scribe: The design of a large-scale event notification infrastructure. In: NGC '01: Proceedings of the Third International

COST264 Workshop on Networked Group Communication, pp. 30–43. Springer-Verlag, London, UK (2001)

38. Santos, P., Pereira, J.: NeEM version 0.5. http://neem.sf.net (2006)
39. Tang, C., Ward, C.: GoCast: Gossip-enhanced overlay multicast for fast and dependable group communication. In: DSN '05: Proc. of the 2005 Intl. Conf. on Dependable Systems and Networks (DSN'05), pp. 140–149. IEEE Computer Society, Washington, DC, USA (2005). DOI http://dx.doi.org/10.1109/DSN.2005.52
40. Vahdat, A., Yocum, K., Walsh, K., Mahadevan, P., Kostic, D., Chase, J., Becker, D.: Scalability and accuracy in a large-scale network emulator. SIGOPS Oper. Syst. Rev. **36**(SI), 271–284 (2002). DOI http://doi.acm.org/10.1145/844128.844154
41. Voulgaris, S., Gavidia, D., Steen, M.: Cyclon: Inexpensive membership management for unstructured p2p overlays. J. Network Syst. Manag. **13**(2), 197–217 (2005). DOI 10.1007/s10922-005-4441-x. URL `http://dx.doi.org/10.1007/s10922-005-4441-x`
42. Winick, J., Jamin, S.: Inet-3.0: Internet topology generator. Tech. Rep. UM-CSE-TR-456-02, EECS, University of Michigan (2002). URL `citeseer.nj.nec.com/526211.html`
43. Zhuang, S., Zhao, B., Joseph, A., Katz, R., Kubiatowicz, J.: Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In: Proceedings of NOSSDAV (2001). URL `citeseer.ist.psu.edu/zhuang01bayeux.html`

Mathematical Modeling of Routing in DHTs

Peter Kersch and Robert Szabo

Abstract Although most Distributed Hash Table (DHT) overlays are structurally similar to the “small-world” navigation model of Kleinberg – architectural and algorithmic details of different DHT variants differ significantly. Lookup performance of DHTs depends on a sets of different and often incompatible parameters, which makes analytical comparison rather difficult. The objective of this chapter is to review existing analytical models for DHT routing performance and to introduce a novel framework for the per-hop routing progress analysis of long-range DHT connections based on a logarithmic transformation. With the logarithmic transformation of the DHT metric space we analyze the distribution of the per-hop routing progress in general and also for the special cases of the deterministic and probabilistic power-law routing overlays. Based on the proposed analytical framework routing performance of DHTs can be described by a triple: the long-range connection density, its coefficient of variation and the number of short-range connections. Finally, we derive upper bound on the expected number of routing hops as a function of network size and the parameter triple.

1 Introduction

A plethora of Distributed Hash Table (DHT) concepts have been proposed and analyzed in the past 6–7 years to provide scalable and robust distributed storage and

Peter Kersch
High Speed Networks Laboratory, Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary,
e-mail: kersch@tmit.bme.hu

Robert Szabo
High Speed Networks Laboratory, Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary,
e-mail: szabo@tmit.bme.hu

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_14, © Springer Science+Business Media, LLC 2010

lookup systems [6, 12, 14, 16, 17, 19–22], etc. Although architectural and algorithmic details of these DHT proposals can differ significantly, the foundations of lookup mechanisms are very similar for most of them. There are several empirical studies (based on simulations) comparing static and dynamic performance of different DHT routing mechanisms using various parameter settings [5, 13]. There exist also detailed analytical models for some DHTs, however these models are usually restricted to one specific DHT implementation. Finally, some aspects of DHT routing are covered by generic models, e.g., static resilience of DHT routing against failure [10] or the impact of lookup strategy, lookup parallelism and replication on DHT routing performance under churn [23]. However, to the best of our knowledge, there exist no generic analytical models capturing the relationship between overlay structure and routing performance of DHTs in static networks. In this chapter, we try to fill this gap proposing a generic stochastic model of DHT overlays and overlay routing covering a large family of DHTs.

The proposed analytical model builds on the fact that most DHT overlays are structurally similar to the "small-world" model of Kleinberg [8] and the sequence of long-range connections of a DHT node becomes linear after logarithmic transformation of distances in the DHT metric space. More specifically, we have identified a large subclass of DHT overlays (regular power-law routing overlays) where this transformed sequence can be described for each node as independently selected random samples from an infinite renewal process. Using this renewal process model, we analyze the distribution of the per-hop routing progress in general and also for the special cases of the deterministic and probabilistic power-law routing overlays. Furthermore, we introduce the λ long-range connection density and the c_v long-range connection density coefficient of variation parameters to characterize long-range connection distribution of an overlay. Finally, using renewal theory, we derive upper bounds on the expected number of routing hops as a function of network size and the above overlay parameters.

The rest of this chapter is structured as follows. First, we give a brief overview of DHTs in general. Then, in Section 2, we discuss challenges of modeling DHT routing, revisit applied mathematical tools from renewal theory and introduce modeling assumptions and notations used in the upcoming sections. In Section 3, we present the concept of logarithmically transformed view for long-range connections. Finally, in Section 4, we describe the proposed stochastic model based on this transformed view and tools from renewal theory.

1.1 DHTs Revisited

From the point of view of an application, Distributed Hash Tables provide similar functionality than ordinary "in memory" hash tables. An application can insert and remove key-value mappings, and given a key, it can retrieve the associated value (in the context of a peer-to-peer system, a key is an identifier used to refer to a shared resource while the associated value is the resource itself or the locator of the resource). All of these operations are performed quickly and efficiently and scale well for large amounts of data in both "in memory" and distributed hash tables. However, as opposed to ordinary hash tables, storage of key-value pairs is distributed over all nodes of the DHT and all hash table methods can be issued from any of these nodes (see Fig. 1). Consequently, internal operation of a DHT differs significantly from the operation of ordinary "in memory" hash tables. To present the architecture and operation of distributed hash tables, we used the terminology and formalism proposed in [1].

One of the key conceptual components of a DHT is the common metric space into which nodes and resources are mapped to. All distributed hash tables use a virtual identifier space $\mathscr{I}$ which possesses a closeness metric $d : \mathscr{I} \times \mathscr{I} \to \mathbf{R}$ so that $(\mathscr{I}, d)$ is a metric space or a quasi-metric space[1] Both the group of peers forming the DHT and the set of all shared resources are mapped to this ID space $\mathscr{I}$ (see Fig. 1). Mapping of peers can be described by a function $F_P : \mathscr{P} \to \mathscr{I}$ where $\mathscr{P}$ is the set of peers forming the DHT. F_P is usually implemented by either drawing a random identifier according to uniform distribution over $\mathscr{I}$ or by applying a hash function to the public key of the peer. Resources are mapped to $\mathscr{I}$ using a function $F_K : \mathscr{K} \to \mathscr{I}$ where K is the set of keys used to refer to shared resources. F_K is most often implemented by applying a hash function to the keys.

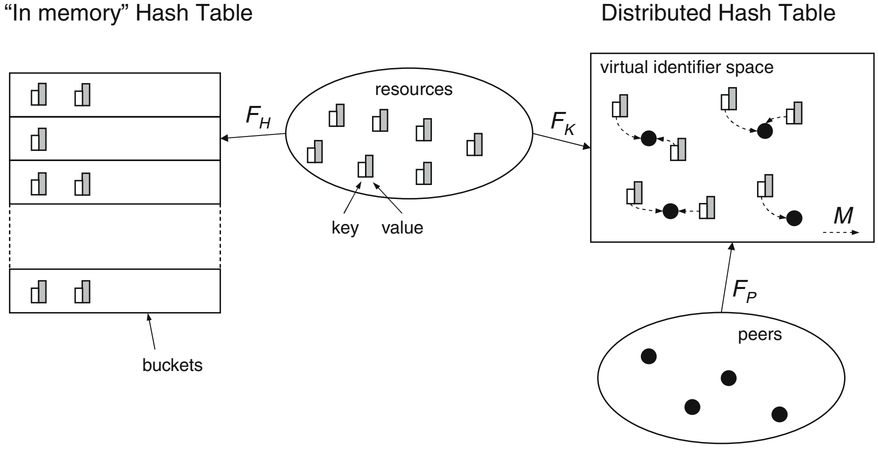


Fig. 1 Comparison of "in memory" hash tables and DHTs

Peers responsible for a given resource are determined based on the above mappings to the common metric space $(\mathscr{I}, d)$. A key-value pair describing a resource is usually stored by the peer (or the set of peers) whose image in $(\mathscr{I}, d)$ is the closest to the image of the given resource in $(\mathscr{I}, d)$. Formally, this can be described using a function $\mathscr{M} : \mathscr{I} \to 2^{\mathscr{P}}$ and a constraint $\forall i \in \mathscr{I} : \forall p \in \mathscr{M}(i), \forall q \notin \mathscr{M}(i) :$

[1] A quasi-metric space does not satisfy the symmetry requirement of metric spaces.

$d(F_P(p),i) \leq d(F_P(q),i)$ on this function. As a result, locating a key-value pair (which describes a shared resource in a DHT) corresponds to finding one of the closest peers to the image of the resource in $(\mathscr{I},d)$.

The function $\mathscr{M}$ is usually complete, which means that each identifier of $\mathscr{I}$ is under the responsibility of at least one peer. To provide fault tolerance $\mathscr{M}$ typically contains more than one element and the cardinality of $\mathscr{M}$ is typically constant, which means that each key-value pair is replicated to the same number of peers (Fig. 1 shows the simplest case when each key-value pair is stored by only one peer).

Comparing distributed hash tables to ordinary hash tables, peers correspond to buckets and the function $\mathscr{M}$ corresponds to the hash function F_H in ordinary hash tables. Changing the number of buckets in an ordinary hash table implies changing the hash function F_H and this usually requires relocation of most key-value pairs. For "in memory" hash tables, bucket size is usually constant (or changes only rarely when reaching a capacity threshold), hence this is not a problem. In contrast, a peer-to-peer network is inherently dynamic and the set of peers in a DHT might change continuously, implying changes in the mapping of key-value pairs to nodes ($\mathscr{M}$). Continuous relocation of key-value pairs in a DHT would generate a huge communication overhead, hence changes in these mappings should be minimized when peers join and leave the network. DHTs address this problem by selecting responsible peers based on proximity in the metric space $(\mathscr{I},d)$ as mentioned above. Consequently, changes in key-value pair → responsible peers mappings are restricted to the neighborhood of the joining or leaving peer in $(\mathscr{I},d)$. (This concept is also called *consistent hashing* [7] and had been proposed for distributed web caching before the era of distributed hash tables.)

Another benefit of selecting responsible peers by proximity in $(\mathscr{I},d)$ is that all DHT operations can be easily implemented on top of a routing algorithm which locates the closest peers to a given point in $\mathscr{I}$. To realize this routing process, DHTs create and maintain an overlay network. An overlay network can be modeled by a directed graph $G = (\mathscr{P},\mathscr{E})$ where $\mathscr{P}$ denotes the set of vertices (peers) while $\mathscr{E}$ denotes the set of edges (overlay connections[2]). Routing in the overlay is typically based on a simple greedy algorithm: a request for a given point in $\mathscr{I}$ is forwarded via the connection pointing to the peer which is the closest to this given point in $(\mathscr{I},d)$.

Overlay topology depends heavily on distances between the images of peers in the metric space $(\mathscr{I},d)$. In most DHT overlays, connections can be categorized into short-range (local) and long-range connections. Each node has short-range connections to some specific subset of the closest peers in $(\mathscr{I},d)$. Additionally, they have long-range connections to some distant nodes so that the distribution of these connections is structurally similar to the family of small-worlds graphs introduced by Kleinberg in [8]. In this small-worlds graph family, the probability of having a long-

[2] A connection from node v_1 to a peer node v_2 means that node v_1 knows the address of node v_2 (this is usually in the form of a pair of ID + IP address / port number). Different algorithms use different names for connections, e.g., Chord [22] calls them successors, predecessors and finger pointers, while in Pastry [21], they are called leaf set and routing table entries, etc.

range connection between to nodes is inversely proportional to the Dth power of their distance (where D is the dimension of the metric space), and Kleinberg has shown that this is necessary to provide efficient distributed search based solely on local information.

The role of short-range and long-range connections in the overlay is complementary. Short-range connections guarantee success of greedy forwarding: since each node is connected to its closest neighbors in $(\mathscr{I}, d)$, it is always possible to forward requests at least a small step closer to the target. In contrast, long-range connections are not critical for successful routing but they expedite the lookup process and usually provide $O(\log n)$ bounds on the average number of lookup hops. This is achieved by ensuring that the distance from the target decreases by a constant factor in expected value after each routing step.

2 Modeling DHT Routing

Defining models and metrics to describe performance of different DHT routing architectures is not a trivial task. An application using the DHT lookup service is mostly interested in lookup latencies and in the ratio of successful lookups. A user running DHT implementations might also be concerned by resource usage (CPU, memory, storage, bandwidth, etc...) while a network operator is only interested in the overall traffic (lookup + control) generated in the network. Since most of these describe conflicting objectives, comparison only makes sense if conflicting performance metrics are analyzed together describing fundamental trade-offs.

Some of the commonly used performance metrics (e.g., overlay network diameter, node state) are not directly relevant for neither applications nor users nor network operators. In [25], the author investigates the trade off between node state and overlay network diameter. Loguinov et al. also use network diameter as the primary metric for routing in [15].

Node state affects primarily memory usage at nodes. However, the amount of memory required to keep track of connections is typically far from being a bottleneck in current systems. Node state can also influence the maintenance bandwidth (e.g., in DHTs using per connection periodic keep-alive messages to detect connection failures). However, it cannot be used as a general metric to characterize maintenance traffic.

Overlay network diameter can be used to derive only lower bounds on the worst-case number of routing hops for a lookup in a given overlay structure. Short paths between nodes do not guarantee that a distributed routing algorithm is also able to find them [8]. Hence, the distribution or the average number of routing hops is a more informative performance metric which also allows to derive [23] lookup latency – a key performance metric from a user perspective.

Analytical comparison of a performance metric (e.g., the number of routing hops) of different DHTs is usually described by asymptotic notation, commonly used to characterize algorithm complexity. E.g., CAN [19] with a D dimensional identifier space provides lookups in $O(\frac{D}{2}n^{\frac{1}{D}})$ hops in a network of n nodes. Although this is a useful and simple way to determine scalability of a particular algorithm, it has its limitations. Due to potentially different unknown constants hidden within the notation, it is not possible to compare two different algorithms with the same asymptotic behavior (e.g., $O(\log n)$ hop count is typical for many DHTs). Furthermore, it is also possible that an algorithm with better asymptotic behavior performs worse for practical network sizes.

Asymptotic notation may even be misleading when not used carefully. The paper presenting Koorde [6] (a DHT based on de Bruijn graphs) is a good example of such a misuse. Using a base-k de Bruijn graph, Koorde completes routing in $O(log_k\ n)$ hops. Based on this, the authors claim that choosing $k = \log n$, routing cost is $O(\log n/\log k) = O(\log n/\log\log n)$. However, the base of the underlying de Bruijn graph cannot be changed on the fly as the network grows since this would require rebuilding the whole DHT from the scratch. Therefore the parameter k should not be treated as a function of network size. (Similarly, the dimension D of a CAN [19] network is not expressed as a function of network size because this is also a parameter that cannot be changed without rebuilding the whole system.) As a consequence, the number of routing hops for Koorde using base-k de Bruijn graphs is in fact $O(\log n/\log k)$.

For a few DHT architectures, there are some exact analytical results: e.g., the average number of routing hops for Chord [22] is $\frac{1}{2}log_2 n$. [23] is one of the few papers which provide a generic analytical framework for the performance comparison of different DHTs. Given the average number of routing hops in static networks, the authors analyze the influence of three key factors on routing performance under churn: lookup strategy, lookup parallelism and replication. Our results on the expected number of routing hops in static networks can be potentially used as an input for this analytical framework to derive these additional performance metrics.

Finally – although not directly related to distributed hash tables – the "small-world" navigation model of Kleinberg [8] is a fundamental contribution to theory of routing in distributed systems. A network is said to be "small-world" when there exists a short path between any two nodes, although most nodes are not directly connected. This low network diameter is a necessary but not sufficient property for efficient distributed routing. In [8], Kleinberg investigates requirements on overlay topology for efficient distributed routing based solely on local information in small-world networks. Similarly to DHT overlays, he defines a graph (embedded into a metric space) with short-range connections to the closest nodes and long-range connection(s) to some distant nodes. As in DHTs, Kleinberg's routing is greedy: requests are forwarded via the peer node being the closest to the target node in the metric space. As the main finding of the paper, Kleinberg shows that distributed routing will achieve the best asymptotical performance when the probability of having a long-range connection to another node is inversely proportional to the D^{th} power of distance of the two nodes (where D is the dimension of the metric space embedding the small-world graph).

Most DHT routing architectures – although not inspired by Kleinberg's work – can be related to the one dimensional Kleinberg small-world model.

2.1 Renewal Processes Revisited

Renewal processes are a special class of stochastic processes used to model independent identically distributed occurrences. Let $X_1, X_2, X_3, ...$ be independent identically distributed (*i.i.d*) and positive random variables defined by the distribution function $P(X < x) = F(x)$. Furthermore, let T_n be defined as $T_n = \sum_{i=1}^{n} X_i$. Then the counting process $Y(t) = \max\{n : T_n \leq t\}$ is a renewal process ($t \geq 0$).

Renewal processes are usually defined in the time domain. In the time domain, $Y(t)$ denotes the number of events until time t, T_n corresponds to the occurrence time of the n^{th} event and the random variables X_i correspond to inter-arrival times between subsequent events. The name renewal process is motivated by the fact that every time there is an occurrence, the process "starts all over again"; it renews itself (since the variables X_i are *i.i.d*).

In contrast to the general usage, renewal processes in my dissertation are not defined in the time domain but in the distance domain of a one dimensional metric space. Furthermore occurrences are not events but the images of long-range connections in this metric space and the random variables X_i correspond to distances between the images of subsequent long-range connections.

In the followings, I briefly list the results of renewal theory that I use in the upcoming sections (for further reading, see [4, 9, 11]). Note that the vocabulary of renewal theory traditionally assumes a time domain for renewal processes. However, all results are equally valid for the distance domain too.

Renewal function The expected value of the number of arrivals in function of the elapsed time is called renewal function: $m(t) = E[Y(t)]$.

Residual life Picking a random point in time (t), the random variable corresponding to the time from this point until the next event (at time $T_{Y(t)+1}$) in a renewal process is called residual life:

$$V(t) = T_{Y(t)+1} - t \tag{1}$$

Residual life is also called *residual lifetime*, *residual time* or *forward recurrence time*.

Expected value of asymptotic residual life The expected value of asymptotic residual life in a renewal process can be expressed as

$$\lim_{t \to \infty} E[v] = \frac{\mu_2}{2\mu}. \tag{2}$$

where $\mu = E[x]$ is the expected value of inter-arrival times and $\mu_2 = E[x^2]$ is the second moment of inter-arrival times.

Distribution of asymptotic residual life Considering a renewal process with an inter-arrival time distribution $F(x)$, the probability density function of asymptotic residual life can be expressed as

$$\lim_{t \to \infty} g(v) = \frac{1 - F(v)}{\mu}, \tag{3}$$

where $\mu = E[x]$ is the expected value of inter-arrival times.

Length of a randomly selected renewal period Picking a random point in time (t) in a renewal process, the *pdf* of the length of the renewal period marked by this point ($T_{Y(t)+1}$ - $T_{Y(t)}$) is asymptotically:

$$\lim_{t\to\infty} h(x') = \frac{f(x')x'}{\mu}, \tag{4}$$

where $f(x)$ is the *pdf* of inter-arrival times and $\mu = E[x]$ is the expected value of inter-arrival times in the renewal process. It is important to note that the distribution of x' and x are not the same since a random point in time will select longer periods at higher probability than shorter periods.

Note that considering a random sample from a renewal process, the above formulas are also valid in general, not only for the asymptotic case.

Lorden bound The renewal function of a renewal process is upper bounded by

$$m(t) \leq \frac{t}{\mu} + \frac{\mu_2}{\mu^2} + 1, \tag{5}$$

where μ is the expected value of inter-arrival times and μ_2 is the second moment of inter-arrival times in the renewal process (see [4], page 110.)

Poisson processes are a special class of renewal processes. Inter-arrival times in a Poisson process are exponentially distributed. A Poisson process can be characterized by the λ parameter of this exponential distribution which is also called the intensity of the process.

A Poisson process of intensity λ can also be defined as a pure birth process: the probability that an arrival occurs during an infinitesimally small interval dt is λdt (independent of arrivals outside this interval) and the probability that more than one arrival occurs is $o(dt)$. This definition is equivalent with the renewal process definition.

Random sampling Random and independent sampling of events with probability p from a Poisson process of rate λ results into a Poisson process of rate $p\lambda$.

Superposition Superposition of two Poisson process of rate λ_1 and λ_2 respectively results into a Poisson process of rate $\lambda_1 + \lambda_2$.

2.2 Assumptions and Notations

To describe the routing overlay of distributed hash tables, we reuse the terminology and reference model defined in [1]. Let's consider a one dimensional Euclidean metric space within the interval $[0,1)$ that wraps around (this can be represented as a ring, see Fig. 2). Distance between two nodes in this metric space is defined as their distance along the ring in clockwise direction,[3] formally: $d(x,y) = y - x + I_{x>y}$.

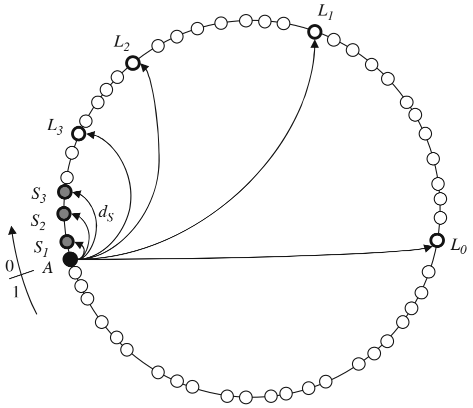


Fig. 2 Model of unidirectional DHT overlays (example)

Each node has two different types of connections to other nodes: short-range connections (called “local” connection in [1]) to a fixed number (N_S) of closest nodes (in clockwise direction) and long-range connections to some distant nodes. These nodes are called short-range and long-range peers of the node, respectively. Fig. 2 shows short-range connections (S_1, S_2, S_3) and long-range connections (L_0, L_1, L_2, L_3) of node (A). The distance of the node and its farthest short-range peer is denoted by d_S.

Routing is assumed to be greedy: a node forwards a lookup request to its peer being the closest to the target node in the metric space of the DHT (without overshooting it). This greedy routing process can be described by the following pseudo-code algorithm:

[3] This definition implies that the metric space is in fact only a quasi-metric space, since it does not satisfy the symmetry requirements. Extending the model to bidirectional routing where distance is defined as the shortest path along the ring (in any of the two directions), a real metric space can be obtained.

```
while node $\neq$ target do
    proxy $\leftarrow$ GetClosestPeer(node,target);
    if Distance(proxy,target) $<$ Distance(node,target) then
        node $\leftarrow$ proxy;
    else
        error
    end
end
```

Algorithm 15: Greedy overlay routing

Routing overlay of many DHT implementations (Chord [22], Pastry [21], Symphony [16], Accordion [12] etc.) can be described (or approximated) using the above system model (e.g., routing in Pastry is more complex but is based on the same greedy algorithm). However, there are a few exceptions, for example DHTs using multidimensional metric spaces (e.g., CAN [19]) or non-Euclidean metric spaces (e.g., Kademlia [17]).

2.2.1 Degree of Randomness

Since randomness and flexibility in the choice of long-range connections plays an important role in both analysis and maintenance of overlays, let us first define two extreme DHT overlay categories:

Definition 1 (Probabilistic power-law routing overlay (PPLRO)). A routing overlay is called probabilistic power-law routing overlay when the choice of long-range connections is not deterministic and they only have to satisfy the following requirements: the probability of having a long-range connection to another overlay node is inversely proportional to the Dth power of the distance between the two nodes in the D dimensional metric space $(\mathscr{I},d)$ where the DHT maps node identifiers [8]. Join algorithm of probabilistic power-law routing overlays create initial long-range connections of joining nodes according to this distance distribution and the choice of long-range connections is mutually independent of each other.

Definition 2 (Deterministic power-law routing overlay (DPLRO)). A routing overlay over a one-dimensional metric space[4] is called deterministic power-law routing overlay if long-range connections are determined by the power series of the distances $d_i = \frac{q}{c^i}$ where c and q are constant so that $c > 1$ and $0 < q \leq 1$. For unidirectional overlays, the ith long-range connection is chosen as the first node, whose distance exceeds d_i while for bidirectional overlays, the ith connection is the node closest to the point at distance d_i.

[4] Extending this definition to multidimensional metric spaces on the analogy of probabilistic power-law routing overlays is not trivial.

Symphony [16], Accordion [12] and the routing scheme proposed in [2] are using probabilistic routing overlays while a deterministic power-law routing overlay can be thought of as a generalization of the Chord [22] overlay (for Chord, $c = 2$).

It is important to note that the term "power-law" is also used to denote the overlay of unstructured P2P systems structurally similar to scale-free random graphs [3]. In that context, it refers to distribution of node degree. However, in Definitions 1 and 2, the term "power-law" refers to distance distribution of long-range connections.

2.2.2 Bidirectional Overlay Model

The unidirectional overlay and routing model presented above can be easily extended to bidirectional overlays with bidirectional routing. In a bidirectional overlay, both short-range and long-range connections are bidirectional. Another important difference is the distance metric of the space $(\mathscr{I},d)$. Using the ring representation, distance of two nodes is defined as their shortest distance along the ring, formally: $d_b(x,y) = +min\,[d(x,y),d(y,x)]$. Hence in contrast to unidirectional overlays, this is a real metric space also satisfying the symmetry requirements. As a result, from the point of view of distances, each node can split the DHT metric space $(\mathscr{I},d)$ into two symmetrical partitions. Connections of a node are created independently in both of these partitions. Figure 3 shows short-range and long-range connections of node A in both partitions of the metric space for an example bidirectional overlay.

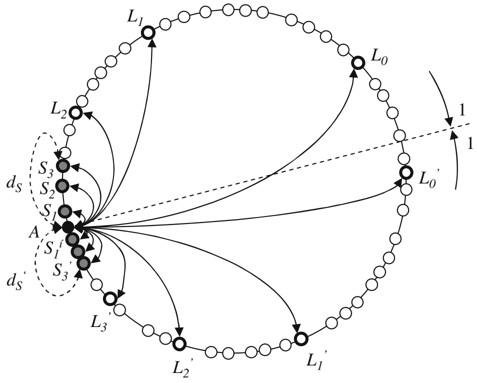


Fig. 3 Model of bidirectional DHT overlays (example)

Setting the circumference of the ring to 2 units for bidirectional overlays, it is easy to derive the bidirectional equivalent of any unidirectional overlay (where the circumference of the ring is set to 1 unit). Connections of the bidirectional overlay obey the distance distribution of the corresponding unidirectional overlay separately in both partitions of the metric space $(\mathscr{I},d)$. Hence, the definition of probabilistic and deterministic power-law routing overlay can be extended for bidirectional connections by applying either Defintions 1 or 2 separately for both partitions of the metric space.

As a consequence of the definition of distance metric for bidirectional overlays, greedy routing also becomes bidirectional; requests can be forwarded in both directions depending on the position of the peer node being closest to the target.

2.2.3 Node ID Distribution

The distribution of node identifiers in the metric space $(\mathscr{I},d)$ also affects mathematical analysis of routing in the overlay. The ID space of node identifiers is discrete and finite for most real systems, hence nodes can only be mapped to a finite subset of points in the Euclidean metric space $(\mathscr{I},d)$. However, the granularity of this finite ID space is so fine (the size of the ID space varies between $2^{128} - 2^{256}$), that node ID mapping can be considered continuous in $(\mathscr{I},d)$.

Furthermore, we assumed that given a network of n nodes, node identifiers are drawn independently at random according to a uniform distribution over the range $[0,1)$ in $(\mathscr{I},d)$ (this is a reasonable assumption in most cases). This implies that distances between adjacent IDs on the ring will be exponentially distributed. In a few cases (explicitly noted), we assumed that peer identifiers partition the metric space $(\mathscr{I},d)$ deterministically in equal partitions. This is not a realistic scenario, but simplifies considerably mathematical analysis. In these later cases, we have always compared analytical results using deterministic identifier assignment to simulation results using random uniform distribution of peer identifiers.

Finally, for long-range connection selection in probabilistic power-law routing overlays, we assume that it is possible to find a peer node at any given distance (drawn according to a given distribution) in the metric space. This is not realistic in a real system composed of a finite number of nodes. In practice, the closest existing node to the given point is used instead. However, the resulting error between these theoretical and real distances is inversely proportional to the size of the network, hence this is negligible for large networks (which are in the main scope of this analysis).

3 Transformed View of Long-Range Connections

Transformation is a widely used mathematical concept in many disciplines to reveal, analyze and exploit hidden system characteristics. One of the best known examples of the application of a transformation method is JPEG encoding where discrete cosine transform maps a 8x8 pixel area into spatial frequency components [18]. In this example, transformation is used to exploit “hidden characteristic” of human vision being much more sensitive to small variations in color and in brightness for lower spatial frequencies than for higher frequencies. Hence higher spatial frequency components can be encoded at smaller resolutions. In our analysis, we apply a logarithmic transformation to distances between node identifiers in the metric space of a DHT to reveal "hidden characteristics" of DHT routing.

Definition 3 (Logarithmically transformed view). Let $(\mathscr{I}, d)$ be the metric space of a DHT (see Section 1.1) where the distance between the image $x_0 = F_P(p_0)$ of a node p_0 and the image $x_i = F_P(p_i)$ of another node p_i is defined as $d(x_0, x_i)$. Then, using the transformation function $f_t(u) = -\ln u$, the distance of p_0 and p_i in the logarithmically transformed view of p_0 is defined as:

$$d'(x_0, x_i) = f_t\left[d(x_0, x_i)\right] = -\ln\left[d(x_0, x_i)\right]. \tag{6}$$

It is important to note that $d'(x_0, x_1)$ is not a distance metric since it does not obey the three metric space properties. However, $d'(x_0, x_1)$ is not used as a distance metric; transformation of distances is only a mathematical tool within the concept of logarithmically transformed view.

The transformed view of a base node p_0 can be used to characterize distances between p_0 and a set of other nodes in a DHT. This transformed view can be represented along a half-line as follows: the base node p_0 itself is at the end of the half-line while other DHT nodes p_i (e.g., peers of the base node, or the target node of a lookup process) are represented along this half-line at distance $d'(x_0, x_i)$ from p_0.

3.1 Long-Range Connection Density

Figure 4 represents long-range connections of a Chord [22], Pastry [21] and Kademlia [17] node as well as long-range connections of a node in a probabilistic power-law routing overlay (e.g., Symphony [16] or Accordion [12]). For Pastry, the parameter b is the bit length of numbers in the routing table, for Kademlia, the parameter k is the maximum size of buckets and for Chord, the parameter c is the parameter used in the definition of deterministic power-law routing overlays (see Definition 2). For each of these DHTs, the upper line shows long-range peers of the node in the real metric space[5] (to ease graphical representation, the ring geometry of the metric space has been straightened) while the lower line shows these peers in the logarithmically transformed view of the node. In the real metric space, the represented node is in point 0. In the transformed view, this point corresponds to $+\infty$. Finally, long-range connections in the transformed view span within the range $[0, -\ln d_S)$, where d_S is the distance from the farthest short-range peer of the node (represented by grey circle in Fig. 4).

Consider now the segment $(d_S, 1)$ covered by long-range connections in the real metric space which corresponds to the segment $(0, -\ln d_S)$ in the transformed view

[5] Since Kademlia uses a XOR metric, long-range peers of the Kademlia node are represented based on their XOR distance from the node. Note that this is different from ID-based placement along the ring.

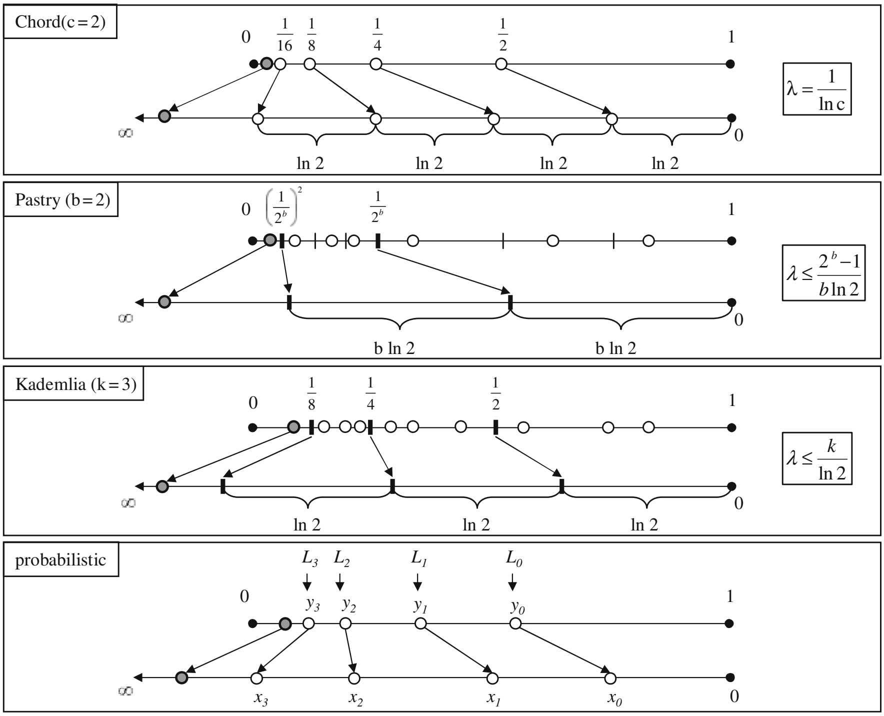


Fig. 4 Comparison of the routing table of some well known DHTs

of the node. Although long-range connection distribution differs for each of the above DHT implementations, Fig. 4 shows that it is possible to partition this segment in the transformed view into equally sized partitions of length Δx so that the number of long-range connections $N_L(\Delta x)$ be the same inside each of these partitions (either deterministically or in expected value). Based on this observation, one can define a $\lambda_{\Delta x}$ long-range connection density parameter as

$$\lambda_{\Delta x} = \frac{E\left[N_L(\Delta x)\right]}{\Delta x}. \tag{7}$$

In general, the choice of Δx is not arbitrary. In order to obtain constant long-range connection density in the entire long-range connection domain of the transformed view, Δx might need to be set to a DHT specific value (see again Fig. 4). However, for a large subclass of DHTs (including regular power-law routing overlays defined in the next subsection), long-range connection density can be defined independent of the size Δx of partitions as:

$$\lambda = \lim_{\Delta x \to 0} \frac{E\left[N_L(\Delta x)\right]}{\Delta x}. \tag{8}$$

The main advantage of the proposed λ (or $\lambda_{\Delta x}$) parameter is that it provides a simple and generic way to characterize long-range connection distribution. Furthermore, in $O(\log n)$ node state DHTs, λ characterizes the overlay independent of network size. For DHTs with constant node degree (e.g., Symphony [16]), λ depends on the size of the network ($\lambda \sim \frac{1}{\ln n}$).

Many DHT implementations have one or more tunable system parameter which affects long-range connection distribution, e.g., the bucket size k for Kademlia or the bit length b of numbers in the routing table for Pastry. The λ long-range connection density parameter allows easy comparison of overlay structure for these DHT implementations despite their mutually incompatible sets of system parameters. Figure 4 shows the λ parameter for each DHT as a function of their tunable system parameters. Note that in many cases (e.g., Chord, Pastry or Kademlia), the theoretical long-range connection values are only upper bounds of the actual long-range connection density since some routing table entries may be empty (especially for shorter distances).

3.2 Regular Power-Law Routing Overlays

Definition 4 (Regular power-law routing overlays (RPLRO)). A power-law routing overlay is called regular if the sequence of long-range connections of a node in its transformed view correspond to a randomly chosen sample of length $-\ln d_S$ from an infinite renewal process and these random samples are chosen independently for each node.

Hence, from the definition of renewal processes, distances between subsequent long-range connections of a node in its transformed view are *i.i.d* in a RPLRO. Furthermore, the distribution function $F(x)$ of these *i.i.d* random variables identifies unambiguously a regular power-law routing overlay. Random sampling from an infinite process (instead of defining long-range connections as a renewal process starting from point 0 in the transformed view) ensures uniformity of long-range connection density in the whole long-range connection range (also including the first part of this range).

Theorem 1. *Long-range connection density of a regular power-law routing overlay is uniformly $\lambda = \frac{1}{\mu}$, where μ is the mean distance between subsequent long-range connections of a node in its transformed view.*

Proof. Let $f(x)$ be the *pdf* and $F(x)$ be the *cdf* of renewal intervals (corresponding to the distances between subsequent long-range connections of a node in its transformed view). As a result of the random sampling property of RPLROs, the starting point of each partition of length $\Delta x \to 0$ can be considered as a randomly chosen point in the corresponding infinite renewal process. Hence this interval of length $\Delta x \to 0$ contains at least one renewal point (long-range connection) either when the corresponding renewal interval (selected by this randomly chosen point) is smaller than Δx or when this interval is larger than Δx but the distance between the randomly selected point and the next renewal is less than Δx. Since the *pdf* of the length of the renewal period selected by a random point is $\frac{f(x)x}{\mu}$, the probability that such an interval contains at least one renewal (long-range connection) can be written as:

$$p_1 = \int_0^{\Delta x} \frac{f(x)x}{\mu} dx + \int_{\Delta x}^{\infty} \frac{f(x)x}{\mu} \frac{\Delta x}{x} dx. \tag{9}$$

The probability that this interval contains k or more arrivals (where $k > 1$) can be upper bounded as follows:

$$p_k \leq p_1 F^{k-1}(\Delta x). \tag{10}$$

The expected number of renewals (long-range connections) within a randomly selected period of length Δx can be written as:

$$E[N_L(\Delta x)] = \sum_{i=1}^{\infty} p_i. \tag{11}$$

Hence, combining (10) and (11):

$$p_1 \quad \leq \quad E[N_L(\Delta x)] \quad \leq \quad p_1 \left[1 + \sum_{i=1}^{\infty} F^i(\Delta x) \right]. \tag{12}$$

Dividing by Δx and applying Equation(7):

$$\frac{p_1}{\Delta x} \quad \leq \quad \lambda_{\Delta x} \quad \leq \quad \frac{p_1}{\Delta x} \left[1 + \sum_{i=1}^{\infty} F^i(\Delta x) \right]. \tag{13}$$

Applying $\Delta x \to 0$ to Equation (9) and the (reasonable) assumptions[6] that $\lim_{\Delta x \to 0} F(\Delta x) = 0$ and $\lim_{\Delta x \to 0} f(\Delta x) < \infty$:

$$\lim_{\Delta x \to 0} \frac{p_1}{\Delta x} = \frac{f(0)\Delta x}{2\mu} + \frac{1 - F(\Delta x)}{\mu} = 0 + \frac{1}{\mu}. \tag{14}$$

Finally, substituting Equation(14) into the Inequality (13):

$$\frac{1}{\mu} \leq \lambda \leq \frac{1}{\mu} \quad \to \quad \lambda = \frac{1}{\mu}. \tag{15}$$

Probabilistic power-law routing overlays are regular (see Section 3.3). Pastry and Kademlia are not regular but are close to being regular with only small distortions. Finally, Chord and deterministic power-law routing overlays in general are not regular, but, they can be made regular: Considering the transformed view of a node in a

[6] These assumptions can be made because 0 distance between subsequent long-range connections does not make sense in an overlay.

DPLRO, its first long-range peer is always located at $\ln c$. Substituting the constant q in Defition 2 by a random variable so that this first long-range peer be evenly distributed in the range $[0, \ln c]$, the overlay becomes regular.

To characterize regular power-law routing overlays, we also introduce a c_v long-range connection coefficient of variance parameter describing the relative variance of distances between long-range connections in the transformed view. $c_v = \frac{\sigma}{\mu}$, where σ is the standard deviation while μ is the mean of distances between consecutive long-range connections in the transformed view of a node. In Section 4.1, we show that using the λ and c_v parameters it is possible to derive a lower bound on routing performance.

3.3 Probabilistic Power-Law Routing Overlays

It is interesting to compare the degree of randomness in the choice of long-range connection for different DHT implementations in Fig. 4. In Chord, each connection is deterministic. Pastry is somewhat more flexible, each routing table entry may contain any node of the network from a given ID range, increasing the degree of randomness. Kademlia goes one small step further in flexibility and randomness and allows the choice of any nodes (up to a maximum number of k) from a given range.

However, the choice of long-range connections can be made “even more random” within the family of routing overlays for which long-range connection density can be defined. For the “most random” routing overlays out of this family, long-range connections of a node in its transformed view correspond to a random and independent placement of points in the range $(0, -\ln d_S)$ according to a uniform distribution, which is equivalent to a truncated (spatial) Poisson process. In the following, we show that this family of “most random” routing overlays is equivalent to the family of probabilistic power-law routing overlays over a one dimensional metric space (see Definition 1).

Theorem 2. *Consider a truncated Poisson process of rate λ in the range $(0, -\ln d_S)$. Furthermore consider a routing overlay where the sequence of long-range connections in the transformed view of each node is defined as a random realization of this Poisson process. Then this routing overlay is a probabilistic power-law routing overlay of long-range connection density λ.*

Proof. Consider a small range $[x, x + \Delta x]$ in the transformed view. Inverse transforming this range back to the real metric space using $y = f_t^{-1}(x) = e^{-x}$ results into the range $[e^{-x-\Delta x}, e^{-x}] = [y - \Delta y, y]$ in the real metric space.

Using the birth process definition of Poisson processes, the probability of having an arrival (long-range connection) in the range $[x, x + \Delta x]$ of the transformed view is $\lambda \Delta x$ when $\Delta x \to 0$. Since the inverse transformation function $f_t^{-1}(x)$ is strictly monotone decreasing, the probability of having a long-range connection in the corresponding range $[y - \Delta y, y]$ of the real metric space is the same.

Using the derivative of the transformation function $f_t(y) = -\ln y$, it is possible to express the relationship between the length of these ranges when they are infinitesimally small:

$$\lim_{\Delta y \to 0} \Delta x = -f_t'(y)\Delta y = \frac{\Delta y}{y}. \tag{16}$$

Hence the probability of having a long-range connection in an infinitesimally small range of length $\Delta y \to 0$ at a distance y from this node is $\lambda \frac{\Delta y}{y}$. This is equivalent to the long-range connection distribution requirement of Definition 1 for one dimensional metric spaces. Being generated from a Poisson process, long-range connections also satisfy the independence requirement of Definition 1, hence the generated overlay is a probabilistic power-law routing overlay.

Finally, the expected number of arrivals in a Poisson process of rate λ for an interval of length Δx is $\lambda \Delta x$, hence substituting into Equation(7) any positive value of Δx interval length, the obtained long-range connection density value for this overlay equals to the λ rate of the generating Poisson process.

Theorem 3. *Consider a probabilistic power-law routing overlay in a one dimensional metric space with a long-range connection density λ. Then the sequence of long-range connections of any node in its transformed view correspond to a random realization of a truncated Poisson of rate λ in the range $(0, -\ln d_S)$.*

Proof. According to Definition 1, the probability of having a long-range connection in an small range $[y - \Delta y, y]$ of a one dimensional metric space is $c\frac{\Delta y}{y}$ when $\Delta y \to 0$ (c is a positive constant).

Transforming this range into the transformed view using $x = f_t(y) = -\ln y$ results into the range $[-\ln y, -\ln(y - \Delta y)] = [x, x + \Delta x]$. Since the transformation $f_t(y)$ is strictly monotone decreasing, the probability of having a long-range connection in the corresponding range $[x, x + \Delta x]$ of the transformed view is the same.

Using the derivative of the inverse transformation function $f_t^{-1}(x) = e^{-x}$, it is possible to express the relationship between the length of these ranges when they are infinitesimally small:

$$\lim_{\Delta x \to 0} \Delta y = -f_t^{-1'}(x)\Delta x = e^{-x}\Delta x = y\Delta x. \tag{17}$$

Hence the probability of having a long-range connection in an infinitesimally small range of length $\Delta x \to 0$ in the transformed view is $c\frac{\Delta y}{y} = c\Delta x$, independent of the value of x within the range $(0, -\ln d_S)$. Since long-range connections of a probabilistic power law routing overlay are also independent of each other according to Definition 1, the sequence of long-range connections of any node in its transformed view correspond to a random realization of a truncated Poisson of rate c in the range $(0, -\ln d_S)$.

Finally, using the definition of long-range connection density and the assumption that the long-range connection density of the given overlay is λ, it is deducible that $c = \lambda$.

Since a Poisson process is a special renewal process, from Theorem 2, it follows that probabilistic power-law routing overlays belong to the subclass of regular power-law routing overlays.

3.4 Distortions in the Transformed View

In Section 3.3, we assumed that a node can find (and create a connection to) a peer node at any given point of the metric space $(\mathscr{I},d)$. In reality, given a network of n nodes, a connection can be created only to $n-1$ points in $(\mathscr{I},d)$ corresponding to the images of all the other nodes in $(\mathscr{I},d)$. Hence in real systems, a connection is established to the peer node whose images is the closest to the "theoretical" point in $(\mathscr{I},d)$ drawn according to the required distribution. This introduces small distortions to theoretical distance distribution.

Similar distortions exist for deterministic power-law routing overlays. E.g., in Chord, long-range connections (fingers) point to the first node whose distance is not smaller than $\frac{1}{2},\frac{1}{4},\frac{1}{8},\ldots$. This results into a sequence of distances $\frac{1}{2}+\varepsilon_1,\frac{1}{4}+\varepsilon_2,\frac{1}{8}+\varepsilon_3,\ldots$. In a network of n nodes uniformly distributed in the range $[0,1)$ of the metric space $(\mathscr{I},d)$, ε will be a random variable with exponential distribution of parameter n.

While the distribution of this small offset is the same for all distances in the real metric space, it depends strongly on the distance in the transformed view of a node. For large real distances, this offset is negligible, however, it increases exponentially and can be considerable for small real distances in the transformed view.

4 Stochastic Analysis of Routing

The role of short-range and long-range connections in the routing process is complementary. While short-range contacts ensure the success of greedy forwarding, long-range contacts expedite routing and provide $O(\log n)$ bounds on the number of routing hops. For deterministic routing geometries, the routing process can be clearly separated into a first phase using only long-range contacts and a second phase using only short-range contacts. For non-deterministic routing geometries, the first routing hops usually take place via long-range connections while the last hops usually take place via short-range connections and the probability of routing via a short-range peer increases monotonously approaching to the target. Nevertheless, for non-deterministic routing geometries, it is not possible to separate routing process into distinct long-range and short-range routing phases.

Analytical study of this dual routing process is rather complicated. Analysis becomes much easier if forwarding is restricted to either only short-range or only long-range connections.

Restricting forwarding to short-range connections, progress toward the target becomes linear. Assuming that node identifiers are drawn independently and at random according to a uniform distribution from the interval $[0,1)$ of the metric space $(\mathscr{I},d)$ (see Section 2.2), each routing hop has the same length in expected value independent of the current distance from the target.[7] The length of consecutive routing hops can be described by a series of independent Erlang distributed random variable with rate n (number of nodes in the network) and shape parameter N_S (number of short-range connections per node). Obviously, routing via only short-range contacts degrades routing performance from $O(\log n)$ to $O(n)$.

In Section 4.1, we show that using logarithmic transformation, analysis is also possible when restricting forwarding to long-range connections; progress toward the target will be linear in the transformed view of the target. However, simply forbidding forwarding via short-range contacts may cause routing failures. Therefore, to analyze long-range only forwarding, we use an imaginary routing overlay where the sequence of long-range connections in the transformed view of a node is infinite instead of being truncated after reaching the short-range connection domain. For regular power-law routing overlays, this means that long-range connections correspond to infinite random samples from a renewal process instead of random samples of length $-\ln d_S$.

In the real routing overlay, forwarding takes place via a short-range peer only when the target is closer than the closest long-range peer of the forwarding node. When the real routing overlay is forwarding via a short-range peer, the modified long-range only model is forwarding through an imaginary long-range peer at a smaller distance

Figure 5 demonstrates the difference between real forwarding (via a short-range connection) and long-range only forwarding via an imaginary long-range connection. The upper line in the figure represent the real metric space while the lower line shows the transformed view of the forwarding node. For a real overlay, forwarding occurs via a short-range peer. However, when restricting forwarding to long-range connections in order to simplify analysis, forwarding takes place via the imaginary long-range peer being the closest to the target node.

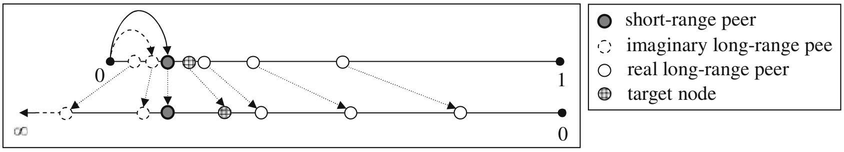


Fig. 5 Forwarding through imaginary long-range connections

Forwarding via an imaginary long-range connection always results in less progress than the real forwarding would result in via a short-range connection. Therefore

[7] If $N_S > 1$, the expected value of the last hop reaching the target is smaller.

results on routing progress obtained from analysis restricted to long-range forwarding can be used as a lower bound on real routing progress.

Modeling long-range only forwarding, a routing process is terminated when the distance of the current forwarding node from the target decreases below d_S (the distance between the target node and the node whose farthest short-range peer is the target node[8]). Let M_l be the number of routing hops for long-range only routing until the termination and let M be the number of routing hops for the real routing process. Since routing progress of long-range only forwarding is always equal to or less than routing progress of real forwarding, the real routing process will always reach a peer with direct short-range connection to the target node in M_l or less hops. Hence $M_l + 1$ can be used as an upper bound on the real number of routing hops:

$$M \leq M_l + 1. \tag{18}$$

The rest of this section is structured as follows: Section 4.1 analyzes progress of the routing process in the transformed view of the target using this long-range only forwarding model. Then Section 4.2 uses the obtained long-range only results to derive upper bounds on the number of routing hops for the real routing process (using both short-range and long-range connections).

4.1 Analysis of Routing in the Transformed view

Analysis of routing in the transformed view can be best introduced through an example. Figure 6a shows one hop of an example routing process: a request reaches forwarding node F_k in step k and node F_k forwards this request to its long-range peer F_{k+1} being the closest to the target node T without overshooting it. Figure 6b shows distances in the real metric space (upper line) and the transformed view (lower line) of node F_k while Fig. 6c shows the same distances as seen in the real and transformed view of the target. Note that the default direction of the ring is reversed in Fig. 6c in order to represent remaining distances from the perspective of the target node.

d_k and d_{k+1} is the distance from the target in step k and $k+1$ respectively, while d'_k and d'_{k+1} are the same distances in the transformed view of the target. To analyze per hop routing progress in the transformed view of the target node, let's express the progress $u_k = d'_{k+1} - d'_k$ toward the target after step k as a function of the distance v_k from the next-hop node in the transformed view of forwarding node F_k. Applying transformation to distances in Fig. 6c:

[8] Assuming uniform distribution of node identifers in the DHT metric space, d_S will be a random variable with Erlang distribution of rate n and shape parameter N_S (where n is the number of nodes in the DHT and N_S is the number of short-range connections per node.)

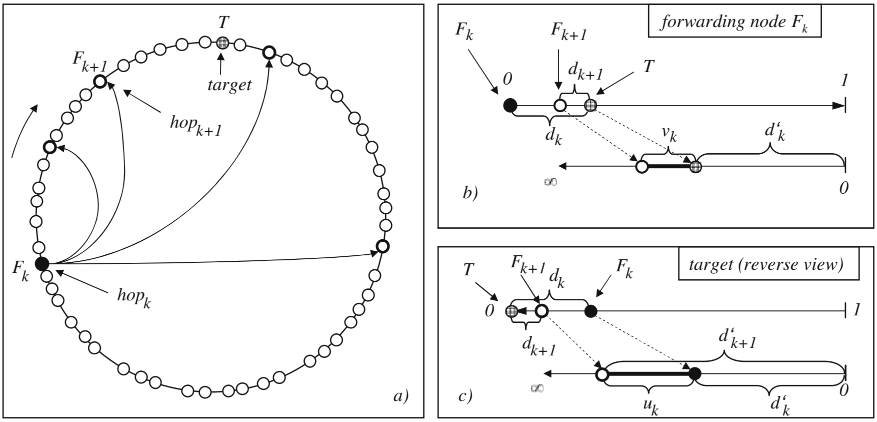


Fig. 6 Routing from hop k to hop $k+1$

$$u_k = -\ln d_{k+1} - d'_k. \tag{19}$$

Inverse transforming distances in Fig. 6/b:

$$d_{k+1} = e^{-d'_k} - e^{-d'_k - v_k} = e^{-d'_k}(1 - e^{-v_k}). \tag{20}$$

Finally, substituting Equation(20) into (19):

$$u_k = -\ln\left[e^{-d'_k}\left(1 - e^{-v_k}\right)\right] - d'_k = -\ln\left(1 - e^{-v_k}\right). \tag{21}$$

Hence the progress u_k after routing hop k in the transformed view of the target can be expressed as a function of the distance v_k from the next-hop node in the transformed view of forwarding node F_k:

$$u = h(v) = -\ln\left(1 - e^{-v}\right). \tag{22}$$

Note: as Section 2.2 mentions, routing in DHTs with non-Euclidean metric spaces cannot be analyzed using this model. The reason is that Equation(20) uses the assumption that $d(x,z) = d(x,y) + d(y,z)$ which holds only for the one dimensional Euclidean metric space. For any other metric spaces: $d(x,z) \leq d(x,y) + d(y,z)$ (triangle inequality).

In the followings, we analyze routing in regular power-law routing overlays in general. Using the above transformation results, we derive a lower bound on routing performance as a function of λ and c_v. Then we analyze two special cases of regular power-law routing overlays in more details: the probabilistic and "regularized" deterministic overlays. We derive exact analytical results on their routing performance and compare these results to the generic lower bound.

4.1.1 Regular Power-Law Routing Overlays

Theorem 4. *Considering the routing process via long-range connections in a regular power law routing overlay, the length of the per-hop progress u_k in the transformed view of the target is* i.i.d *for subsequent hops. Furthermore, the expected value of this per-hop routing progress is lower bounded by*

$$E[u_k] \geq -\ln\left[1 - e^{-\frac{1+c_v^2}{2\lambda}}\right]. \tag{23}$$

where λ is the long-range connection density and c_v is the long-range connections density coefficient of variation in the overlay.

Proof. In regular power-law routing overlays, the sequence of long-range connections of different nodes in their transformed view correspond to independently selected random samples from an infinite renewal process. Therefore, the target node can be considered as a uniformly distributed random point in the transformed view of a forwarding node (see Fig. 6b). As a consequence, the random variable v_k corresponds to the distance of a random point from the next renewal (long-range connection) in the renewal process of long-range connection (residual life). Hence, the series of the random variables v_k will be *i.i.d*, and applying Equation (22), the series of the random variables u_k will be also *i.i.d*.

Let $\mu = E[x]$ be the expected value and $\mu_2 = E[x^2]$ be the second moment of the length of renewal periods (corresponding to distances between subsequent long-range connection in the transformed view of a node). Then, from renewal theory, the mean residual life in this renewal process (corresponding to $E[v]$) can be expressed as:

$$E[v] = \frac{E[x^2]}{2E[x]} = \frac{Var(x) + E^2[x]}{2E[x]} = \frac{E[x]}{2}(1 + c_v^2). \tag{24}$$

Using Theorem 1, $E[x] = \frac{1}{\lambda}$ for any RPLRO, hence:

$$E[v] = \frac{1 + c_v^2}{2\lambda}. \tag{25}$$

Using Equation 22, the distribution of the per-hop routing progress in the transformed view of the target (u) can be expressed as a convex function $h(v)$ of the random variable v. Therefore the Jensen inequality can be applied as follows:

$$E[u] = E[h(v)] \geq h(E[v]) = -\ln\left[1 - e^{-\frac{1+c_v^2}{2\lambda}}\right]. \tag{26}$$

4.1.2 Probabilistic Power-Law Routing Overlays

According to Theorem 3 on PPLROs, the sequence of long-range connections in the transformed view of a node can be described as a realization of a stationary

Poisson process of rate λ, where λ is the long-range connection density of this overlay. Hence, in the transformed view of the forwarding node F_k, the target of the lookup process corresponds to an arbitrary point while the image of the next-hop node F_{k+1} corresponds to the next arrival in this Poisson process. The random variable v_k describes the distance between these two points in the transformed view of F_k (see Fig. 6b).

As a consequence of the memoryless property of Poisson processes, picking an arbitrary point in the process, the distance to the next arrival will always be exponentially distributed with parameter λ. Hence the distribution of v_k is the same for each step of the routing process (via long-range connections) and the *pdf* of v v_k is:

$$g_{prob}(v) = \lambda e^{-\lambda v}. \tag{27}$$

Another consequence of the memoryless property of Poisson processes is that the random variables v_k and v_{k+1} are independent, hence the series v_k are *i.i.d* random variables. Since u_k (the progress toward the target in the k^{th} routing hop) can be derived from v_k using Equation (22), u_k is also a series of *i.i.d* random variables (since PPLROs are regular, this could be derived also applying Theorem 4). The *pdf* of u can be obtain by transforming the *pdf* of v using the function $h(v)$:

$$f_{prob}(u) = g_{prob}\left(h^{-1}(u)\right)\left|\frac{dh^{-1}(u)}{du}\right| = \lambda(1-e^{-u})^{(\lambda-1)}e^{-u}. \tag{28}$$

Hence:

$$F_{prob}(u) = \int_0^u f_{prob}(t)dt = (1-e^{-u})^{\lambda}. \tag{29}$$

Finally, the expected value of the length of one routing hop in the transformed view of the target[9]:

$$E_{prob}[u] = \int_0^\infty f_{prob}(u)u\,du = H_\lambda, \tag{30}$$

where H_x is the harmonic number [24] (generalized for real numbers) of x. For practical λ values, the following approximation can be used[10]:

$$H_\lambda \approx \ln\left[(e-1)\lambda+1\right]. \tag{31}$$

The above results can be transformed back from the transformed view of the target node to the real metric space of the DHT as follows.

Theorem 5. *Consider the routing process in a probabilistic power-law routing overlay of long-range connectiondensity λ. Then the series of random variables*

[9] Calculated using the Mathematica software from Wolfram Research Inc. (http://www.wolfram.com)

[10] This approximation provides less than ±1% relative error if $\lambda > 0.5$ and less than $+5\%$ relative error if $0 < \lambda < 0.5$. Note that λ is typically larger than $\frac{1}{\ln 2} \approx 1.41$ for most DHTs (see Section 3.1)

$w_k = \frac{d_{k+1}}{d_k}$ *describing the ratio of distances from the target after and before a routing hop via a long-range connection are* i.i.d *and the* pdf *and expected value of* w_k *are:*

$$f^w_{prob}(w) = \lambda(1-w)^{(\lambda-1)} \quad \text{if} \quad 0 < w < 1 \quad \text{and0otherwise} \tag{32}$$

and

$$E[w] = \frac{1}{1+\lambda}. \tag{33}$$

Proof. Since $u_k = d'_{k+1} - d'_k$ in the transformed view of the target and since transformed distances can be obtained as $d'_k = -\ln d_k$ and $d'_{k+1} = -\ln d_{k+1}$ from distances in the real metric space, u_k can be expressed as:

$$u_k = -\ln d_{k+1} - (-\ln d_k) = -\ln \frac{d_{k+1}}{d_k}. \tag{34}$$

Hence defining, the random variable $w_k = \frac{d_{k+1}}{d_k}$ as the ratio of distances after and before a routing hop via a long-range connection, this random variable w_k can be expressed as a function $w_k = \Phi(u_k) = e^{-u_k}$ of the random variable u_k. According to Theorem 4, u_k is a series of *i.i.d* random variables, therefore w_k will be also a series of *i.i.d* random variables (to simplify notation, u_k and w_k are denoted simply by u and v hereafter). $\Phi(u) = e^{-u}$ is a strictly monotone decreasing function. Hence the *cdf* of w can be expressed from the *cdf* of u as:

$$F^w_{prob}(w) = 1 - F_{prob}\left(\Phi^{-1}(w)\right) = 1 - \left[1 - e^{-(-\ln w)}\right]^{\lambda} = 1 - (1-w)^{\lambda}. \tag{35}$$

The *pdf* of w can be obtained as the derivative of $F^w_{prob}(w)$:

$$f^w_{prob}(w) = \frac{dF^w_{prob}}{dw} = \lambda(1-w)^{(\lambda-1)}. \tag{36}$$

As a result of greedy routing, $d_{k+1} < d_k$ holds for each routing step, hence $0 < w_k < 1$ and the expected value of the random variable w is:

$$E[w] = \int_0^1 f^w_{prob}(w)w dw = \left[\frac{(1-w)^{\lambda}(1+\lambda w)}{1+\lambda}\right]_0^1 = \frac{1}{1+\lambda}. \tag{37}$$

Hence the distance to the target decreases in expected value by a factor of $1+\lambda$ after each routing hop via a long-range connection. Since these distance decrease ratios in subsequent routing hops are independent, the expected value of distance decrease after i routing hops via long-range connections can be expressed as:

$$E\left[\frac{d_{k+i}}{d_k}\right] = \left(\frac{1}{1+\lambda}\right)^i. \tag{38}$$

4.1.3 Deterministic Power-Law Routing Overlays

Definition 2 introduces deterministic power-law overlays which can be considered as a generalization of the Chord overlay. These overlays are not regular because the sequence of long-range connections in the transformed view of nodes correspond to the same renewal process for each node and lacks the random sampling property of regular power law routing overlays. However, DPLROs can be made regular substituting the constant q in Definition 2 with a random variable so that the first long-range peer is evenly distributed over the range $[0, \ln c]$ in the transformed view of the node. In this subsection, we analyze these "regularized" deterministic power-law routing overlays.

To ease comparison with PPLROs and regular power-law routing overlays in general, the generic λ long-range connection density is used during the analysis instead of the parameter c in the definition of deterministic power law routing overlays. The relationship between these two parameters can easily be determined from Fig. 4:

$$\lambda = \frac{1}{\ln c} \quad \Leftrightarrow \quad c = e^{\frac{1}{\lambda}}. \tag{39}$$

As for any regular power-law routing overlay, target nodes in the transformed view of forwarding nodes can be considered as uniformly distributed random points. Hence the *pdf* of the random variable v_k for DPLRO:

$$g_{det}(v) = \begin{cases} \lambda & \text{if} \quad 0 < v < \dfrac{1}{\lambda} \\ 0 & \text{otherwise} \end{cases} \tag{40}$$

Transforming this distribution using Equation(22), the *pdf* of the random variable u_k:

$$f_{det}(u) = g_{det}\left(h^{-1}(u)\right) \left| \frac{dh^{-1}(u)}{du} \right| = \begin{cases} \lambda \dfrac{e^{-u}}{1-e^{-u}} & \text{if} \quad u > -\ln\left[1 - e^{\frac{1}{\lambda}}\right] \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

Hence the expected value of the per-hop routing progress in the transformed view of the target node can be obtained as[11]:

$$E_{det}[u] = \int_{-\ln\left[1-e^{-\frac{1}{\lambda}}\right]}^{\infty} \lambda \frac{e^{-u}}{1-e^{-u}} u du. \tag{42}$$

4.1.4 Comparison of Per-Hop Routing Progress in the Transformed View

In the previous subsections, we analyzed routing via long-range connections in regular power-law routing overlays. In Theorem 4, we show that the length of the

[11] The analytical form of this integral is too complicated, Fig. 7 represents the results numerically

per-hop progress in the transformed view of the target (distance between the images of subsequent forwarding nodes in this transformed view) is *i.i.d.* In other words, lookup approaches toward the target in its transformed view at constant "speed" in expected value for any regular power-law routing overlay.

Furthermore, knowing the λ long-range connection density and the c_v long-range connection density coefficient of variation parameters of the overlay, it is possible to derive a lower bound on the expected value of the length of this per-hop progress in the transformed view of the target (see Equation (23)).

We have also derived analytically the expected value of this per hops progress for two special cases, namely the probabilistic and the "regularized" deterministic power-law routing overlays. Figure 7 compares these expected values as well as their generic lower bounds (derived using Inequality 23) as a function of the long-range connection density. The result shows that deterministic overlays provide better per-hop progress than probabilistic overlays for all values of λ. This is a consequence of different coefficients of variation (c_v) for distances between subsequent long-range connections. According to Equation (23), the lower bound on $E[u]$ decreases monotonically with increasing c_v values. For DPLROs, where the distance between subsequent long-range connections is constant in the transformed view of a node:

$$c_v^{det} = 0, \tag{43}$$

while for PPLROs, where these distances are exponentially distributed:

$$c_v^{prob} = \frac{\sigma}{\mu} = \frac{\frac{1}{\lambda}}{\frac{1}{\lambda}} = 1. \tag{44}$$

Although Equation (23) can be used to express lower bounds on the expected value for any regular power-law routing overlay, the distribution of per-hop routing progress can differ significantly for different overlays. Figure 8 compares the *pdf* $f(u)$ for different long-range connection density values both for probabilistic and deterministic power-law routing overlays. As it can be expected, the deterministic routing overlay guarantees a minimum progress for each routing hop. In probabilistic routing overlays, there is no such lower bound on the length of one single hop.

Equation (29) reveals another interesting property of the distribution of u for probabilistic power-law routing overlays: $F_{prob}(u)$ can be obtained by raising the *cdf* of an exponential distribution to the power λ. As a result, $\lambda = 1$ is a very special long-range connections density value where the length of one routing hop in the transformed view of the target node is exponentially distributed with parameter 1. This means that the sequence of routing hops in the transformed view of the target node corresponds to the same stochastic process as the sequence of long-range connections in the transformed view of any nodes; both can be described by a Poisson process of rate 1. For any other λ values, the length of routing hops is not exponentially distributed. Figure (8) shows well the difference in the shape of

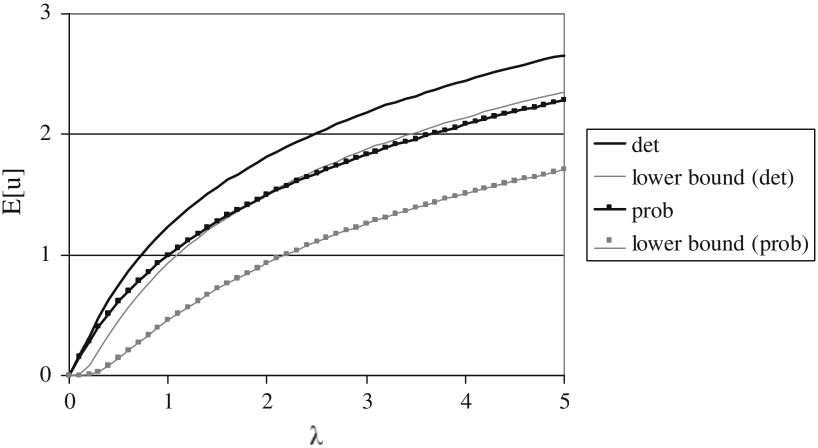


Fig. 7 Expected value of *u* and its estimated lower bound as a function of λ

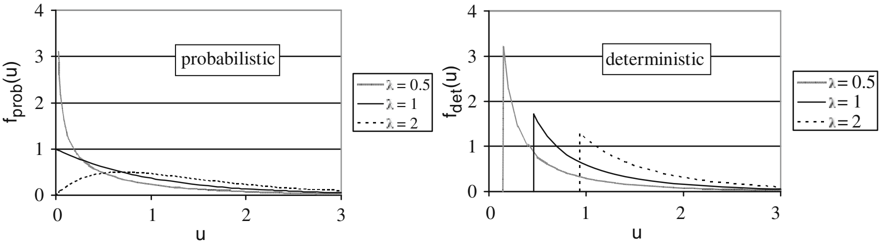


Fig. 8 Pdf of *u* for different λ values

the probability density function for long-range connection density values $\lambda = 1$, $\lambda < 1$ and $\lambda > 1$.

The sequence of long-range connections in the transformed view of a node can also be approximated by a renewal process for many "non-regular" DHTs (see Section 3.1). However, irregularities in the distribution of distances between successive long-range connections induces distortions to the "constant speed" progress in the transformed view of the target and make mathematical analysis difficult.

For example, in Pastry, long-range connection density have a slight periodic variation. The length of this period is $b \ln 2$ in the transformed view (where b is the bit length of numbers in the routing table). This slight periodic fluctuation is visible on the graph showing the expected number of routing hops as a function of network size (see Fig. 4 in paper [21]).

4.2 Upper Bound on the Expected Number of Routing Hops

In the previous subsections, we have analyzed routing via long-range connections in the transformed view of target nodes. We have shown that the per-hop routing progress u_k is *i.i.d* for regular power-law routing overlays, hence the images of forwarding nodes in the transformed view of the target can be described as a renewal process. Furthermore, we proposed a lower bound on the expected value of this per-hop progress as a function of the λ and c_v overlay parameters (see Theorem 4).

Although these results cannot be used directly to characterize overlay performance, the proposed renewal process model allows analytical derivation of an important overlay performance metric: an upper bound on the expected number of routing hops as a function of network size and the overlay parameters (λ, c_v and N_S). The computation of this overlay performance metric is based on the upper bound of Lorden for renewal processes (see [4], page 110):

$$U_L(t) \leq \frac{t}{\mu} + \frac{\mu_2}{\mu^2} - 1, \tag{45}$$

where $U_L(t)$ is the renewal function (the expected value of renewals until time t), μ is the mean of renewal periods and μ_2 is the second moment of renewal periods. Applying this bound to the renewal process corresponding to the sequence of forwarding nodes in the transformed view of the target node, the variables in Inequality(45) correspond to the followings:

- $U_L(t)$ corresponds to the expected value of the number of routing hops for long-range only routing;
- t corresponds to the length of the long-range routing path in the transformed view of the target node (from the image of the initiator node to the image of the first node having a direct short-range connection to the target node);
- μ corresponds to $E[u]$, for which, Theorem 4 gives a lower bound as a function of the overlay parameters λ and c_v;
- μ_2 corresponds to $E[u^2] = \int_0^\infty u^2 f(u)du$.

Lemma 1. *Considering the routing process via long-range connections in a regular power law routing overlay, the second moment of the length of the per-hop progress u in the transformed view of the target node is upper bounded by*

$$E[u^2] \leq 2.41\lambda, \tag{46}$$

where λ is the long-range connection density of the overlay

Proof. The random variable u can be obtained from the random variable v using Equation (22) while the variable v itself corresponds to the residual life in the renewal process corresponding to the sequence of long-range connections in the transformed view of a node (see proof of Theorem 4). From renewal theory, the *pdf* of the residual life v can be expressed as follows:

$$g(v) = \frac{1 - F(v)}{E(x)} = \lambda(1 - F(v)), \tag{47}$$

where $F(x)$ is the *cdf* of renewal intervals (corresponding to the distances between subsequent long-range connections of a node in its transformed view), $E[x]$ is the expected length of these intervals and we have used $\frac{1}{E[x]} = \lambda$ from Theorem 1.

Since a *cdf* is always a non-decreasing function and $0 \leq F(x) \leq 1$, the *pdf* $g(v)$ is a non-increasing function upper bounded by $g(v) \leq g(0) = \lambda$.

This upper bound can be used to derive an upper bound on $f(u)$. Using Equation(22) to transform v to u:

$$f(u) = g(-\ln(1 - e^{-u}))\frac{e^{-u}}{1 - e^{-u}} \quad \leq \quad \lambda \frac{e^{-u}}{1 - e^{-u}}. \tag{48}$$

Hence

$$E[u^2] = \int_0^\infty u^2 f(u)du \quad \leq \quad \int_0^\infty \lambda \frac{e^{-u}}{1 - e^{-u}} u^2 du \quad \leq \quad 2.41\lambda. \tag{49}$$

Lemma 2. *Assuming uniform distribution of node identifiers in the metric space of the DHT and uniform selection of target nodes for the routing process, the expected length of the routing path in the transformed view of the target is:*

$$E[t] = \ln n + \gamma - 1 - H_{N_S - 1} + \varepsilon, \tag{50}$$

where N_S is the number of short-range connections per node, n is the number of nodes in the overlay, H_k is the k^{th} harmonic number, γ is the Euler-Mascheroni constant [12] *and ε is a small positive error term*

$$\varepsilon \in o\left(\frac{n^{N_S}}{e^n}\right) \tag{51}$$

negligible except for very small network sizes.

Proof. When using the long-range only forwarding model, a routing process (as seen in the transformed view of the target) starts from the image of the initiator node and ends at the image of the first node having a direct short-range connection to this target node. The expected length $E[t]$ of this routing path can be obtained as the difference between the expected location of these start and end points.

Assuming that nodes of the overlay are uniformly distributed in the range $[0,1)$ of the DHT metric space and target nodes are selected also uniformly by initiator nodes, the distance between initiator and target nodes will be uniformly distributed over the interval $(0,1)$. Applying logarithmic transformation to this distribution according to Equation (6) results into exponentially distributed distances in the transformed view of the target with the *pdf*: $f'(x) = e^{-x}$. Hence the expected value of the distance between the target and initiator nodes in the transformed view of the target

[12] The Euler-Mascheroni constant is defined as $\gamma = \lim_{k\to\infty}(H_k - \ln k) \approx 0.5772$.

will be

$$E[L_{start}] = \int_0^\infty e^{-x} x dx = 1. \tag{52}$$

Assuming again that nodes of the overlay are uniformly distributed in the metric space of the DHT, the distance between a node and its farthest short-range peer will be Erlang distributed with rate n and shape parameter N_S, where n is the number of nodes in the overlay and N_S is the number of short-range connections per node. Hence the *pdf* of this distance distribution will be:

$$f_{Erl}(y) = \frac{n^{N_S} y^{N_S - 1} e^{-ny}}{(N_S - 1)!}. \tag{53}$$

Transforming the *pdf* of this Erlang distribution according to Equation (6), the *pdf* in the transformed view will be:

$$f'_{Erl}(x) = f_{Erl}\left(f_t^{-1}(x)\right) \frac{df_{Erl}^{-1}}{dx} = \frac{e^{-\left[ne^{-x} + x(N_S - 1)\right]} n^{N_S}}{(N_S - 1)!}. \tag{54}$$

Hence the expected value of the distance between a node and its farthest short-range peer in its transformed view[13]:

$$E[L_{end}] = \int_0^\infty f'_{Erl}(x) x dx = \ln n + \gamma - H_{N_S - 1} + \varepsilon, \tag{55}$$

where H_k is the kth harmonic number, γ is the Euler-Mascheroni constant and ε is a small positive error term upper bounded by

$$\varepsilon < e^{-n}\left[1 + \frac{(n + N_S)^{N_S - 2}}{(N_S - 1)!}\right]. \tag{56}$$

Hence ε can be expressed using the following asymptotic bound:

$$\varepsilon \in o\left(\frac{n^{N_S}}{e^n}\right). \tag{57}$$

Typically, N_S is a small number, hence – except for very small network sizes – ε is negligibly small.

Theorem 6. *The expected number of routing hops U in a regular power-law routing overlay is upper bounded by:*

$$U(n, \lambda, c_v, N_S) \quad \leq \quad \frac{\ln n - H_{N_S - 1} - 0.42}{-\ln\left[1 - e^{-\frac{1 + c_v^2}{2\lambda}}\right]} + \frac{2.41\lambda}{\ln^2\left[1 - e^{-\frac{1 + c_v^2}{2\lambda}}\right]} + \varepsilon, \tag{58}$$

[13] Calculated using the Mathematica software from Wolfram Research Inc. (http://www.wolfram.com)

where n is the number of nodes in the overlay, λ is the long-range connection density, c_v is the long-range connection density coefficient of variation, N_S is the number of short-range connections per node and ε is a small positive error term

$$\varepsilon \in o\left(\frac{n^{N_S}}{e^n}\right) \tag{59}$$

negligible except for very small network sizes.

Proof. Substituting the results of Lemma 1, Lemma 2 and Theorem 4 into the Lorden bound (Inequality 45), an upper bound can be obtained on the expected number of routing hops for long-range only forwarding:

$$U_L(t) \quad \leq \quad \frac{\ln n - H_{N_S-1} - 0.42 + \varepsilon}{-\ln\left[1 - e^{-\frac{1+c_V^2}{2\lambda}}\right]} + \frac{2.41\lambda}{\ln^2\left[1 - e^{-\frac{1+c_V^2}{2\lambda}}\right]} - 1. \tag{60}$$

Then, the upper bound on the number of routing hops for real routing (via both short-range and long-range connections) can be obtained using $U(t) < U_L(t) + 1$ from Inequality (18).

When the first and second moments of the per-hop progress u in the transformed view of the target are known, the upper bound of Theorem 6 can be further tightened:

Theorem 7. *The expected number of routing hops U in a probabilistic power-law routing overlay is upper bounded by:*

$$U(n,\lambda,N_S) \quad \leq \quad \frac{\ln n - H_{N_S-1} - 0.42}{H_\lambda} + \frac{1.645 - \psi'(1+\lambda)}{H_\lambda^2} + 1 + \varepsilon, \tag{61}$$

where n is the number of nodes in the overlay, λ is the long-range connection density, and N_S is the number of short-range connections per node, $\psi'(x)$ is the first derivative of the digamma function and ε is a small positive error term

$$\varepsilon \in o\left(\frac{n^{N_S}}{e^n}\right) \tag{62}$$

negligible except for very small network sizes.

Proof. Using Equation (30), the first moment of u is $\mu = H_\lambda$, where H_x is the harmonic number generalized for real numbers. The second moment of u can be derived from the *pdf* of u given by Equation (28):

$$\mu_2 = \int_0^\infty f_{prob}(u)u^2 du = \frac{\pi^2}{6} + H_\lambda^2 - \psi'(1+\lambda). \tag{63}$$

Substituting μ, μ_2 and the result of Lemma 2 into the Lorden bound (Inequality 45) an upper bound can be obtained on the expected number of routing hops for long-range only forwarding:

$$U_L(t) \quad \leq \quad \frac{\ln n - H_{N_S-1} - 0.42 + \varepsilon}{H_\lambda} + \frac{\frac{\pi^2}{6} + H_\lambda^2 - \psi'(1+\lambda)}{H_\lambda^2} - 1. \qquad (64)$$

Performing simplifications, the upper bound on the number of routing hops for real routing (using both short-range and long-range connection) can be obtained using $U(t) < U_L(t) + 1$ from Inequality (18).

5 Summary

Although most DHT overlays are structurally similar to the "small-world" navigation model of Kleinberg [8] – architectural and algorithmic details of different DHT variants differ significantly. Furthermore, lookup performance depends on a sets of different and often incompatible parameters which makes analytical comparison rather difficult. The objective of this chapter was to propose a general analytical model that can be used to investigate and compare static routing performance of most DHT implementations as a function of their overlay structure.

To capture the above mentioned common foundations of overlay structure, we have introduced the concept of logarithmically transformed view, where distances between a reference node and other nodes are represented after a logarithmic transformation. We have shown that long-range peers of a node form a linear sequence in this transformed view for most DHTs. Furthermore, we have identified an important subclass of DHT overlays – regular power-law routing overlays – where this sequence can be described as a random sample from an infinite renewal process. Based on this stochastic model, we have introduced the λ long-range connection density and c_v long-range connection density coefficient of variation parameters. For $O(\log n)$ node state, these parameters characterize long-range connection distribution independent of network size.

Using the renewal process model of long-connections, we have analyzed stochastically the progress of lookup process via long-range connections. We have shown that the sequence of intermediate forwarding nodes in the transformed view of the target node can be also described as a renewal process. Additionally, we have derived (i) the distribution of this per-hop routing progress for the spacial cases of probabilistic and "regularized" deterministic power-law routing overlays (ii) a generic upper bound on the per-hop routing progress in the transformed view of the target as a function of the λ and c_v long-range connection distribution parameters.

Finally, using the renewal process model of the routing process, we have derived closed form upper bounds on the expected number of routing hops as a function of network size and the overlay parameters λ, c_v and N_S.

The above model and results can be applied directly to any DHT using probabilistic power-law routing overlays (e.g., Symphony [16], Accordion [12], etc.). Additionally, overlay structure and static routing performance of any DHT using a one-dimensional metric space and being structurally similar to the "small-world" navigation model of Kleinberg can be approximated applying this model (e.g., Chord [22] and its variants, Pastry [21], Bamboo [20], Kademlia [17], etc.)

References

1. Aberer, K., Alima, L.O., Ghodsi, A., Girdzijauskas, S., Haridi, S., Hauswirth, M.: The essence of P2P: A reference architecture for overlay networks. In: Proceedings of the 5th IEEE International Conference on Peer-to-Peer Computing, pp. 11–20. Konstanz, Germany (2005)
2. Aspnes, J., Diamadi, Z., Shah., G.: Fault tolerant routing in peer-to-peer systems. In: Proceedings of the 21st Annual Symposium on Principles of Distributed Computing (PODC '02), pp. 223–232. Monterey, CA, USA (2002)
3. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
4. Beichelt, F., Fatti, L.P.: Stochastic Processes and Their Applications. CRC Press (2001)
5. Gummadi, K., Gribble, S., Ratnasamy, S., Shenker, S., Stoica, I.: The impact of DHT routing geometry on resilience and proximity. In: Proceedings of ACM Sigcomm, pp. 381–394. Karlsruhe, Germany (2003)
6. Kaashoek, F., Karger, D.: Koorde: A simple degree-optimal distributed hash table. In: Proceedings of the 2nd International Workshop on Peer-to-Peer Systems, LNCS 2735, pp. 98–107. Berkeley CA, USA (2003)
7. Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., Lewin, D.: Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In: Proceedings of the 29th annual ACM symposium on theory of computing (STOC '97), pp. 654–663. El Paso, TX, USA (1997)
8. Kleinberg, J.M.: The small-world phenomenon: an algorithmic perspective. In: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, pp. 163–170. Portland, OR, USA (2000)
9. Kleinrock, L.: Queueing Systems, vol. 1. Wiley (1975)
10. Kong, J.S., Bridgewater, J.S.A., Roychowdhury, V.P.: A general framework for scalability and performance analysis of DHT routing systems. In: Proceedings of the International Conference on Dependable Systems and Networks (DSN'06), pp. 343–354. Philadelphia, PA, USA (2006)
11. Lawler, G.F.: Introduction to Stochastic Processes. CRC Press (2006)
12. Li, J., Stribling, J., Morris, R., Kaashoek, M.F.: Bandwidth-efficient management of DHT routing tables. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, pp. 99–114. Boston, MA, USA (2005)
13. Li, J., Stribling, J., Morris, R., Kaashoek, M.F., Gil, T.M.: A performance vs. cost framework for evaluating DHT design tradeoffs under churn. In: Proceedings of INFOCOM '05, pp. 225–236. IEEE, Cambridge, MA, USA (2005)
14. Locher, T., Schmid, S., Watternhofer, R.: eQuus: A provably robust and locality-aware peer-to-peer system. In: Proceedings of the 6th IEEE International Conference on Peer-to-Peer Computing, pp. 3–11. Cambridge, UK (2006)
15. Loguinov, D., Kumar, A., Rai, V., Ganesh, S.: Graph-theoretic analysis of structured peer-to-peer systems: Routing distances and fault resilience. In: Proceedings of ACM SIGCOMM'03, pp. 395–406. Karlsruhe, Germany (2003)
16. Manku, G.S., Bawa, M., Raghavan, P.: Symphony: Distributed hashing in a small world. In: Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems, pp. 127–140. Seattle, WA, USA (2003)
17. Maymounkov, P., Mazieres, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: Proceedings of the 1st International Workshop on Peer-to-Peer Systems, LNCS 2429, pp. 53–65. Cambridge, MA, USA (2002)

18. Rao, K.R., Yip, P.: Discrete cosine transform: algorithms, advantages, applications. Academic Press, San Diego, CA, USA (1990)
19. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Proceedings of ACM SIGCOMM, pp. 161–172. San Diego, CA, USA (2001)
20. Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J.: Handling churn in a DHT. Tech. Rep. UCB/CSD-3-1299, UC Berkeley, Computer Science Division, UC Berkeley, USA (2003)
21. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms, LNCS 2218, pp. 329–350. Springer, Heidelberg, Germany (2001)
22. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: Scalable peer-to-peer lookup service for internet applications. In: Proceedings of ACM SIGCOMM, pp. 149–160. San Diego, CA, USA (2001)
23. Wu, D., Tian, Y., Ng, K.W.: Analytical study on improving DHT lookup performance under churn. In: Proceedings of the 6th IEEE International Conference on Peer-to-Peer Computing, pp. 249–258. IEEE, Cambridge, UK (2006)
24. Harmonic number definition. `http://mathworld.wolfram.com/HarmonicNumber.html`
25. Xu, J., Kumar, A., Yu, X.: On the fundamental tradeoffs between routing table size and network diameter in peer-to-peer networks. IEEE Journal on Selected Areas in Communications **22**(1), 151–163 (2004)

Part IV
Search and Query Processing

Keyword Search in Unstructured Peer-to-Peer Networks

Dingyi Han and Yong Yu

Abstract Keyword search is a preliminary application for peer-to-peer (P2P) networks. It is important for users to find relevant resources in the highly dynamic system. Many factors affect the results of keyword search, including the underlying structure of peer-to-peer network, the dynamics of peers, the distribution of resources, etc.

This chapter introduces the methods used for keyword search in unstructured peer-to-peer networks, and further discusses their extensions for multi-keyword search. These methods can be categorized into two types. One is blind routing. Methods of this type do not consider the distribution of resources. Hence, they get the name of "blind routing". These methods are typically robust. However, their network traffics are high. The other is routing indices. Methods of this type exploit the distribution of resources or query keywords. Therefore, they have low network traffic, especially for popular resources or queries.

For each method, an algorithm flow is presented, followed by an analysis of the pros and cons. A comparison is also made to demonstrate the differences between these methods.

At the end of this chapter, a discussion on extending the search methods to the multi-keyword search problem is held. The methods introduced in this chapter work differently in the multi-keyword search scenario. Some may need no adaption while some shall be modified for multi-keyword indices. The efficiencies of these methods in this problem are also considered and compared.

Dingyi Han
Apex Data & Knowledge Management Lab, Department of Computer Science & Engineering, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai, 200240, China,
e-mail: handy@apex.sjtu.edu.cn

Yong Yu
Apex Data & Knowledge Management Lab, Department of Computer Science & Engineering, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai, 200240, China,
e-mail: yyu@apex.sjtu.edu.cn

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_15, © Springer Science+Business Media, LLC 2010

1 Introduction

1.1 Background

In unstructured peer-to-peer (P2P) networks, search is a complicated task. Due to the unstructured nature, building global index is almost impossible. Therefore, those search methods relying on a well-designed index is not applicable in unstructured peer-to-peer networks.

However, it is possible to perform search tasks using other methods. In fact, The widely adoption of the idea of peer-to-peer applications begins from the simple yet powerful searching and downloading application named Napster[1] [4]. The application was developed by Shawn Fanning in 1999 and was kept operating around the millennium year. It allowed users to easily share files, especially MP3 format music files. It greatly impacted how people use the Internet and showed the great potential of decentralized peer-to-peer applications. The application built a central index for search music files and their locations in the system, a user submitting a query can know where the results were stored, and then a downloading process could be initialized.

Although the application only transfers files in a peer-to-peer way while still perform searches in a central way, it does inspire many peer-to-peer search methods. Gnutella[2] [15] is among the pioneers. In its early versions, the protocol introduces flooding, which is regarded as the first search mechanisms in unstructured peer-to-peer networks. The protocol forwards a query from a peer to its neighbors for potential results. When a query is sent throughout the whole network, it is expectable that the results can be obtained if exist. From the macro perspective, the query can be regarded as the source of a flood, it is then spread to the whole network. That is why the name “Flooding” is labeled on this method. A property of this method and its variations is that they do not consider the distribution of resources, so-called “Blind Routing”. Therefore their network traffic utility is not optimum. However, they are simple for implementation and they are also robust.

Another type of search methods is called “Routing Indices”. Contrary to blind routing, methods of this type somehow collects information about resources distribution. Some of them build indexes of the resources stored on the local peer or the neighbor peers. Others build indexes of the queries sent by the local peer or the neighbor peers. Therefore, when a query is initialed or received by some peer, it is easy for the peer to decide which neighbor can possibly return good results, as well as locate the relevant resources on the peer itself.

In most search system in unstructured peer-to-peer networks, a single keyword search is a simple task. To make the search more beneficial, multi-keyword query must be supported. There are also several ideas to support such queries. A simple method is to regard multi-keyword query as another type of single keyword query.

[1] http://napster.com/.

[2] http://rfc-gnutella.sourceforge.net/.

This method greatly enlarges the dictionary size while it only need a small modification. Another way is to split the multi-keywords into several single keyword queries, and combine the results of them. Although it may take more time for search, it usually gets better results.

The keyword search approaches in other types of peer-to-peer networks (e.g., structured or semi-structured P2P networks [1]) can be quite different from those in unstructured P2P networks. The approaches introduced in this chapter may probably work in other types of P2P network because they rely little on network structures. However, they are not so efficient as those designed for structured P2P networks because of the different network nature.

1.2 Keyword Search Problem

Keyword search process in P2P systems is just like a torch relay process. A peer would issue a query when its user want to conduct a search task. Typically, a query is composed by a single keyword or several keywords. The peer is called query issuer. Each peer can forward the query message to its neighbor peers once they receive the query. If a peer has found any resources (texts, videos, etc.) relevant to the query, it then composes a query result message and tries to send the result back to the original query issuer. We say the peer has a query hit.

For each message, there are at least five fields, message type, query id, message content, message trail and TTL. Message type indicates whether it is a query or a response. Query id is the unique id generated when the query is initialized, typically, it is randomly generated as an 128 bit hash key. The id is the identifier for all the peers in the network to identify the query and react accordingly. The peers in the network use this field to identify whether the query has been received and whether the query has been forwarded to its neighbors before. Message content contains the keyword query or the results of the query. Message trail records the peer ids that have forwarded the query so that the response can be send back along the reverse order of the peers that forwarded the query. TTL (Time To Live) value implies the lifetime of the query message, it would be decreased by 1 in one hop. If the value is reduces to 0, the message should not be forwarded any more to avoid over spreading the query. Typically, the initial TTL value is less than 10, because there exists small world phenomenon [13] in P2P networks and the diameter of the network is typically less than 10.

Although different keyword search methods may have different behaviors, the search process in P2P networks can still be decomposed into the following five steps.

1. *Query Issuing*. When a peer issues a query, it randomly generates a unique query id. It also sets its peer id in the head of message trail, so that other peers can know where the query comes from and where the results should be sent. Also, a TTL value is also set to restrict the lifetime of the query.

2. *Query Forwarding*. When a peer receives a query, it first decrease the TTL value by 1 and check local resources for relevant results. If the decreased TTL value is larger than zero, the query is then sent to the neighbor peers for further processing after its peer id being appended to the message trail field. The peer also needs to record the message id in its local memory for a short period in case processing the same query again after receiving the same query from other peers. Query forwarding is also called query routing.
3. *Local Lookup*. Once a peer received a query message, it should lookup local resources for relevant resources. If it successfully finds some, a query hit occurs. Then, a query result message should be sent to the query forwarder.
4. *Result Transferring*. If any relevant resources are found, the peer generates the result records and then send it back to the query issuer. Since all the peers forwarding the query has appended their peer ids in the trail field, it is possible to transfer the result in the reverse order of the trail.
5. *Result Merging*. When the query issuer receives the results from its neighbors, it should merge all the results into a ranked list. The step includes group items with the same id, calculating the relevances of items and sorting the list by descending order.

The difference between search methods is the forwarding policy, i.e., how to select among neighboring peers to forward a query message.

What actually happens in the local lookup step is determined by the keyword search application. If the search application is designed to search filenames, local lookup is to check all the names of shared files. If it is designed to search text files, local lookup may check local indexes to find text files containing the query terms. Local lookup is important for a good search results. However, how to obtain a good local search result is an information retrieval [2] issue. It is beyond the scope of this chapter.

1.3 Evaluation Metrics

To evaluate how well the search methods work in unstructured P2P systems, several evaluation metrics are proposed. They reflect different aspects of the search methods.

Coverage. As a query message is forwarded across the whole network, it is important to check how many peers are able to receive the message and to further check their local resources for potential hit. It can be defined as the number of peers receiving the query message divided by the number of peers in the whole network. Typically, a wide coverage means the search gains a better chance to get more results.

Message Count. For a search task, the number of messages is an important metric of network overhead. For two methods with the same coverage, a small message count indicates which is network efficient of the method.

Message Duplication Rate. In unstructured P2P networks, it is still unavoidable for a peer to receive multiple copies of a single query. These copies are duplicated messages. A good method should keep a low message duplication rate.

These metrics focus on evaluating the network performances of keyword search methods in unstructured P2P networks. There are also other metrics focus on evaluating search results, such as precision, recall, F1 score, etc. They are traditional information retrieval metrics. Therefore, they are not the focus of this chapter.

2 Blind Routing

Blind routing methods include flooding, its variations and random walk based methods.

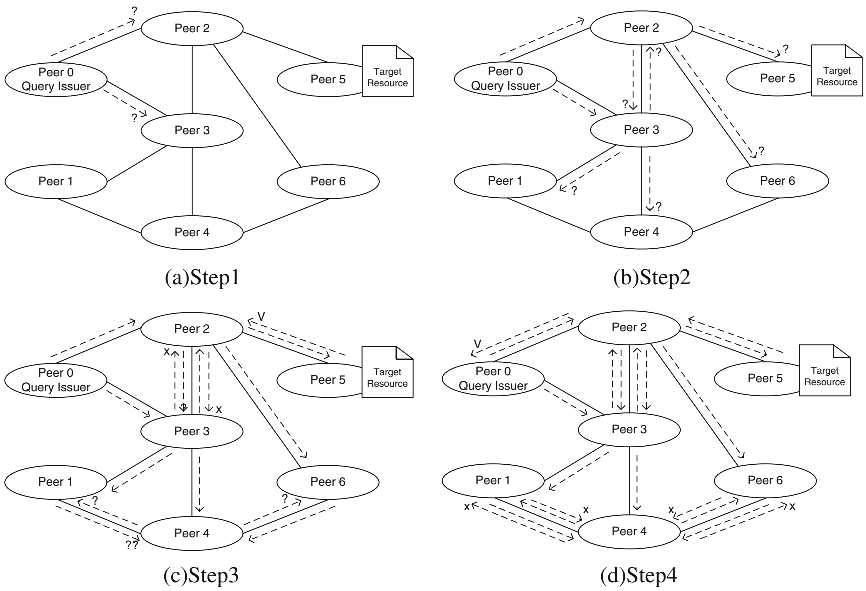
2.1 Pure Flooding and Its Variations

2.1.1 Pure Flooding

The basic idea of flooding is to spread a query across the whole network (or the majority of the network) by forwarding queries to all the neighboring peers. The simplest form of flooding is called pure flooding.

Figure 1 shows an example of the flooding process. The network is composed by 7 peers (Peer 0,1,2,3,4,5,6). Peer 0 sends a query to its neighbors in Step 1. Peer 5 has the relevant resource. In Step 2, both Peer 2 and Peer 3 receive the query and forward the query to its neighbors except Peer 0, where the query is coming from. In Step 3, both Peer 2 and Peer 3 receive the query again, therefore they soon send an error message as the response so that the query is not processed again by one same peer. Peer 1, 4 and 6 act as the same to Peer 2 and 3 in the last step, receive and forward the query. Peer 5 has no neighbors except Peer 2, the one who send the query. Therefore, it will not forward the query. However, it has found there are relevant resources on the peer itself. So it sends a successful query hit message, and transfer the result message to Peer 2, the last peer in the trail of the query. Similarly, three things happened in step 4. The first one is that Peer 2 transfers the query result of Peer 5 to the query issuer, Peer 0. The remainders are that Peer 1 and Peer 6 send the error message like what Peer 2 and Peer 3 have done in the last step. For peers having only one neighbors, such as peer 5, once they receive the query message, they will only response if they have relevant resources. They do not forward queries.

Flooding is a simple blind routing method. It uses query forwarding to make all the peers accessible to a query and the query result can be transferred to the query issuer according to the message trail. However, it is a good method in a highly dynamic environment such as unstructured peer-to-peer network. The query forwarding method ensures the query would be go on forwarding and reach a wide



(a)Step1

(b)Step2

(c)Step3

(d)Step4

Fig. 1 An example of flooding process

coverage even if one or several peers go offline. However, if a peer in the message trail happens to be offline when a query result is being transferred, the result will be unreachable to the query issuer. Going offline means a path from the query issuer and the peer storing relevant resources (query answerer) are broken. If the peer is able to receive the query from other paths, it is still possible for the query issuer to get the results.

The method spends a lot of bandwidth to ensure the robustness. As indicated in Fig. 1, a lot of peers in the whole network would receive multiple copies of a same query message from different neighbors. They have to send error messages to inform the query forwarders. It is very important to choose a proper TTL value. A small TTL value leads to a small coverage. Therefore, the query issuer may not be able to reach the peers having relevant resources. On the other hand, although a large TTL value can ensure a good coverage, it may result in a great number of message duplication, i.e., a great portion of peers in the whole network receive a same query for multiple times. Such a situation may cause network overload, or even crash. As measured in late 2000 and early 2001, in a real Gnutella[3] network containing nearly 50K peers, 95% of peers could be reached within 7 hops (TTL=7) by pure flooding while the diameter of the network is 12 [16].

Pure flooding works in a way that the number of peers involved in a search task increases exponentiallyas TTL value decreases. A major problem is that it is almost

[3] Gnutella is a system using pure flooding as its search method in its early implementations (version 0.4 and before).

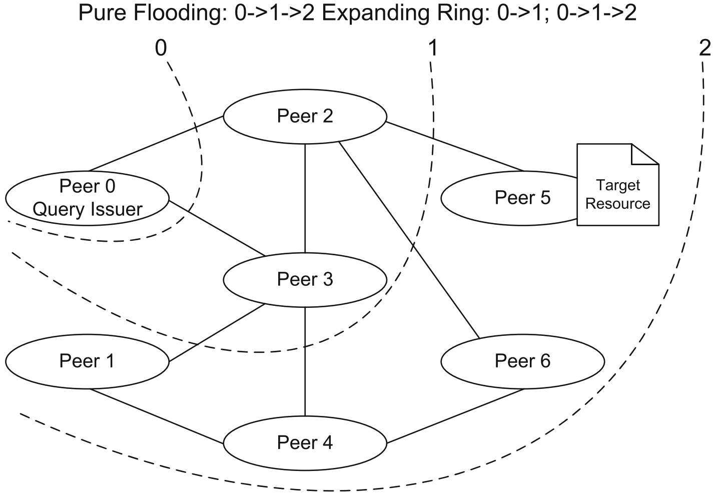


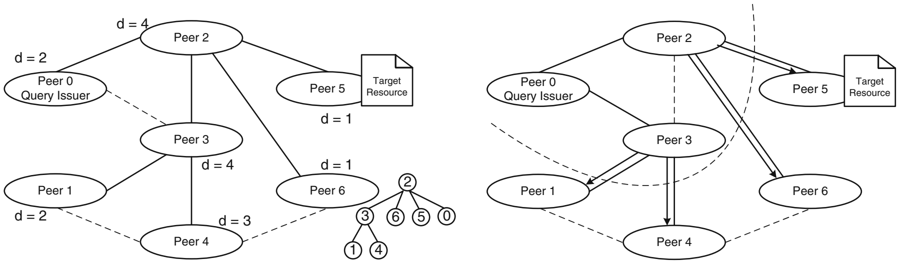
Fig. 2 Pure flooding and expanding ring

impossible to set the TTL value to attain both high coverage and low message duplication. Some researchers try to solve the problem by modify the forwarding policy. Expanding ring [11] and Light flooding [8] are among them.

2.1.2 Expanding Ring

In a lot of cases, it is not necessary to cover the majority of the network to retrieve relevant resources, especially for those popular resources. Therefore, expanding ring [11] utilizes this character to dynamically set the TTL value in pure flooding. More specifically, it first sets a small TTL value (e.g., 1) to cover a small neighborhood, if there are enough relevant resources sent back, the search process is over. However, if the results are not enough, it increases the TTL value and start a new search process. Since there are much more peers involved, hopefully more results can be retrieved. As illustrated in Fig. 2, the dashed curves indicate the edges of the waves in the flooding process. In pure flooding, peer 0 first initialize a wave whose edge is 0, then it is spread to 1, and 2 later. In expanding ring, peer 0 first initialize a wave spread till 1. when it finds out no relevant resources can be retrieved when TTL is 1, it starts another wave spread till 2, and so on.

It is clear that expanding ring try not to involve too many peers if an acceptable result can be retrieved. In most cases, the relevant resources are popular, i.e., they are near the query issuer. Therefore, a query message with a low TTL value is enough to acquire some query hits. However, for those infrequent resources, although expanding ring involves no more peers than pure flooding, it does involve the peers more times than pure flooding.



(a) FloodNet

(b) An example of light flood search process.

Fig. 3 FloodTree and light flood

2.1.3 Light Flood

Light flood [8] is based on the idea of FloodNet, which is a tree-like overlay built on peer-to-peer networks. The overlay can be constructed with local information only. This is an advantage since global information is hard to obtained accurately in P2P networks. The construction process contains two steps.

1. Each peer notifies its neighbors of its connectivity degree.
2. A peer selects the neighbor with the maximum degree as its father, if the peer is not its child.

Figure 3a shows an example of FloodNet. Peer 1 and Peer 4 choose Peer 3 as their father while Peer 5 and Peer 6 choose Peer 2 as their father. Peer 0 randomly chooses Peer 2 as its father since Peer 2 and Peer 3 have the same degree. Peer 2 is not able to choose a father as all its neighbors have chosen Peer 2 as their father. Therefore, Peer 2 becomes the root of the tree.

Constructing such an overlay is beneficial. It ensures that there are no loops in the overlay. Therefore the flooding process can be one way and no peers receive a same query more than once. However, it will greatly increase the diameter of the whole network because many existing neighborhood between peers are ignored. In this case, to cover a majority of the whole network needs a high TTL value. If a query with TTL=7 can cover 85% peers on the original P2P overlay, it may have to be with TTL>25 to attain the same coverage on the FloodNet overlay. Larger TTL value means longer response time, which are drawbacks in a dynamic environment like P2P. Therefore, FloodNet is not good enough to replace pure flooding.

Light flood uses both pure flooding and FloodNet. It first sends a query with small TTL value by pure flooding to ensure a small coverage with quite low message duplication rate. Then, it starts a second step to do FloodNet query forwarding to avoid message duplication while retain a wide coverage.

Although light flood is a flooding based method, it proposes an innovative idea of selectively choosing neighbors to forward queries instead of forwarding to all the neighbors. The idea can help to reduce the message duplication while keeping a similar coverage, compared with pure flooding.

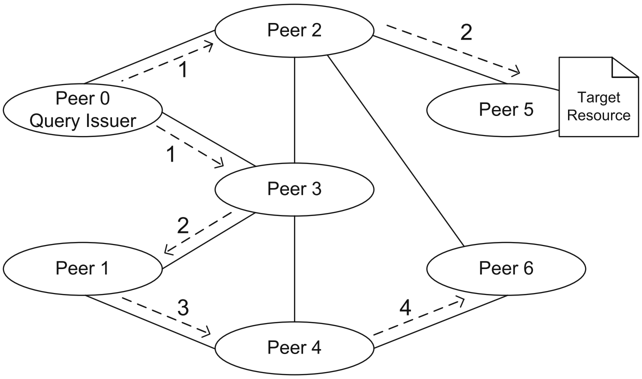


Fig. 4 An example of uniformly random walk

2.2 Random Walk and Its Variations

Another way to selectively choose forwarding neighbors is random walk [7]. In random walk, a query issuer sends out several walker. Each walker choose the next peer by a random strategy with or without certain bias. In this case, the whole system can keep a wide search query coverage while reducing a great number of duplicated messages.

An important aspect of random walk is its randomness. Because its policy of choosing neighboring peers is randomness, sometimes with bias, the search result can be different from time to time even if the whole network remains unchanged. Three typical policies can be chosen. They are uniformly random walk and max-degree biased walk.

2.2.1 Uniformly Random Walk

In uniformly random walk, the simplest method, a walker randomly chooses a neighbor from peer (p_1)'s neighbors (p) . i.e.,

$$P(p_1 \rightarrow p) \propto \frac{1}{n}$$

where n is the number of p_1's neighbors.

Figure 4 is an example of uniformly random walk. Suppose the query issuer starts two walkers. In the first step, $P(p_0 \rightarrow p_2) = P(p_0 \rightarrow p_3) = 1/2$. It is highly possible that they choose peer 2 and peer 3 respectively. In the second step, $P(p_2 \rightarrow p_3) = P(p_2 \rightarrow p_5) = P(p_2 \rightarrow p_6) = 1/3, P(p_3 \rightarrow p_1) = P(p_3 \rightarrow p_2) = P(p_3 \rightarrow p_4) = 1/3$. Suppose peer 2 choose peer 5 and peer 3 choose peer 1. Since peer 5 has only one connection, the walker therefore has no way to go on. It stops and successfully reaches a relevant resource. Peer 5 can now start a feedback message to peer 0

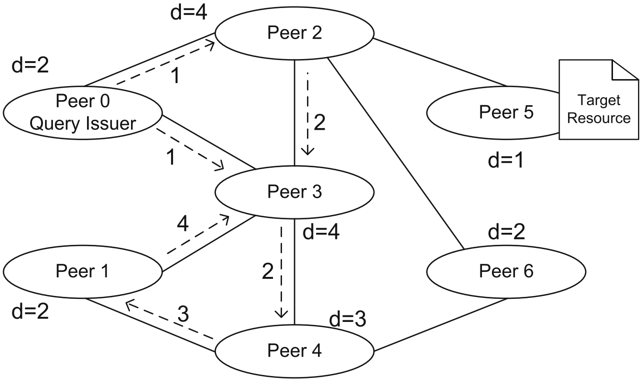


Fig. 5 An example of max-degree biased walk

through peer 2. The other walker can still go on as the third step. $P(p_1 \rightarrow p_4) = 1$ and Peer 1 has no choice but to forward the query to Peer 4. In the fourth step, $P(p_4 \rightarrow p_3) = P(p_4 \rightarrow p_6) = 1/2$. Suppose peer 4 forwards the query to peer 6.

As illustrated in the process, it does send less messages for a query compared with flooding-based method. However, it takes longer steps for a query message from peer 0 to reach peer 6 in uniformly random walk than it does in flooding. That is to say, it may take longer time to get a response if the resource are located at peer 6 instead of peer 5. Longer steps also means lower coverage when a query message is sent by uniformly random walk then it is sent by pure flooding without modifying the TTL value. Also, there is a chance that peer 0 will get no response if peer 2 choose peer 6 instead of peer 5, to forward the query message. It is called missing hits. However, these drawbacks are minor issues compared with the issue of decreasing duplicated query messages. For most queries, their query results are widely distributed across the whole network. Therefore, even if it may have a lower coverage and sometime miss some potential hits, it can still provide an acceptable query results.

2.2.2 Max-Degree Biased Walk

In max-degree biased walk, every walker chooses a neighbor node with high probability if the neighbor has a high degree. i.e.,

$$P(p_1 \rightarrow p) \propto \frac{d_p}{\sum\limits_{p_i \in N(p)} d_{p_i}}$$

where $N(p)$ is the set of p's neighbor peers and d_p is the degree of peer p. The information of a neighbor's degree is periodically exchanging throughout the whole network. Consequently, the query path would be different as the network is evolving.

Figure 5 is an example of max-degree biased walk. Suppose there are still two walkers. The forwarding probabilities are listed as the follows. $P(p_0 \rightarrow p_2) = P(p_0 \rightarrow p_3) = 1/2$. Therefore, it is highly possible they choose peer 2 and peer 3 respectively. $P(p_2 \rightarrow p_3) = 4/7, P(p_2 \rightarrow p_5) = 1/7$, $P(p_2 \rightarrow p_6) = 2/7$. It is highly possible the walker chooses peer 3. Since peer 3 has already received the same query, the walker stops. $P(p_3 \rightarrow p_2) = 4/9$, $P(p_3 \rightarrow p_4) = 3/9 = 1/3$, $P(p_3 \rightarrow p_1) = 2/9$. Suppose this time the walker chooses peer 4 instead of peer 2, the most probable one. Then, $P(p_4 \rightarrow p_1) = P(p_4 \rightarrow p_6) = 2/4 = 1/2$. If it chooses peer 1, the walker is going to stop after the forth step. Since peer 5 does not receive any query message, it is also impossible for peer 0 to get any query hits in this case. There is, however, a small possibility that peer 5 can get the query message if peer 2 chooses peer 5 instead of peer 4.

In the example, lower coverage and missing hits also exists as they are both caused by the idea of selectively choosing neighbors to forward queries. Similar to uniformly random walk, they are minor issues.

2.3 Comparison and Summary

Flooding based methods and random walk based methods are two major types of blind routing search methods. Peers in flooding based networks forwards a received query to all its neighbors while peers in random walk based networks only choose one of its neighbors to go on processing the query.

Therefore, flooding based methods cost much more network resources than random walk based methods. Message count and message duplication rate in random walk based methods are much smaller than they are in flooding based methods. On the other hand, the coverage of random walk based methods is not as good as flooding based methods. For infrequent resources, there is a higher chance that it will not be retrieved in random walk based methods than that in flooding based methods.

For popular resources, expanding ring and max-degree biased walk are usually the best methods among its belonging types. They use the minimum message counts to retrieve relevant resources and they have few duplicated messages. For uncommon resources, light flood and uniformly random walk are typically the best ones. They can cover a wide range by just causing a relatively small number of messages.

3 Routing Indices

In max-feedback biased walk, it uses search history information to help guiding the peers when they are choosing neighbors to forward query messages. It indicates that utilizing extra information other than the network topologies can also help to improve search performance. However, the method still has no idea about which path is possibly the best for a certain query. Therefore, it still belongs the the category of

Table 1 A general comparison of flooding based methods and random walk based methods. (1-pure flooding, 2-expanding ring, 3-light flood, 4-uniformly random walk, 5-max degree biased walk.) For low popularity queries, 4 and 5 maybe unable to retrieve any results.

Query popularity	Metrics	Comparison result
High	Coverage	$5 \leq 4 \leq 2 \leq 3 \leq 1$
	Message count	$5 \leq 4 \leq 2 \leq 3 \leq 1$
	Message duplication rate	$5 \approx 4 \leq 2 \approx 3 \leq 1$
Low	Coverage	$5 \leq 4 \approx 3 \approx 2 \leq 1$
	Message count	$4 \leq 5 \leq 3 \leq 1 \leq 2$
	Message duplication rate	$4 \leq 5 \leq 3 \leq 1 \leq 2$

flooding based methods. Actually, building some indices to guide forwarding query messages is rather helpful. Such methods are called routing indices based methods. There are many types of information that can help improving search performance. The content stored on the peers and the queries sent by the peers are among the representative ones. Accordingly, two types of indices can be built for routing the queries. They are content oriented routing and query oriented routing.

3.1 Content Oriented Routing

3.1.1 Naive Routing Indices for Non-cyclic Topologies

A simplest method of routing indices [6] assumes the network topology is a non-cyclic graph structure. It can be built like FloodNet[8] or other cycle-avoidance topologies. An extension of routing indices for cyclic graph is described later. The method also adopts a global dictionary for both resources and queries. The dictionary maybe a taxonomy if the search application is targeted at finding a class of resources. It can also be a keyword list if the application is targeted at finding resources containing certain keywords.

Naive routing indices use tables for query routings. Each peer maintains a two-dimensional routing table. One dimension is the neighbor peers while the other is the dictionary items. Figure 6 presents an example. It exhibits a non-cyclic topology and a table stored on peer 0. There are two records in the table. One for each neighbor peer. In each record, it counts the total number of accessible resources and the number of accessible resources containing a certain keyword (or belongs to a certain class) through the peer. In this case, peer 0 can learn from the routing indices that it can access 50 more resources through peer 3 than it can do through peer 2. Also, it is better to ask peer 2 when the query is “A” or “C”.

To build and maintain such indices, peers in the system should periodically exchanging their routing indices. In the building stage (Fig. 7), two peers should

Peer	#	A	B	C	D
2	100	20	10	15	30
3	150	10	16	8	50

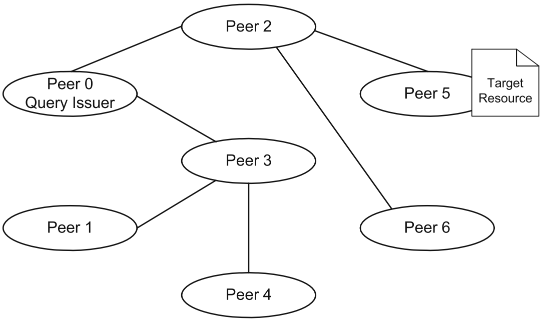


Fig. 6 An example of routing indices

Peer 0:

Peer	#	A	B	C	D
3	150	10	16	8	50
2	100	20	10	15	30

Peer 2:

Peer	#	A	B	C	D
5	40	6	3	12	15
6	20	8	4	0	5
0	200	15	20	10	50

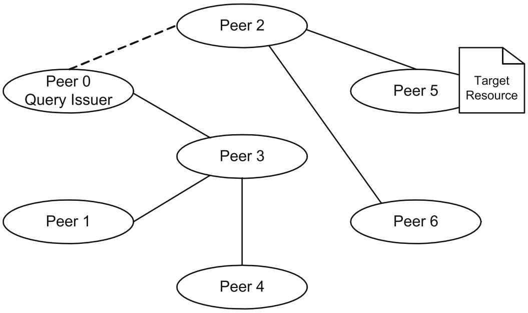


Fig. 7 An example of building routing indices

exchange their routing indices (part of the tables with solid borders) when they become neighbors (establish the dash line connection in the graph).

After the records are created, they should be updated to ensure any changes in the resources allocation and the network topologies are included. The update process is similar to the building process. When a peer notices its indices changed, it just notifies the relevant neighbors for the change. For example, in Fig. 7, if peer 6 add a new resource belongs to B, it should notify peer 2. Then, peer 2 should notify peer 5 and peer 1 for the change. And so on, until all the peers in the system are notified.

Searching by naive routing indices is easy. When a peer initializes or tries to forward a query, it can lookup the indices table to find the peers who contains most promising results. For example, when peer 0 in Fig. 7 wants a resource belongs to C, it can choose the forwarding destination peer from the routing indices. Since peer 2 can reach more "C" resources than peer 3, it will forward the query to peer 2 if it can only forward the query to half of its neighbors.

Maintaining and updating naive routing indices require additional network costs. However, such costs can greatly reduce the network costs needed for query processing. This is ageneral feature of routing indices based methods. Therefore, if the

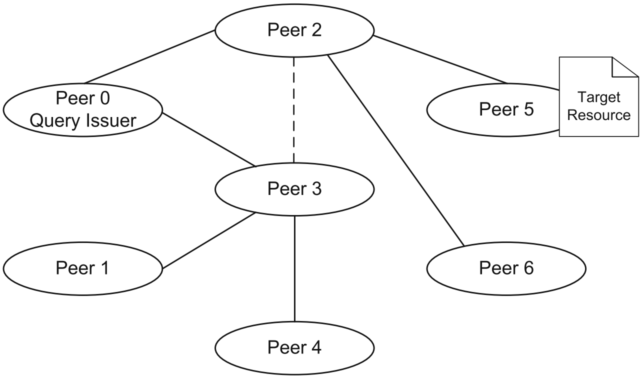


Fig. 8 An example of cyclic topology

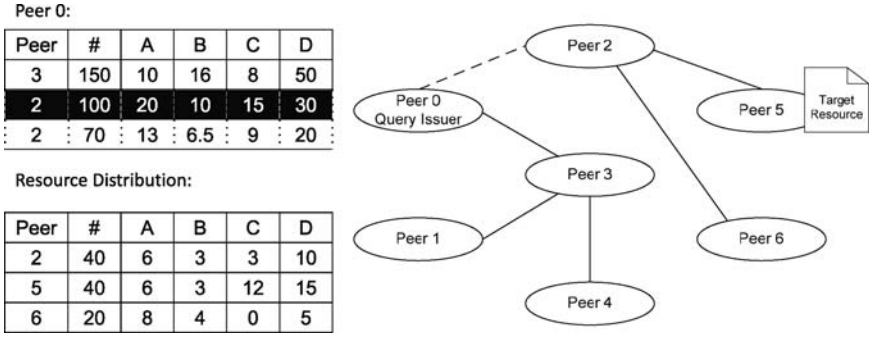
system are targeted at great number of searches, it is better to use routing indices rather than flooding. However, if the system has rare queries, it is not economy to use routing indices, flooding may be a better choice.

Naive routing indices is simple and immature. It does not take several important factors into consideration. For example, what if there is a circle in the topology? What if the resource is quite far from the query issuer? The variances make some modifications to make it better.

3.1.2 Naive Routing Indices for Cyclic Topologies

In cyclic topologies, building and maintaining naive routing indices cannot use the simple method because it will make the routing indices deceitful. Figure 8 illustrates a cyclic topology. The connection between peer 2 and peer 3 is newly established. Peer 0, 2 and 3 forms a circle. Suppose peer 5 adds a “C” file and peer 2 notices. Then peer 2 shall notify both peer 0 and peer 3 that it can access one more “C” file. Peer 0 and peer 3 shall also notify peer 2/3 and peer 0/1/2/4 respectively, and so on. Therefore, the updating process becomes endless and the routing indices get inaccurate. Therefore, there shall be some modifications.

Two modifications can be adopted. One is the cycle avoidance solution, the other is the cycle detection and recovery solution [6]. In the cycle avoidance solution, the major task is to build a non-cyclic overlay (such as FloodNet) over the existing network. In the cycle detection and recovery solution, the major task is to detect the circles and then to eliminate (or neutralize) the effect of the cycles by, for example, mark one connection to avoid the indices update process through the connection.



Peer 0:

Peer	#	A	B	C	D
3	150	10	16	8	50
2	100	20	10	15	30
2	70	13	6.5	9	20

Resource Distribution:

Peer	#	A	B	C	D
2	40	6	3	3	10
5	40	6	3	12	15
6	20	8	4	0	5

Fig. 9 An exponentially aggregated routing indices

3.1.3 Hop-Based Routing Indices

Regardless of non-cyclic topologies or cyclic topologies, previous routing indices do not take the path length into considerations. If there are many resources on one path, and the resources are far away from the query initializer or forwarder, it would be still a good choice for searching these resources. However, actually, it is much better to get the results quickly rather than to get more results slowly in many cases. Methods considering this problem are called hop based routing indices.

One hop replication is a simple example. It is adopted by Gia client[5]. Peers in this system maintain the indices of not only local resources, but also it neighbor's resources. Therefore, queries in this systems are forwarded one hop less than the previous systems.

A more sophisticated method is exponentially aggregated routing indices. The count of each keyword from one path is composed by the count on the peer and the discounted aggregate value of its neighbors. Figure 9 presents an exponentially aggregated routing indices. Suppose the resource distribution is listed in the lower table. When peer 0 asks peer 2 for the routing indices, it uses the following equation to calculate and report.

$$I(p,k) = R(p,k) + \sum_{i=1}^{n} \frac{I(N_i(p),k)}{\alpha}$$

where $I(p,k)$ is the reporting index of resources containing k from peer p. $R(p,k)$ is the count of resources containing k on peer p. $N_i(p)$ is the i-th neighbor of peer p. n is the count of neighbors. α is the discounting rate. It is 2 in the example. The record with black background is the record by original routing indices while the dash border record is generated by exponentially aggregated routing indices.

Hop-based routing indices not only help to retrieve relevant results more quickly, it can also help on solving the problem caused by cyclic topologies in the original routing indices. Since there is a discounting rate, it will not endlessly accumulate the count of the resources on one peer.

3.1.4 Scalable Query Routing

Routing indices are helpful to guide the query issuers and the query forwarders. However, if the system indexes a great number of keywords, the routing indices on each peer may take a large space for storage. On the other hand, most text documents do not contain all the keywords listed in the routing table. Therefore, there are only a small portion of keywords existed on every peer. If the routing indices are stored as a table, there could be a great portion of the indices being 0.

Scalable query routing is [10] designed to use the storage efficiently. The main technology used, exponential decay bloom filter, is an extension of bloom filter [3]. Bloom filter uses k hash functions, $h_1(x), h_2(x), \cdots, h_k(x)$, to convert the keyword x into a k bit value, each bit is either 0 or 1. exponential decay bloom filter still use k hash functions. However, it only converts x into $\theta(x) = |\{i|h_i(x) = 1, i = 1, 2, \cdots, k\}|$, i.e., the number of 1's in the filter[10]. Consequently, the query is also hashed when a peer choosing its neighbors to forward.

Since bloom filter is based on hash functions, it has the same conflict problem. i.e., Two keywords may have the same hash values when the range of hash function is small. Therefore, the value of k should be properly chosen to make the conflict rate low, one in a thousand or even lower. However, such a problem does not affect scalable query routing a lot. Since it can greatly reduce the routing indices, it is the most mature method in content oriented routing.

3.2 Query Oriented Routing

Different to content oriented routing, query oriented routing does not index the resource distribution. Methods in this category index the queries. According to some statistics [9, 12], the popularity of queries in deployed P2P systems fits a Zipf-like distribution. i.e., a small set of query strings represent the majority of the queries. There is a chance that building indices for only those popular queries, instead of all the shared contents, will achieve good routing performance with little network overhead and small local index sizes.

3.2.1 Max-Feedback Biased Walk

In max-feedback biased walk. every walker chooses a neighbor node with high probability if the neighbor has already returned a feedback with many result records before. i.e.

$$P(p_1 \to p) \propto \frac{f_p}{\sum\limits_{p_i \in N(p)} f_{p_i}}$$

where f_p is the number of previous returned result records of peer p.

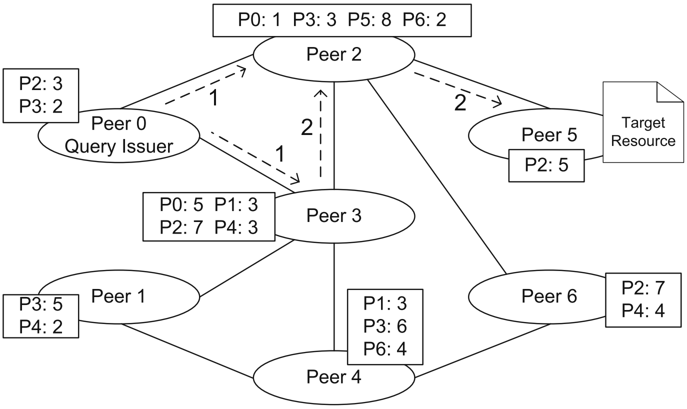


Fig. 10 An example of max-feedback biased walk

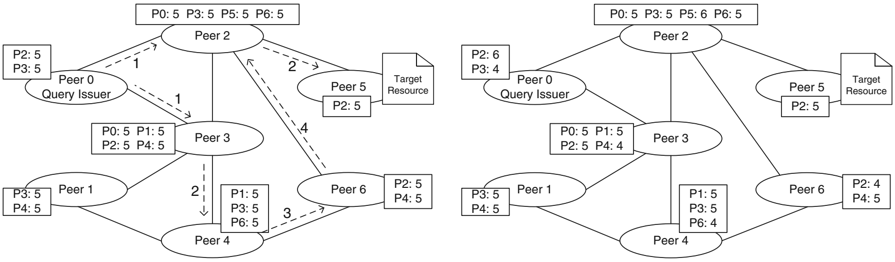
Figure 10 is an example of max-feedback biased walk. Also suppose there are two walkers in the system and the destination of the first steps are peer 2 and peer 3 respectively. Each peer in the system records how many query hit records have been returned by it neighbors (shown in the box in the figure). The forwarding probabilities are listed as the follows. $P(p_2 \rightarrow p_3) = 3/15$, $P(p_2 \rightarrow p_5) = 8/15$, $P(p_2 \rightarrow p_6) = 2/15$, $P(p_3 \rightarrow p_1) = P(p_3 \rightarrow p_4) = 3/13$, $P(p_3 \rightarrow p_2) = 7/13$. The figure is also a most probable process. It is clear that peer 5 get the query message in rather a short time. The history feedback does help to route queries efficiently.

In max-feedback biased walk, the hit record history is collected through the past queries. Therefore, when the system is initialized, it is equivalent to uniformly random walk. Similar to max-degree biased walk, the query path would be different as more queries being processed by the system, because the accuracy of hit record history reflecting the ability of returning hit records is growing.

3.2.2 Adaptive Probabilistic Search

Adaptive probabilistic search [19] is similar to max-feedback biased walk. The difference is that max-feedback biased walk only accumulates positive feedback while adaptive probabilistic search also accumulates negative feedback.

Figure 11a is the initial routing indices of a P2P system. The arrows indicate the routing pathes of a query. After the query is processed, the system updates the indices as shown in Fig. 11b. The index of peer 5 on peer 2 increases by 1, while the index of peer 3 on peer 0 (along with the index of peer 4 on peer 3, the index of peer 6 on peer 4, the index of peer 2 on peer 6) decreases by 1.



(a) The initial routing indices. (b) Routing indices after processed a query.

Fig. 11 An example of adaptive probabilistic search

3.2.3 Learning Based Query Routing

Learning based query routing [17, 18] is an advanced query oriented routing method.

First, it adopts a strategy somewhere between random walk and flood, which forward the query message to only one neighbor and all the neighbors respectively. The number of copies forwarded by a peer is

$$\#query = \frac{\alpha}{hop + 1}$$

where α is a parameter and *hop* is the distance between current peer and the query issuer. Suppose $\alpha = 4$, the query issuer shall send 4 query message copies to its neighbors, and 8 query message copies would be sent by the neighbors. After 4 rounds, there will be totally more than 30 query message copies, and the count stops growing. Such a technique can prevent mass message duplication in the later stage of flooding while keeping enough query paths to ensure reaching the resource holder.

Second, it adopts an advanced indices maintenance technique. The idea comes from reinforcement learning [14]. A successful query hit results in a feed back.

$$f(p_i, q_j) = num_{p_i} + \gamma \times \sum_{m=0}^{M} f(p_m, q_j)$$

where num_{p_i} is the importance of the query hit on peer p_i, and $f(p_m, q_j)$ is the feedback received by p_i from its neighbor p_m. γ is a factor that punishes a hit path by its length. This is a similar technique adopted in hop-based routing indices. Compared with max-feedback bias walk and adaptive probabilistic search, the effect on the weights of feedback is also more complicated.

$$weight_n(p_i, q_j) = \beta_n \times weight_{n-1}(p_i, q_j) + (1 - \beta_n) \times f(p_i, q_j)$$

where $weight_n(p_i, q_j)$ is the n-th weight of choosing peer p_i as the forwarding destination for query q_j. It is calculated when the query has been processed for n times. $\beta_n = \frac{visit_{n-1}(p_i,q_j)}{visit_n(p_i,q_j)}$. $visit_0(p_i, q_j) = 0$. $visit_n(p_i, q_j)$ records the times that q_j has been routed to p_i. It is updated according to the following equation.

$$visit_n(p_i, q_j) = visit_n(p_i, q_j) + \frac{1}{hop_n + 1}$$

Such an technique encourages a larger feedback value if the resource is on the neighbor peer itself rather than far away. It also encourages a shorter query path.

3.3 Comparison and Summary

Routing indices, regardless of content oriented or query oriented, forwards queries selectively according to the indices. They choose a peer because they know it is a better path that have more relevant resources or have been retrieved good resources before. Therefore, they are not “blind”. To build such indices, they have to collect information about resource distribution or query pathes. So these methods are almost equivalent to flooding or random walking at the initial state. The advantage of routing indices will only be visible when the indices are ready.

Content oriented routing indices can be built when the system is online. It is possible that the indices are ready when the first query is issued. Query oriented routing indices can only be built when queries are issued. Therefore, the first hundreds of query issuers are contributing to the system while gaining no or little advantage through the indices.

Most routing indices can work on both flooding based methods and random walk based methods because they can only tell which neighbor can retrieve more relevant resources. Considering the low network cost, it is random walk based methods that are commonly adopted.

A good routing indices should considering the following factors besides the accuracy of the indices.

1. Does a circle in the topology generate a useless index?
2. Is a resource far away from the issuer or forwarder different from a resource near it?
3. Do the indices occupy a large local storage?
4. Are the indices fresh? Is the updating cost acceptable?

Reconsidering the six methods, naive routing indices solves the first problem. Hop-based routing indices solves the second problem. Scalable query routing solves the third problem. Query oriented indices are not affected by the circle topology. They can limit the size of their indices by using first-in-first-out policy to solve the third problem. The last problem can also be solved as new queries being proceed. Max-feedback biased walk adaptive and probabilistic search do not consider the second problem while learning based query routing does. Regarding the accuracy, each of the six methods somehow consider to provide accurate indices. According to simulation results [17], learning based query routing outperforms the others. There is also a detailed comparison in [20].

4 Multi-Keyword Search

Multi-keyword search is an important problem to make keyword search systems more user-friendly. There are several ways to tackle the problem.

The first way is to perform search tasks for each keyword. After receiving all the results, the query issuer can merge the results locally. This technique increases the network cost. To support a 10-keyword query, it has to send 10 single keyword queries to retrieve all the relevant resources for each keyword. The query issuer also has to handle 10 times of search results in one query. This technique shall be adopted if only the search method can not use the other two techniques.

The second way is to regard multiple keywords as a single query. It is a good way in flooding based methods. Compared to the first technique, it can distribute the result merging overhead to all the peers in the network. The query issuer can focus on merging the results from different peers, instead of merging the results for different keywords from different peers. However, for routing indices based methods, the storage spaces will grow. Since a typical search query may contain two or three keywords, the query space can be hundreds or thousands larger than the query space of single keyword. Therefore the storage spaces may also be hundreds or thousands times as larger as the spaces need for single keyword.

The third way is to consider the problem in the query routing process. This technique is designed to solve the storage space problem in the second way for routing indices methods. Since the routing indices based methods can predict the possibility of retrieving relevant resources for a single keyword, it is also possible to predict the possibility for multiple keywords. For example, suppose the keywords in the query $(k_1, k_2, \cdots, k_n)$ are independent, as most information retrieval methods do, the probability of peer p choosing its neighbor p_j as the forwarding destination is calculated as the following equation.

$$p(k_1, k_2, \cdots, k_n, p_j) = \frac{\prod_{i=1}^{n} p(k_i, p_j)}{\sum_{m=1}^{M} \prod_{i=1}^{n} p(k_i, p_m)}$$

where $p(k_i, p_j) = \frac{weight(k_i, p_j)}{\sum_{i=1}^{N} weight(k_i, p_j)}$, N is the size of indexed keywords, and M is the number of peer p's neighbor peers. By using similar estimation methods, this technique does not increase the size of routing indices, which is also an advantage of routing indices based methods.

The three techniques can help to solve multi-keyword search tasks. They introduce some modifications based on the single-keyword search methods. Accordingly they inherit the pros and cons of the single-keyword search methods.

5 Conclusion

In this chapter, two types of keyword search techniques have been introduced. They are blind routing and routing indices. In blind routing based methods, both pure flooding and random walk can work robustly, because they do not make any assumptions on the resource location distribution and/or user query distribution. However, they typically consume more network resources. In routing indices based methods, content oriented routing and query oriented routing are two major ways. Compared with blind routing based methods, they consume less network resources. Several minor issue, including the variations of the proposed methods and their comparison, as well as the multi-keyword search issue are also covered.

References

1. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Computing Survey **36**(4), 335–371 (2004). DOI http://doi.acm.org/10.1145/1041680.1041681
2. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999)
3. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of ACM **13**(7), 422–426 (1970)
4. Carlsson, B., Gustavsson, R.: The rise and fall of napster - an evolutionary approach. In: AMT '01: Proceedings of the 6th International Computer Science Conference on Active Media Technology, pp. 347–354. Springer-Verlag, London, UK (2001)
5. Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S.: Making gnutella-like p2p systems scalable. In: SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 407–418. ACM, New York, NY, USA (2003). DOI http://doi.acm.org/10.1145/863955.864000
6. Crespo, A., Garcia-Molina, H.: Routing indices for peer-to-peer systems. In: ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02), p. 23. IEEE Computer Society, Washington, DC, USA (2002)
7. Gkantsidis, C., Mihail, M., Saberi, A.: Random walks in peer-to-peer networks. In: INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, pp. 120–130 (2004)
8. Jiang, S., Guo, L., Zhang, X.: Lightflood: an efficient flooding scheme for file search in unstructured peer-to-peer systems. In: Proceedings. 2003 International Conference on Parallel Processing, pp. 627–635 (2003)
9. Klemm, A., Lindemann, C., , Waldhorst, O.: Relating Query Popularity and File Replication in the Gnutella Peer-to-Peer Network. In: Proceedings 12th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB). Dresden, Germany (2004)

10. Kumar, A., Xu, J., Zegura, E.: Efficient and scalable query routing for unstructured peer-to-peer networks. INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE **2**, 1162–1173 (2005)
11. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: ICS '02: Proceedings of the 16th international conference on Supercomputing, pp. 84–95 (2002)
12. Meng, S., Shi, C., Han, D., Zhu, X., Yu, Y.: A statistical study of today's gnutella. Lecture Notes in Computer Science **3841/2006**, 189–200 (2006)
13. Milgram, S.: The small world problem. Psychology Today **2**,60–67 (1967)
14. Mitchell, T.M.: Machine Learning. McGraw-Hill Higher Education (1997)
15. Ripeanu, M.: Peer-to-peer architecture case study: Gnutella network. p. 99. IEEE Computer Society, Los Alamitos, CA, USA (2001)
16. Ripeanu, M., Foster, I., Iamnitchi, A.: Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. IEEE Internet Computing Journal **6**, 50–57 (2002)
17. Shi, C., Han, D., Liu, Y., Meng, S., Yu, Y.: A dynamic routing protocol for keyword search in unstructured peer-to-peer networks. Journal of Computer Communications **31**(2), 318–331 (2008). DOI http://dx.doi.org/10.1016/j.comcom.2007.08.009
18. Shi, C., Meng, S., Liu, Y., Han, D., Yu, Y.: Reinforcement learning for query-oriented routing indices in unstructured peer-to-peer networks. p2p **0**, 267–274 (2006)
19. Tsoumakos, D., Roussopoulos, N.: Adaptive probabilistic search for peer-to-peer networks. In: P2P '03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing, p. 102. IEEE Computer Society, Washington, DC, USA (2003)
20. Tsoumakos, D., Roussopoulos, N.: Analysis and comparison of p2p search methods. In: InfoScale '06: Proceedings of the 1st international conference on Scalable information systems, p. 25. ACM, New York, NY, USA (2006). DOI http://doi.acm.org/10.1145/1146847.1146872

Distributed Search and Pattern Matching

Reaz Ahmed and Raouf Boutaba

Abstract Peer-to-peer (P2P) technology has triggered a wide range of distributed applications including file-sharing, distributed XML databases, distributed computing, server-less web publishing and networked resource/service sharing. Despite of the diversity in application, these systems share common requirements for searching due to transitory nodes population and content volatility. In such dynamic environment, users do not have the exact information about available resources. Queries are based on partial information. This mandates the search mechanism to be *flexible*. On the other hand, the search mechanism is required to be bandwidth *efficient* to support large networks. Variety of search techniques have been proposed to provide satisfactory solution to the conflicting requirements of search efficiency and flexibility. This chapter highlights the search requirements in large scale distributed systems and the ability of the existing distributed search techniques in satisfying these requirements. Representative search techniques from three application domains, namely, P2P content sharing, service discovery and distributed XML databases, are considered. An abstract problem formulation called Distributed Pattern Matching (DPM) is presented as well. The DPM framework can be used as a common ground for addressing the search problem in these three application domains.

1 Introduction

Peer-to-peer (P2P) technology has triggered a wide range of distributed systems beyond simple file-sharing. Distributed XML databases, distributed computing, server-less web publishing and networked resource/service sharing are only a few to

Reaz Ahmed
Bangladesh University of Engineering and Technlogy, Dhaka, Bangladesh,
e-mail: reaz@cse.buet.ac.bd

Raouf Boutaba
University of Waterloo, Ontario, Canada, e-mail: rboutaba@bbcr.uwaterloo.ca

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_16, © Springer Science+Business Media, LLC 2010

name. Despite of the diversity in applications, these systems share a common problem regarding search and discovery of information. This commonality stems from the transitory nodes population and volatile information content in the participating nodes. In such dynamic environment, users are not expected to have the exact information about the available objects in the system. Rather queries are based on partial information, which requires the search mechanism to be flexible. On the other hand, to scale with network size the search mechanism is required to be bandwidth efficient. High levels of content and node dynamism in modern large scale distributed systems, including P2P content sharing, service discovery and P2P databases, impose additional requirements on the search mechanism. Flexibility in query expressiveness and fault-resilience of the search mechanism become more important in such environments.

Since the advent of P2P technology experts from industry and academia have proposed a number of search techniques to provide satisfactory solution to the conflicting requirements of search efficiency and flexibility in distributed environment. This chapter will present the challenges in distributed search and the existing search techniques in three well-known P2P application domains: content sharing, service discovery, and distributed XML databases. A generic formulation, namely Distributed Pattern Matching (DPM) problem, will also be presented. The DPM construct can be used as a generic framework for addressing the search requirements in different P2P applications. Two known solutions to the DPM problem will be highlighted as well.

This chapter is organized as follows. Common properties of large scale distributed systems from three application domains, namely, content sharing, service discovery and distributed XML databases, are presented in Section 2. Desirable characteristics of a distributed search mechanism are presented in Section 3, while the components of a distributed search system are identified in Section 4. Representative search techniques form the above mentioned three application domains are investigated in Sections 5–7, respectively. The DPM abstraction is presented in Section 8. Finally, concluding remarks are placed in Section 9.

2 Large Scale Distributed Systems

Networks of tens or hundreds of thousands of loosely coupled devices have become common in today's world. The interconnection networks can exist in physical or logical dimensions as well as wired and wireless domains. The Internet is the largest distributed system that connects devices through TCP/IP protocol stack. On top of this network there exists many logical overlay topologies, where networked nodes federate to achieve a common goal. Examples of such federations include, the Domain Name resolution System (DNS), the World Wide Web (WWW), content sharing P2P systems, world wide service discovery systems and emerging distributed XML database systems. Among these systems, the WWW and the DNS are mature enough and are characterized by relatively static population of hosts.

Content dynamism is also much lower in these two systems, compared to P2P systems. Centralized and clustered search techniques (e.g., web crawlers and proxy caches) work well for a network of relatively stable hosts (or web sites) or domain name resolvers. Decentralized (control) and distributed (workload) search techniques are required for a network composed of transient populations of nodes having intermittent connectivity and dynamically assigned IP addresses.

Content sharing, service discovery and distributed XML databases are three representative P2P application domains that can be taken as the representatives of modern large scale distributed systems. The identifying properties that separate these application domains from contemporary large scale networks, like WWW and DNS, can be summarized as follows:

Population dynamism: Transient population of nodes mandates the routing mechanism to be adaptive to failures. Redundant routing paths and replication can improve availability and resilience in such environments.

Content dynamism: Frequent arrival of new content, relocation (e.g., transfer) of the existing contents and shorter uptime of peers (compared to internet hosts) are the main causes of content dynamism in these systems. Users in these systems often do not have the exact information (e.g., exact filename, or Service Description) about the content they are willing to discover. Rather most of the queries are partial or inexact, which requires the search mechanism to be flexible.

Heterogeneity: In these systems participating population of nodes display wide variation in capacity, e.g., computing power, network bandwidth and storage. This mandates the index information and routing traffic to be distributed based on nodes' capacities.

2.1 Content Sharing

Content (e.g., file) sharing is the most popular P2P application. A classification of the topologies adopted in various P2P content sharing systems can be found in [20]. In [7], a survey and taxonomy of content sharing P2P systems are presented. All content sharing P2P systems offer mechanisms for content lookup and for content transfer. Although content transfer takes place between two peers, the search mechanism usually involves intermediate entities. To facilitate effective search, an object is associated with an index file that contains the name, location, and sometimes a description (or keywords) of the content. Search for a content typically involves matching a query expression against the index files. P2P systems differ in how this index file is distributed over the peers (architecture) and what index scheme is used (i.e., index structure). From an architectural point of view (see Fig. 1), content sharing P2P systems can be *centralized*, *decentralized*, or *partially-decentralized* [7]. *Centralized* P2P systems are characterized by the existence of a central index server, whose sole task is to maintain the index files and facilitate content search. Napster belongs to this category. Centralized P2P systems are highly effective for partial keyword search, but the index system itself becomes a bottleneck and a single point

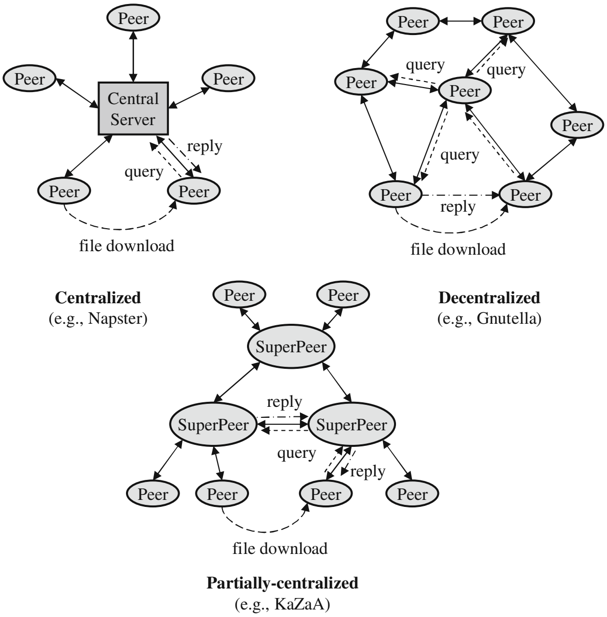


Fig. 1 Content sharing P2P architectures

of failure. *Decentralized* architectures remedy this problem by having all peers index their own local content, or additionally cache the index of their direct neighbors. Content search in this case consists in flooding the P2P network with query messages (e.g.,through TTL-limited broadcast in Gnutella). A decentralized P2P system such as Gnutella is highly robust, but the query routing overhead is overwhelming in large-scale networks. Recognizing the benefit of index servers, many popular P2P systems today use *partially-decentralized* architectures, where a number of peers (called superpeers) assume the role of index servers. In systems such as KaZaA and Morpheus, each superpeer has a set of associated peers. Each superpeer is in charge of maintaining the index file for its peers. Content search is then conducted at the superpeer level, where superpeers may forward query messages to each other using flooding. The selection of superpeers is difficult in such a scheme, as it assumes that some peers in the network have high capacity and are relatively static (i.e., available most of the time). A newer version of Gnutella [71] also uses this approach.

The indexing scheme used by content sharing P2P systems can be categorized as *unstructured*, *semi-structured*, or *structured* [7]. *Unstructured* P2P systems use flat index files, where a index file has no relation to other index files. Napster, Gnutella and KaZaA/Morpheus belong to this category. *Semi-structured* P2P systems, such as Freenet [21] and JXTA [15], use a local routing table at each peer. A search is based on filenames that are hashed to binary keys. The query is routed at each peer to the closest matching key found on the local routing table. To prevent infinite querying, a time-to-live (TTL) value is used. Such mechanism is effective when the content is well replicated over many peers. However, it is virtually impossible to enforce data consistency for file updates. *Structured* P2P systems are specialized in exact matching queries using fully distributed routing structure. Examples of this approach include P2P systems that use distributed indexing and querying schemes, such as Chord [64], CAN (Content Addressable Network) [54] and Tapestry [72].

Advertisements in these systems mostly contain the filename and author-name. For example a movie file can be advertised as *"The Lord of the Rings – The Two Towers – 2002 (Extended Edition) DVDrip.avi"*. For a user it is very unlikely to know the exact name of the advertised file. Rather the user specifies some keywords that may be present in the advertised file name. For example a typical query for the above movie would be *"Lord of the Ring Two Tower"*. Note the keywords *"Ring"* and *"Tower"*; they do not contain the *"s"* as contained in the advertised keywords. This mandates the support for partial keyword matching in P2P content sharing systems.

2.2 Service Discovery

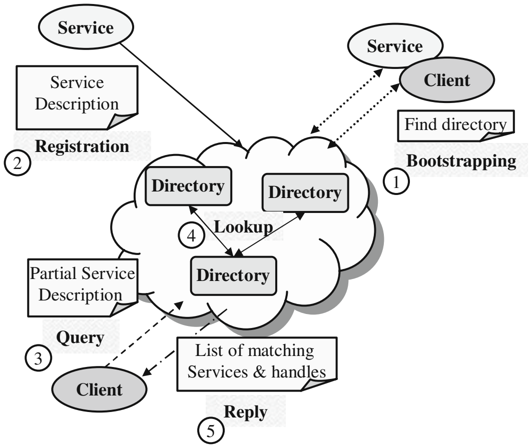


Fig. 2 Service discovery: Generic architecture and steps

Service discovery is an integral part of any service infrastructure. A large-scale service infrastructure requires a service discovery system that is open, scalable, robust and efficient. Most of the service discovery systems rely on a three-party architecture, composed of *client*s, *service*s and *directory* entities. Directory entities gather advertisements from service providers and resolve queries from clients. The generic service discovery mechanism can be viewed as a five-step process (see Fig. 2) [5]; (1) *bootstrapping*, where clients and service providers attempt to initiate the discovery process via establishing the first point of contact within the system, (2) *service advertisement*, where a service provider publishes information (a Service Description containing a list of property-value pairs) to a directory entity about the provided service (3) *querying*, where a client looks for a desired service by submitting a query (usually a partial Service Description) to a directory entity, (4) *lookup*, where the directory entity searches the network of directory entities for all Service Descriptions matching the query and (5) *service handle retrieval*, the final step in the discovery mechanism, where a client receives the means to access the requested service. Some of these steps may be omitted in various discovery approaches. Some of the discovery approaches are based on two-party (client-server) architecture without any directory infrastructure.

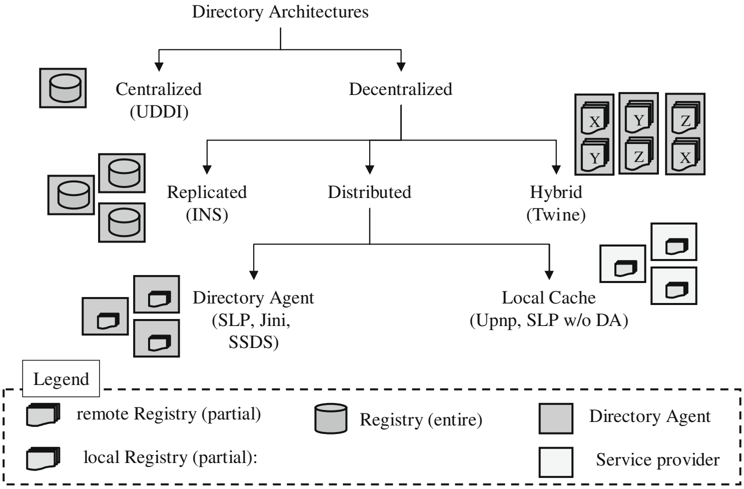


Fig. 3 Taxonomy of the directory architectures

Directory architectures adopted by different service discovery approaches can broadly be classified as *centralized* and *decentralized* (see Fig. 3) [5]. In a centralized architecture, a dedicated directory entity or registry maintains the whole directory information (as in centralized UDDI [69]), and takes care of registering services and answering to queries. In decentralized architectures, the directory information is stored at different network locations. Decentralized systems can be categorized as *replicated*, *distributed* or *hybrid*. In the replicated case, the entire directory information is stored at different directory entities (as in INS [2]). In the distributed case, the directory information is partitioned, and the partitions are either stored in dedicated directory entities (DA) (as in SSDS [24], SLP [30] and Jini [65]) as per a three-party model or cached locally by the service providers in the system (e.g., UPnP [47] and SLP in DA-less mode), according to a two-party model. Finally, in the hybrid case, the system stores multiple copies of the entire directory information without assigning the entire registry to a single directory entity (as in Twine [11]).

In large-scale networks, a centralized directory becomes a performance bottleneck and a single point of failure. Consistency of the replicas is a major issue in the replicated architecture (like INS), since maintaining consistent replicas is usually bandwidth-consuming. On the other hand, when the directory information is distributed, e.g., partitioned among dedicated directory entities, the failure of one of them leads to the unavailability of part of the directory information. The fully distributed two-party architecture attempts to remedy all these issues, however these systems generally do not scale well, since they use multicast-like communications which are expensive in terms of bandwidth. Hybrid architectures seem to offer the best compromise between bandwidth consumption, scalability, and fault-tolerance.

Advertisement	***Query***
Service-type=service:print	**Service-type=service:print**
Scope-list=staff, grad	*Scope-list=grad*
Location=DC3335	*Paper-size=A4*
color=true	
language=PS	
Paper-size= legal, A4, B5	
URL=diamond.uwatreloo.ca/PCL8	

Fig. 4 Example advertisement and query in service discover systems

An example of a generic advertisement and a query in service discovery systems is presented in Fig. 4. In these systems a service is advertised using a list of descriptive property-value pairs, called a *Service Description*. A Service Description typically contains service type (e.g., *Service-type=service:print*), service invocation information (e.g., *URL=diamond.uwatreloo.ca/PCL8*) and service capabilities (e.g., *Paper-size= legal, A4, B5*). In most cases a Service Description is instantiated from a *Service Schema*, which contains meta-information regarding the Service Descriptions for a given class of service (e.g., print service or *service:print*). A Service Schema governs the allowable properties and their types (e.g., string, integer, float, *etc.*) within the Service Descriptions of a given class of services. In most service discovery systems it is assumed that the available Service Schemas are globally known.

Queries in these systems (see Fig. 4) usually contain the requested service type and a list of required capabilities of the service (e.g., Paper-size=A4). The list of capabilities provided in a query is a subset of the capabilities list provided in the advertisements it should match against. The result of a query consists of a list of Service Descriptions matching the query.

2.3 Distributed XML Databases

Distributed XML databases on P2P systems, or P2P Databases Systems (PDBS) in short, have been investigated, more recently, following the success of P2P content sharing. A P2P database system can be thought of as a data sharing network built on top of a P2P overlay. Search in PDBS demands more flexibility than that required by the P2P content sharing systems. This requirement stems from the existence of semantic (schema) information associated with the shared data. Most of the research works focus on building an additional layer on top of the existing P2P search techniques.

Though PDBSs evolved as a natural extension of Distributed Database Systems (DDBS), they have a number of properties that distinguish them from the DDBS and traditional Database Management Systems (DBMS). Unlike DDBS, PDBS has no central naming authority, which results into heterogenous schemas in the system. Due to the absence of any central coordination and the large-scale evolving topology, a peer knows about only a portion of the available schemas and data. This mandates a mechanism (e.g., ontology) for unifying semantically close schemas. In a DDBS, arrival or departure of nodes is performed in a controlled manner, which is not true for PDBSs. Finally, in contrast to DDBS, a peer in a PDBS has full control over its local data.

In PDBS, semantic mapping of schema is a challenging problem. It requires interoperation between heterogenous data models. XML [63] is used as the defacto standard for this purpose. A survey on the use of XML in PDBS can be found in [39]. In PDBS, XML is used in two ways. Firstly, XML is used for representing data and data models (i.e., schema information). Secondly, XML is used to represent semantic relationships among heterogeneous data models at three different levels: schema level, element level and data level. These levels of granularity also influence the indexing mechanism adopted in these systems.

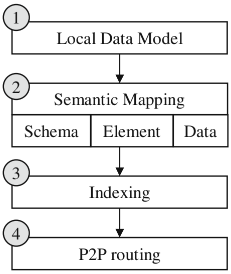
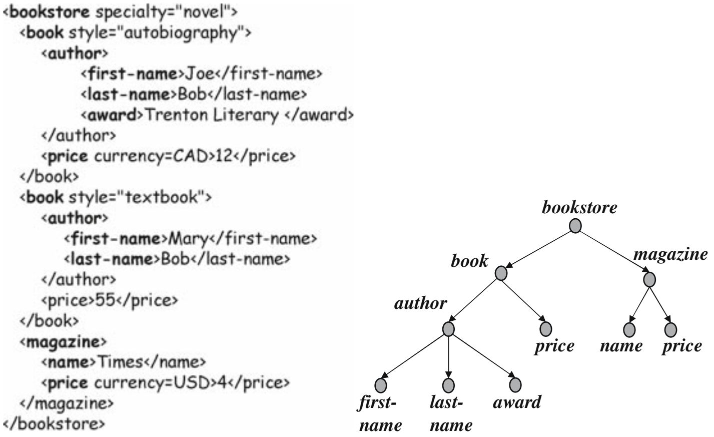


Fig. 5 Functional layers in a PDBS system

Figure 5 presents the possible functional layers in a PDBS. Each peer in the system has its own local data model independent of the other peers' data models. The process of translating a local query to other peers' data models is performed by the semantic mapping layer at different granularities. The third layer is optional, and can maintain indices at different granularities. Finally, the fourth layer is usually one or a combination of the routing mechanisms present in traditional file sharing P2P systems.

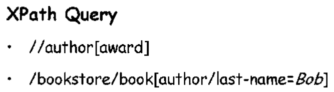
Many research works on PDBS assume the existence of an underlying P2P substrate for efficient and flexible query routing, and concentrate on higher level issues including semantic mapping between heterogenous schemas and distributed query processing and optimization.



```
<bookstore specialty="novel">
  <book style="autobiography">
    <author>
        <first-name>Joe</first-name>
        <last-name>Bob</last-name>
        <award>Trenton Literary </award>
    </author>
    <price currency=CAD>12</price>
  </book>
  <book style="textbook">
    <author>
      <first-name>Mary</first-name>
      <last-name>Bob</last-name>
    </author>
    <price>55</price>
  </book>
  <magazine>
    <name>Times</name>
    <price currency=USD>4</price>
  </magazine>
</bookstore>
```

(a) An XML advertisement (b) Tree representation

Fig. 6 Advertisement in PDBS



XPath Query

- //author[award]
- /bookstore/book[author/last-name=*Bob*]

Fig. 7 XPath query examples

Advertisement and query in PDBS are more complicated than that in P2P content sharing and service discovery systems. Figure 6 depicts an example of an XML advertisement which contains information about two books and one magazine. A tree representation of the corresponding *XML Schema* [26] has been presented in Fig. 6b. Analogous to Service Schema, an XML Schema contains meta-information regarding a class of XML documents. However, the syntax used for describing XML documents and XML Schema are standardized and widely used, compared to the variations in Service Description and Service Schema definition syntaxes used by different service discovery systems.

The most popular query syntax used in PDBS is XPath. Figure 7 presents two examples of XPath queries based on the advertisement presented in Fig. 6. The first query finds all *author*s having at least one *award*. The second example finds all books for which *last-name* of the *author* is *Bob*.

3 Distributed Search Requirements

Search is an essential functionality offered by any distributed system. A search mechanism in a distributed system can be either centralized or distributed. For *Centralized Search* there exists a central core of one or more machines responsible for indexing the contents distributed across the network and for responding to user queries. For networks with lesser degree of dynamism, centralized search mechanisms prove to be adequate. Google, Yahoo, Alta vista, *etc.* are the living examples of centralized search mechanisms, where a set of crawlers running on a cluster of computers index the Webpages around the globe. Compared to the lifetime of the contents shared in P2P networks, Webpages are long lived. Centralized search techniques do not prove to be efficient in large scale distributed systems due to content and population dynamism (as explained in Section 2). *Distributed Search* mechanisms assume that both indexing mechanism (analogous to crawlers) and indexed information are distributed across the network. Consequently the design requirements for Distributed Search techniques are different from that for Centralized Search techniques. In the following is a list of the most important design requirements for a Distributed Search mechanism.

- *Decentralization*: For a Distributed Search mechanism to be successful, decentralization of control and data are necessary. Decentralization of control refers to the distribution of the index construction process among the participating nodes. There should not be any central entity governing the index construction process in different nodes. Unlike web search engines, the index itself should be distributed across the participating nodes for achieving uniform load distribution and fault-resilience.
- *Efficiency*: The search mechanism should be able to store and retrieve index information without consuming significant resource: mainly storage and bandwidth. In a large scale distributed system advertisements are frequent due to the arrival of new documents and relocation of existing documents. The large user base generates queries at a high rate. This mandates both advertisement and search process to be bandwidth efficient.
- *Scalability*: Efficiency of the search mechanism should not degrade with increase in network size. In addition the number of links per node should not increase a lot with growth in network size. Join and topology maintenance overhead depends largely on the number of links that a node has to maintain, especially in dynamic environments.
- *Flexibility*: Due to content dynamism, users do not usually have the exact information about the advertised objects. The query semantics offered by the search mechanism should be flexible to support inexact or subset queries. The scalability and efficiency requirements should not be sacrificed for achieving the flexibility requirement.
- *Search completeness*: Search completeness is measured as the percentage of advertised objects (matching the query) that were discovered by the search. Required level of search completeness varies from application to application. A search mechanism should have guarantee on the discovery of rare objects. In the case of popular or highly replicated objects, only a predefined number of matches would suffice for most cases. For specific queries, the number of matching objects would be low and all of them should be discovered by the search. Broad queries, on the other hand, would match a large number of advertised objects. In this case search result may be restricted within a predefined limit to avoid high bandwidth consumption.

- *Fault-resilience*: In large scale distributed systems, participating nodes connect autonomously without administrative intervention. Nodes depart from the network without a priori notification. The search mechanism is expected to advertise and discover objects in a continuously evolving overlay topology, resulting from the frequent arrival and failure of nodes. In many cases index replication and pair-wise, alternate routing paths are used to improve availability.
- *Load distribution*: Heterogeneity in nodes' capabilities, including processing power, storage, bandwidth and uptime, is prominent in large scale distributed systems. To avoid hot spots and to ensure efficiency, the advertisement and search mechanisms should distribute routing, storage and processing loads according to the capabilities of the participating nodes. In other words, uniform distribution of load may result into poor system performance in a large scale distributed system.

In addition to the above mentioned design requirements, a number of other requirements of secondary importance may arise in different scenarios. For example,

- *autonomy* of index placement and routing path selection may be required for security and performance reasons;
- *anonymity* of the advertising, indexing and searching entities may be required in censorship resistance systems;
- *ranking* of search results may be required for full-text search or information retrieval systems; *etc*.

4 Components of a Distributed Search System

In a large scale distributed system, a distributed search mechanism can be composed of three components as depicted in Fig. 8 and presented in following list.

1. *Query semantics* refer to the expressiveness of a query and the allowed level of semantic heterogeneity in queried and advertised information.
2. *Translation* is a function governing the transformation of semantic information present in a query to a form, suitable for query routing.
3. *Routing* refers to the mechanism of forwarding a query to the nodes suitable for answering the query.

Each of these components are explained in greater detail in the following subsections.

4.1 Query Semantics

Any visible (e.g., shared or advertised) object in a distributed system is associated with a set of properties describing the behavioral and functional aspects of that object. Meta information on a set of related properties associated with a class of objects

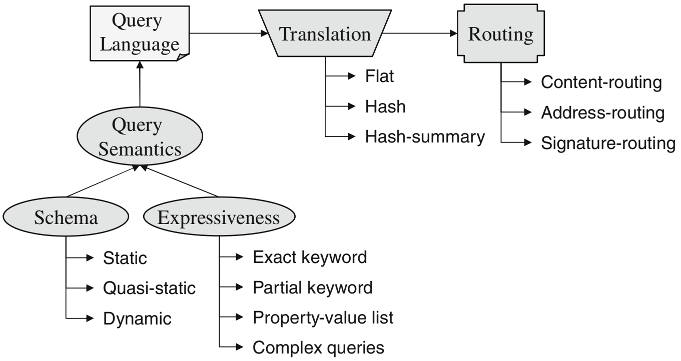


Fig. 8 Components of a distributed search system

is defined as the *schema* for that class of objects. In a distributed search system, structure and scope (temporal and spatial) of the available schemas influence the query language capability and underlying routing mechanism. The rest of this section highlights two aspects of query semantics: *schema* and *query expressiveness*.

4.1.1 Schema

Based on the temporal and spatial scope of the schema, large scale distributed systems can be classified as follows:

- *Static schema*: Most of the file sharing P2P systems have been designed to share one or more specific types of files, e.g., song, movie, software *etc*. For each type of file a specific set of properties is defined that remain unchanged throughout the life of the system. Essentially these systems have one or more static schemas that are globally known.
- *Quasi-static schema*: Most of the service discovery systems fall into this category. Unlike file sharing P2P systems, service discovery systems allow dynamic creation of schema for describing services. Each service instance is advertised as a *Service Description* governed by a predefined *Service Schema* (or template). All schemas in a given service discovery system have to contain a minimal set of predefined *properties* to comply with the specific system under consideration. Though schema can be created dynamically, the rate of such events is very low and the number of available schemas in a given system is much lower than that in PDBS. Furthermore, it is assumed that all the existing schemas in the system are globally known.
- *Dynamic schema*: Most of the PDBSs fall into this category. In these systems heterogeneous schemas exist. Temporal scope of a schema is often bounded by the lifespan of the peer advertising data with that schema. Spatial scope is local to the originating peer and its neighbors; no global knowledge is assumed. In such systems, Automating the process of semantic mapping between similar

schemas is a challenging problem, which may require additional support from the underlying routing mechanism.

4.1.2 Expressiveness

Query expressiveness refers to the capability of the query language in expressing information retrieval requirements. Exiting research works focus on a wide variety of query expressiveness ranging from simple keyword-based queries to complex queries, such as LDAP filter [34] and XPath [38]. Below is a non-exhaustive list of the different levels of query expressiveness commonly found in distributed search techniques.

- *Exact keyword match* is the minimum level of query expressiveness supported by any search mechanism, and is present in most of the file sharing P2P systems, especially the ones based on DHT techniques. For this level of expressiveness, a globally known fixed schema (with a limited number of properties) is assumed.
- *Partial keyword match* is supported by most of the unstructured techniques as well as some extensions to the DHT techniques. Two major variants in this category can be found. Most extensions to DHT techniques support *partial prefix matching* and unstructured techniques support true *partial matching*.
- *Property-value list* is used by many service discovery techniques. Service Descriptions are specified as a property-value list, and queries are specified as a subset of the advertised property-value list. Most service discovery techniques assume a flat list of property-value pairs and do not support wildcard-based partial matching in property names or values.
- *Complex queries* involve logical and relational operators (i.e., *range queries*), and hierarchical relations between properties. Complex queries are supported by a few service discovery approaches and most of the distributed XML database systems. As a means of expressing a distributed query forml query languages, such as LDAP filters, XQuery [18], XPath [38] and SPARQL [51] are used.

4.2 Translation

In most distributed systems the query expression specified by a user is not used *as is* by the underlying routing mechanism. Instead, the query expression goes through some kind of transformation before it is fed to the routing process. The translation function works as a bridge between user specified queries and the routing mechanism. The *domain* of the translation function is governed by the query semantics as discussed in the previous section. The *range* of the translation function, on the other hand, depends on the routing mechanism used by the underlying overlay. Based on the particular combination of query semantics and routing mechanism, this function can exhibit a wide variation. Translation functions can be broadly classified into the following three categories:

- *Flat*: This type of translation functions do a very little (e.g., filtering) or no change to the query expression and associated semantic information. Such functions are usually used by unstructuredand semi-structured routing mechanisms, and most of the industrial approaches to service discovery.
- *Hash* : Hashing is mostly used by structured and semi-structured search mechanisms. A wide variety of hashing techniques have been proposed for distributed search systems. However, the major problem with this type of translation functions is that they loose semantic information during the hash transformation process. As a result only exact or prefix matching is supported by the search mechanisms that adopt hashing as translation function.
- *Hash-summary*: This type of translation enables efficient query routing while preserving query semantics. Variants of Bloom filters are the most popular means of representing hash summaries. Hash summaries are mostly used by unstructured and semi-structured search mechanisms.

4.3 Routing

In overlay networks, routing refers to the process of forwarding a message from a source node to a destination node. The source and the destination nodes are usually at a number of hops away from each other on the overlay. Routing algorithms in overlay networks can be broadly classified into two categories: *uninformed* and *informed*. *Uninformed* routing algorithms do not use the knowledge of query semantics or target node's address in making routing decisions at each hop. *Flooding* [1], *Random walk* [45] and *Iterative deepening* [71] are the representative algorithms in this category. These algorithms are not efficient in terms of the generated volume of search traffic, but the robustness is good in highly dynamic environment. Based on the nature of information used for next hop selection, *Informed* routing algorithms can be classified into the following three categories:

- *Content routing (CR)*: Content routing algorithms utilize the semantic information, embedded in user query, for making routing decisions at each hop. Hence, the associated translation function should be from the *flat* category. Examples of such routing strategy include, selective flooding [22] and hint based routing [68]. Content-routing allows partial match and complex queries, but the offered query routing efficiency is low. Moreover, there exists no guarantee on search completeness or the discovery of rare objects.
- *Address routing (AR)*: Address routing is adopted in DHT-based structured P2P overlays, such as Chord [64], CAN [54], Pastry [57] and Kademlia [46]. Different *hash* techniques are used to transform a query into a virtual address on the overlay, and this address is used to route the query to a responsible node. Routing algorithms in this category are efficient in terms of query routing traffic, but they are not appropriate for semantic laden search (e.g., partial matching and complex queries).

- *Signature routing (SR)*: A number of distributed search techniques, including [3, 24, 42] construct a signature (usually a Bloom filter) of the target object and routes queries based on this signature. These techniques strive to combine the merits of both content-routing and address-routing strategies. Signatures retain (part of or the whole) query semantics and allow aggregation for efficient indexing. However, search completeness and robustness are not as good as that in address-routing and content routing, respectively.

5 Search Techniques in Content Sharing P2P Systems

5.1 Structured Techniques

Majority of the structured search techniques rely on Distributed Hash Tables (DHT). In general DHT-based techniques, like Chord [64], CAN [54], Tapestry [72], are not adequate for supporting flexibility requirement for content sharing P2P systems, which warrant minimum flexibility of partial keyword matching. This inadequacy stems from mainly for two reasons. Firstly, DHT-techniques use numeric distance based clustering of hashed keywords which is not suitable for partial keyword matching. Secondly, DHT-techniques cannot handle *common keywords problem* well. Popular keywords can incur heavy load on the peers responsible for these keywords; as a result, the distribution of load will become unbalanced among the participating peers.

Inability to support partial keyword matching is considered a handicap for DHT-techniques. In the last few years a number of research efforts have focused on extending DHT-techniques for supporting keyword search. Most of these approaches adopted either of the following two strategies:

- Build an additional layer on top of an existing DHT routing mechanism. The aim is to reduce the number of DHT lookups per search by mapping related keywords to nearby peers on the overlay. This strategy is proposed in a number of research works including [37, 44, 61, 67] .
- Combine structured and unstructured approaches in some hierarchical manner to gain the benefits of both paradigms. Few research works, including [28, 36, 66], focus on this strategy.

A *generic inverted index* [31] on top of a DHT-based routing can be used for achieving an expressiveness level of partial-keyword matching. In this approach, a keyword is translated into a routing key in two steps. First, the keyword is fragmented into η-grams. Then each η-gram is hashed and stored at the responsible peer on the DHT overlay. The hashed η-grams form an inverted index, where an advertised η-gram can be discovered by specifying its hash value. This approach will solve partial keyword matching problem in $O(\omega \log N)$ time, where ω is the number of η-grams in a query and N is the number of peers in the system, assuming that the underlying DHT network has logarithmic routing efficiency.

Keyword fusion [44] is another inverted indexing mechanism on top of Chord routing. Supported level of query expressiveness is keyword search only. A document advertised with keywords $\{k_1, k_2, \ldots, k_t\}$ is routed to peers responsible for keys $h(k_1), h(k_2), \ldots, h(k_t)$, where $h(\cdot)$ is the DHT hash function. To reduce the number of DHT-lookups per search, a system-wide dictionary of common keywords is maintained. A query is routed using the most specific keyword and then filtered using the more common keywords specified in the query. Thus the translation function filters out common keywords and then applies hashing. This strategy suffers from two problems. Firstly, the advertisement overhead is significant and proportional to the number of keywords. Secondly, maintaining the global dictionary for common keywords is not suitable for large, dynamic networks.

Joung et al. [37] proposed a distributed indexing scheme, build on a logical, d-dimensional hypercube vector space over Chord routing. In this scheme each advertisement is translated into a d-bit vector according to its keyword set (similar to Bloom filter construction). They treat d-bit vectors as points in d-dimensional hypercube. No restriction on the mapping of a d-dimensional point to a 1-dimensional key space (required for Chord) has been specified. An advertisement is registered to the peer responsible for the d-bit advertisement vector. A query vector (say Q) is computed in the same manner as the advertisement vector. A query is routed to all the peers in the Chord ring that are responsible for a key (say P_i) that is a superset of the query vector Q. Number of DHT lookups per search and query is significant for this approach.

The work by Joung et al. [37] and the inverted indexing method used in Keyword Fusion [44] represent the two extremes of advertisement and query traffic trade off. In [37], an advertisement is registered at one peer (responsible for the advertised key) and a query is routed to all possible peers that may contain a matching advertisement. On the other hand, in Keyword Fusion[44] an advertisement is registered at all the peers responsible for the advertised keywords and the query is routed to the peer responsible for the most uncommon keyword specified in the query.

pSearch [67] utilizes Information Retrieval (IR) techniques to construct the translation function on top of CAN routing for facilitating content-based full-text search. Keywords associated with an advertised document (or query) are represented as unit vectors. IR techniques like vector space model (VSM) and latent semantic indexing (LSI) are used to compute a unit vector from the keyword list specified in an advertisement (or a query). Similarity between a query and an advertisement (or between two advertisements) is measured using the dot product of the vector representation of the corresponding advertisement and query. Semantically close advertisements and queries are expected to be translated to geometrically close point vectors in the Cartesian space. Now the semantic point vectors from LSI or VSM are treated as geometric points in the Cartesian space of CAN. CAN partitions a d-dimensional, conceptual, Cartesian space into zones and assigns each zone to a peer. However this mapping technique (from LSI/VSM to d-dimensional CAN space) uses the same dimensionality for LSI space and CAN. Thus it needs to have a priori knowledge of the possible keywords (or terms) in the whole system. In reality there can be thousands of possible keywords, and CAN performance degrades at higher dimensions.

Squid [61] has been designed to support partial prefix matching and range queries on top of DHT-based structured P2P networks. It uses Hilbert Space-filling Curve (HSFC) [58] for translating keywords to keys on top of Chord routing mechanism. HSFC is a special type of locality preserving hash function that can map points from a d-dimensional grid (or space) to a 1-dimensional curve in such a way that the nearby points in d-dimensional space are usually mapped to adjacent values on the 1-dimensional curve. Squid converts keywords to base-26 (for alphabetic characters) numbers. A d-dimensional point is constructed from d keywords specified in the query or advertisement. Then a d-dimensional HSFC is used to translate a d-dimensional region (i.e., set of points) specified by the query into a set of curve segments in 1-dimension. Finally, each segment is searched using a Chord-lookup followed by a local flooding. Squid supports partial prefix matching (e.g., queries like compu* or net*) and multi-keyword queries; however, Squid does not have provision for supporting true inexact matching of queries like **net**. Another major problem is that the number of (partial) keywords specified in a query or advertisement is bounded by the dimensionality d of the HSFC in use.

MKey [36] is a hybrid approach to keyword search. Architecturally there exists a DHT (here Chord) backbone. A backbone node in the Chord ring works as a head for a cluster of nodes, organized in an unstructured fashion. Search within a cluster is based on flooding. On the other hand, Bloom filter is used as index in the backbone. But DHT techniques do not allow Hamming distance based indexing as required for matching Bloom filters. For allowing pattern matching on Chord, the following strategy is used. Nodes on the Chord ring are allowed to have an ID with at most two 1-bits. An advertisement pattern, say 01010111, is advertised to peers 01010000, 00000110 and 00000001; i.e., DHT-keys are obtained from an advertisement pattern by taking pairs of 1-bits in sequential order from left to right. To construct DHT-keys from a query pattern, say 01010011, only the leftmost three 1-bits are used. In this example the 1-bits at 2nd,4th and 7th positions. The DHT-keys are obtained by taking the 1-bit in center position (here 4th) and another bit within the left position (here 2nd) and the right position (here 7th). Hence for the query pattern 01010011, generated DHT-keys are 01010000, 00110000, 00011000, 00010100 and 00010010. Evidently the number of DHT-lookups per search or advertisement depends linearly on the number of keywords and the size of the used Bloom-filter. This can be more inefficient than a generic inverted indexing mechanism for inappropriate parameter settings. Besides, the nodes on Chord ring may become performance bottlenecks for the system.

There exists only a few non-DHT structured approaches to the search problem in P2P networks. *SkipNet* [32] and *SkipGraph* [9] are prominent among them. Both of these approaches use *Skip List* [52] for routing. A skip List is a probabilistic data structure consisting of a collection of ordered linked lists arranged into levels. The lowest level (i.e., level 0) is an ordinary, ordered linked list. The linked list in level i skips over some elements from the linked list at level $(i-1)$. An element in level i linked list can appear in level $(i+1)$ linked list with some predefined, fixed probability, say p. Storage overhead can be traded for search efficiency by varying p. Search for an element say Q starts at the topmost level. Level i list is sequentially searched until Q falls within the range specified by current element and next element in the list. Then the search recurs to level $i-1$ list from the current element until level 0 is reached. In both SkipGraph and SkipNet, nodes responsible for the upper level elements of the Skip List become potential hot spots and single points of failure. To avoid this phenomena, additional lists are maintained at each level.

5.2 Un-structured and Semi-structured Techniques

Unstructured systems identify objects by keywords. Advertisements and queries are expressed in terms of the keywords associated with the shared objects. Structured systems, on the other hand, identify objects by keys, generated by applying one-way hash function on keywords associated with an object. Key-based query routing is much efficient than keyword-based unstructured query routing. The downside of key-based query routing is the lack of support for partial-matching semantics as discussed in the previous section. Unstructured systems, utilizing blind search methods such as *Flooding* and *Random-walk*, can easily be modified to support partial-matching queries. But, due to the lack of proper routing information, the generated query routing traffic would be very high. Besides, there would be no guarantee on search completeness.

Many research activities are aimed at improving the routing performance of unstructured P2P systems. Different routing hints are used in different approaches. In GIA [19], routing is biased by peer capacity; queries are routed to peers of higher capacity with higher probability. In APS [68, 71], peers learn from the results of previous routing decisions and bias future query routing based on this knowledge. In Associative Search [22], peers are organized based on common interest, and restricted flooding is performed in different interest groups. Many research works (GIA [19], NSS [42, 71], *etc.*) propose storing index information from peers within a radius of 2 or 3 hops on the overlay network. All of these techniques reduce the volume of search traffic to some extent, but none provides guarantee on search completeness.

Bloom filters are used by a few unstructured P2P systems as translation function for improving query routing performance. In NSS [42] each peer stores Bloom filters from peers one or two hops away. Experimental results presented in this work show that logical OR-based aggregation of Bloom filters is not suitable for indexing information from peers more than one hop away. In PLR [55] each peer stores a list of Bloom filters, named Attenuated Bloom filter, per neighbor. The *i*th Bloom filter in the list of Bloom filters for neighbor M summarizes the resources that are $i-1$ hops away from neighbor M. A query is forwarded to the neighbor with a matching Bloom filter at the smallest hop-distance. This approach aims at finding the closest replica of a document with a high probability.

5.3 Summary

Table 1 summarizes the query semantics, translation functions and routing mechanisms as observed in different search techniques in P2P content sharing domain as discussed in this section.

Table 1 Components of selected search techniques in P2P content sharing

P2P content sharing					
References	Name	Query	Translation	Type	Routing Mechanism
[44]	Keyword fusion	Multi-keyword	Inverted index	AR	Chord
[37]	Joung et al.	Multi-keyword	Query superset	AR	Chord
[67]	pSearch	Full text, multi-keyword	VSM/LSI	AR	CAN
[61]	Squid	Prefix match	Hilbert SFC	AR	Chord
[36]	MKey	Subset match	Query superset	AR	Chord +Flooding
[32]	SkipNet	Prefix match	Flat	AR	Skip List
[19]	GIA	Partial keyword	Flat	CR	Capacity bias
[68]	APS	Partial keyword	Flat	CR	Result bias
[42]	NSS	Multi-keyword	Bloom filter (BF)	SR	Controlled flood
[55]	PLR	Multi-keyword	Attenuated BF	SR	Hint bias

6 Search Techniques in P2P Service Discovery

Many service discovery systems rely on a three-party architecture, composed of clients, services and directory entities. Directory entities gather advertisements from service providers and resolve queries from clients. Major protocols for service discovery from industry, like SLP [30], Jini [65], UPnP [47], Salutation [59], etc., assume a few directory agents, and do not provide any efficient mechanism for locating Service Descriptions. Solutions from academia, like Secure Service Discovery Service (SSDS) [24] and Twine [11], target Internet-scale service discovery and face the challenge of achieving efficiency and scalability in locating Service Descriptions based on partial information.

Secure Service Discovery Service (*SSDS*) [24] arranges directory entities in a tree-like structure and uses hierarchy routing. It uses Bloom filters for translating service descriptions into routing signatures. Bit-wise OR-base aggregation scheme is adopted for reducing the volume of index information at higher level directory entities in directory tree. In SSDS an advertisement can be discovered by specifying a subset of the advertised property-value list in the query expression. SSDS suffers from load-balancing problem and is vulnerable to the failure of higher level directory entities along the directory tree.

Twine [11] uses a hierarchical naming scheme and relies on Chord as the underlying routing mechanism. A resource is described using a *name-tree*, composed of the properties and values associated with the resource. Hierarchical relations between properties are reflected in the tree, e.g., while describing the location of a resource, "room no." appears as a child of the "building" in which it resides. The translation function in Twine generates a set of strands (substrings) from the advertisement or query (which are expressed in XML format), computes keys for each of these strands, and finally uses these keys for the search or advertisement process. The stranding algorithm in Twine is designed to support partial prefix matching within a name-tree. The number of DHT-lookups increases with the number of property-value pairs in the advertisement (or query) and consequently the amount of generated traffic becomes high. Load-balancing is another major problem in this system. Peers responsible for small or popular strands become overloaded, and the overall performance degrades.

Web Services (WS) [14] provide a standard way of interoperating between different software applications, running on a variety of platforms and/or frameworks. Universal Description, Discovery and Integration (UDDI) [69] is the defacto standard for WS discovery. Many research activities are devoted to enhancing and overriding the legacy UDDI specification thriving for efficiency, scalability and flexibility in the discovery mechanism. A detailed survey of such activities can be found in [29]. Table 2 summarizes some of the proposed architectures for WS discovery. Based on the use of WS ontologies, these approaches can be broadly classified as *semantic-laden* and *semantic-free*. Semantic-laden approaches rely on WS ontology mapping techniques like OWL (Web ontology language) [8] or DAML (DARPA Agent Markup Language) [16] for incorporating intelligence to the discovery process, i.e., for intelligently mapping conceptually related terms in queries and advertisements. Semantic-free approaches, on the other hand, do not utilize WS ontology mapping techniques. These approaches are closely related to the traditional service discovery systems. A number of research work in this category rely on locality preserving hash techniques for translating queries to semantically close advertisements.

6.1 Summary

Table 3 summarizes the query semantics, translation functions and routing mechanisms as observed in different search techniques in service discovery domain as discussed in this section.

Table 2 Summary of Web service discovery architectures

Centralized	Registry	Authoritative, centrally controlled store of service descriptions, e.g., UDDI registry [69]	
	Index	Non-authoritative, centralized repository of references to service providers; see [14] for details. Web crawlers are used for populating an index database	
Decentralized	Federation	Publicly available UDDI nodes collaborate to form a federation and act together as a large scale virtual UDDI registry [56]	
	P2P-based	Semantic-laden	In [60] peers are arranged into a hypercube topology [25] and ontology [70] is used to facilitate efficient and semantically-enabled discovery. An agent-based approach is proposed in [48]. It uses DAML [16] representation for ontology and relies on unstructured search techniques.
		Semantic-free	Both [43, 62] use Chord overlay for indexing and locating service information. Reference [43] extracts property-value pairs from service descriptions and uses MD5 hashing. Reference [62] uses Hilbert Space Filling Curves for mapping similar Service Descriptions to nearby nodes in the Chord ring. These two approaches are similar to Twine [11] and Squid [61], respectively. In [35], another Chord based solution has been proposed. Here, the ID-space is partitioned in numerically ordered subspaces, and each peer in the Chord-ring maintains links to one peer in each subspace in addition to the regular Chord links.

Table 3 Components of selected search techniques in service discovery

Service discovery					
References	Name	Query	Translation	Type	Routing
[30]	SLP	LDAP filter	Flat	CR	Flooding
[24]	SSDS	Subset/PV-list	Bloom filter	SR	Global hierarchy
[11]	Twine	Subtree match	Stranding + hash	CR	Chord
[43]	PWSD	XML path prefix	Stranding + hash	CR	Chord
[62]	Schmidt et al.	Prefix match	Hilbert SFC	CR	Chord
[60]	Schlosser et al.	Semantic match	Ontology concept → d-coord.	CR+ AR	2-tier hypercube

7 Search Techniques in Distributed XML Databases

Several research works on distributed XML databases have adopted DHT techniques, such as Chord [64], CAN [54] and Hypercube [60], for routing. A number of these proposals, including [13, 17, 27], rely on Chord as the underlying P2P substrate. Hypercube topology has been used in [49].

XP2P [13], uses XML data model for schema representation, and provides support for resolving XPath [38] queries. Any XML document can be represented as a tree, and an XPath query is used to specify a subtree using a prefix-path originating from the root of the document. For supporting partial prefix-path matching, all possible paths, originating from the root, have to be registered with the Chord ring. To reduce the number of paths to be hashed in the Chord ring during the advertisement and query process, XP2P adopts a fingerprint construction technique presented in [53]. In this technique, the fingerprint of a binary string $A(t) = (a_1, a_2, \dots, a_m) = a_1 \times t^{m-1} + a_2 \times t^{m-2} + \cdots + a_m$ is computed as $f(A) = A(t)\%P(t)$, where $P(t)$ is an irreducible polynomial. A useful property of the fingerprint function, utilized by XP2P, is that $f(A \odot B) = f(f(A) \odot B)$, where $\odot$ is the concatenation operator.

Galanis et al. [27] presented a framework for supporting XPath queries on top of Chord routing. XPath queries of the form $/a_1[b_1]/a_2[b_2]/\dots/a_n\, op\ value$ and queries containing relative path operator (i.e., $//$) are supported. Here, a_i is an element in an XML document, b_i is an XPath expression relative to element a_i, op is an XPath operator like $=$ or $<$, and *value* is an atomic element in the XML document. The core idea is to build a distributed catalog, where a peer in the Chord ring stores all the prefix-paths for a given element in any XML document stored in the network. In other words, if E is an element in some XML files, then the peer responsible for the key $hash(E)$ stores all the absolute paths (i.e., $/a_1/a_2/\dots/E$) leading to E in any document stored in the network and the contact information of the peers storing those documents. An XPath query of the form $/a_1/a_2/\dots/a_k//E$ is routed to the peer (say N) responsible for the key $hash(E)$ and the list of all peers containing XML documents matching the query are extracted. Finally the query is forwarded and executed in the corresponding peers.

RDFPeers [17] uses Resource Description Framework (RDF) [41] for document representation and Chord for routing. An RDF document is contains many $< Resource, Property, Value >$ triples presented in XML format. A triple, say $< R, P, V >$, is stored in three peers (in the Chord ring) responsible for the keys $hash(R)$, $hash(P)$ and $hash(V)$, respectively. For string literals SHA1 hash function is used. For numeric values (in the *value* component of a RDF-triple) locality preserving hash function is used. A query can be constructed by specifying any of the three components in a triple. In RDFPeers each document has to be indexed at three peers, which results into increased increased advertisement and update traffic.

PeerDB [50] uses an agent-based framework on top of unstructured P2P overlay to achieve distributed data sharing. To accommodate heterogeneity in schema definitions from autonomous peers in the system, PeerDB associates keywords as synonyms with each schema and elements under that schema. These keywords are used as a means of semantic mapping and for finding semantically similar schemas using P2P keyword search techniques. Mobile agents are sent to appropriate peers and a query is executed locally at the target peer, which helps in reducing the volume of network traffic.

A hybrid technique, named *Humboldt discoverer*, has been presented in [33]. RDF [41] has been used for describing an advertised resource. SPARQL (Simple Protocol and RDF Query language) [51] has been used for constructing query expressions. SPARQL is a query language for RDF documents that allows formation of complex queries involving relational and logical operators. Routing is done using a three tier architecture, where peers are classified as bottom, middle or top tier peers. Bottom tier peers provide information sources. These peers are clustered into many groups based on the similarity of used ontologies. A middle tier peer is responsible for an ontology and manages a single cluster of bottom tier peers. Middle tier peers advertise their existence to top tier peers, which are organized in a Chord ring and are addressed by the hash of the URIs of the ontologies. In effect, middle tier peers covering the same ontology are grouped under the same top level peer. To resolve a query, all the required ontologies are first determined. For a given ontology, the set of responsible middle tier peers can be reached through the top tier Chord network. Finally, the query is forwarded to each of the middle-tier peers that are responsible for the ontology.

7.1 Summary

Table 4 summarizes the query semantics, translation functions and routing mechanisms as observed in different search techniques in distributed XML database domain as discussed in this section.

Table 4 Components of selected search techniques in PDBS

P2P databases					
Ref	Name	Query	Translation	Type	Routing
[13]	XP2P	XPath(absolute)	Fingerprint	AR	Chord
[27]	Galanis et al.	XPath(relative)	XML element hash	AR	Chord
[17]	RDFPeers	Partial RDF triple	RDF element hash	AR	Chord
[50]	PeerDB	SQL	Synonym/flat	CR	Flooding
[33]	Humboldt Discoverer	SPARQL/RDF	URI-hash+Flat	AR+CR	Chord+ Controlled flooding

8 The DPM Abstraction

It is evident from the foregoing survey that flexibility in query expressiveness is essential for a search mechanism in all three domains, considered so far. This section presents an abstraction for the search problems in different application domains into a generic framework or problem formulation, called Distributed Pattern Matching (DPM).

Among the requirements of a search mechanism discussed in Section 3, flexibility deals with query language semantic and the rest relate to system performance and thus are implementation specific. The DPM construct encapsulates the flexibility requirement, and any solution to the DPM problem should focus on the performance specific requirements.

8.1 Distributed Pattern Matching (DPM)

The DPM problem can be considered as a distributed version of the traditional pattern matching problem or more specifically the subset matching problem. The generic pattern matching problem and its variants have been extensively studied in Computer Science literature. Given a *text* (or raw data) $\mathbb{P}$ and a *pattern* $\mathbb{Q}$, the generic problem of pattern matching is to locate (all) the *occurrence*s of $\mathbb{Q}$ in $\mathbb{P}$. The definition of *text*, *pattern* and *occurrence* depends on the application domain. The *text* and *pattern* are two dimensional arrays in Image Processing applications, *string*s in Text Editing systems, trees in *tree pattern matching* [40], and arrays of sets in *subset matching* [23]. Variations in the definition of *occurrence* include exact matching, parameterized matching [10], approximate matching and matching with “don’t cares” [6].

The variant of the pattern matching problem considered in this work is closely related to the *subset matching* problem. In subset matching, pattern $\mathbb{Q} = \{Q_1, Q_2 \dots Q_m\}$ and text $\mathbb{P} = \{P_1, P_2 \dots P_n\}$ are collections of sets of characters drawn from some alphabet $\sum$. A pattern $\mathbb{Q}$ occurs at text position i if the set Q_j is a subset of the set P_{i+j-1}, for all $1 \leq j \leq m$.

The *Distributed Pattern Matching (DPM)* problem is defined as a variant of the *subset matching* problem with the following restrictions:

- The pattern $\mathbb{Q}$ has a single element in its array, i.e., $m = 1$ and $\mathbb{Q} = Q$.
- The n elements of $\mathbb{P}$ are distributed across a large number of networked nodes.

In many cases, bit-vectors of length $|\sum|$ are used to present texts (i.e., P_i) and pattern (i.e., Q), where a 1 (or 0) at the ith bit of a bit vector resembles the presence (or absence) of the ith symbol in $\sum$. In the DPM formulation it is assumed that each element of $\mathbb{P}$ (i.e., P_i) summarizes the identifying properties (e.g., keywords, service description) of a shared object (e.g., a file or a service). One possible form of such a pattern is a Bloom filter [12] obtained by hashing the properties associated with a shared object.

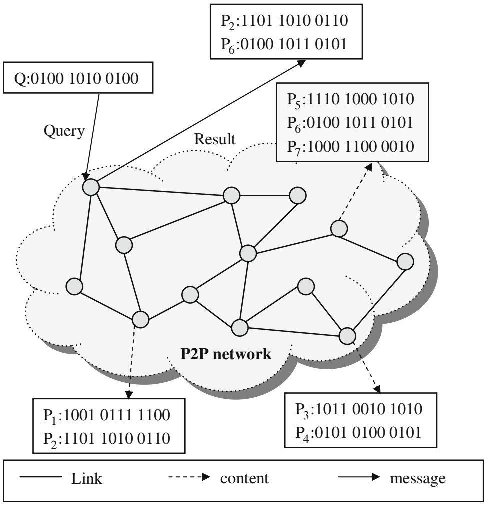


Fig. 9 The distributed pattern matching (DPM) problem. ($|\sum| = 12$)

Figure 9 presents a pictorial view of the DPM problem. In this figure, P_i represents an advertised pattern and Q stands for a search pattern. The properties of shared objects are encoded in the advertisement pattern, $P_1 - P_7$ in Fig. 9. On the other hand, a query (Q in Fig. 9) is constructed by encoding the desired properties in a bit vector. The encoding technique must ensure that if the desired properties specified in a query is a subset of the properties of an shared object then the 1-bits of the corresponding query pattern must be a subset of the 1-bits in the advertised pattern. In other words, the result of a search should contain all the advertised patterns (P_2 and P_6 for the example in Fig. 9) that are supersets of the search pattern (Q in Fig. 9).

8.2 Mapping to DPM Framework

This section describes possible ways of encoding advertisements and queries for the three application domains.

8.2.1 P2P Content Sharing

An advertisement in a P2P content sharing system consists of a number of keywords describing the content being shared. For a file-sharing P2P system, it is unusual for a user to know the exact name of an advertised file. Instead, queries are based on a subset of the (partial) keywords that may be present in the advertisement. Bloom filters can be used for encoding advertised and queried keywords in the following manner. An advertisement Bloom filter can be constructed using the trigrams extracted from the keywords associated with an advertised document. Similarly, a query Bloom filter can be computed from the keywords presented in the query string. Thus there will be one Bloom filter per advertisement or query. Subset relationship between advertised and queried trigrams will hold for advertisement and query Bloom filters. For example, in Fig. 10 trigrams for the first query constitute a subset of the advertised trigrams; as a result query pattern Q_1 is a subset of the advertisement pattern P. On the other hand, trigrams from the second query do not correspond to any subset of the advertised trigrams, and with high probability Q_2 will not be a subset of P.

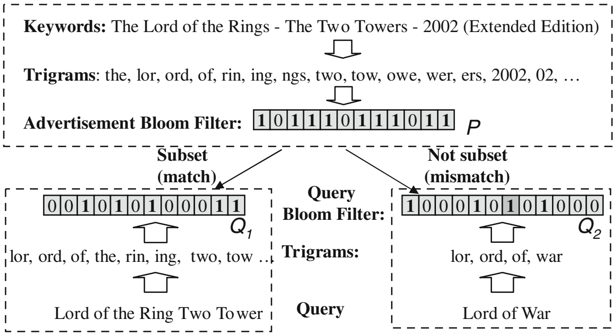

Keywords: The Lord of the Rings - The Two Towers - 2002 (Extended Edition)
Trigrams: the, lor, ord, of, rin, ing, ngs, two, tow, owe, wer, ers, 2002, 02, …
Advertisement Bloom Filter: 1 0 1 1 1 0 1 1 1 0 1 1 P
Subset (match)
Not subset (mismatch)
0 0 1 0 1 0 1 0 0 0 1 1
Q1
lor, ord, of, the, rin, ing, two, tow …
Lord of the Ring Two Tower
Query Bloom Filter:
Trigrams:
Query
1 0 0 0 1 0 1 0 1 0 0 0
Q2
lor, ord, of, war
Lord of War

Fig. 10 Partial keyword matching using DPM

8.2.2 Service Discovery

For most service discovery systems a service description is advertised as a set of property-value pairs and a query for a service consists of a subset of the advertised property-value pairs. All of the decentralized techniques for (Web) Service discovery are aimed at achieving flexibility (i.e., partial matching capability) without sacrificing efficiency in the search mechanism. The basic problem of service description matching and semantic matching in a distributed environment can be mapped to the DPM problem in different ways. A few possibilities are listed below.

- The simplest way of mapping a service description to a Bloom filter is to treat each property-value pair as set elements, see Fig. 11. Query pattern can be constructed in a similar fashion.
- While generating an advertisement pattern from a Service Description, multiple synonyms of the properties and values can be hashed and inserted into the Bloom-filter. This will require larger Bloom-filters, yet will enable one to discover a service by specifying any of the synonyms of a property (or value) specified in the advertisement.
- It is also possible to use ontologies during Bloom filter construction, rather than using simple synonym list. Use of Ontology can be more efficient than synonym list if there exists a global Ontology in the system.

8.2.3 Distributed XML Databases

Distributed search in distributed XML databases faces two new challenges, in addition to the ones present in P2P content sharing and service discovery: continuously changing schemas in the system and the requirement for semantic mapping. It is possible to cope with these challenges using the DPM construct.

For P2P database systems, as shown in Fig. 11, XML documents are used as advertisements and XPath [38] is the most commonly used query language. Figure 11 presents an XPath query of the form $/a_1[b_1]/a_2[b_2]\ldots/a_n[b_n]$. Here, a_i is an element in an XML document, b_i is an XPath expression relative to element a_i. In this case, path prefixes from an XML document or the XPath expression (e.g., $/a_1$, $/a_1/b_1$, $/a_1/a_2\ldots$) can be used as the *set element*s for Bloom filter construction.

In order to accommodate continuously changing schemas, advertised patterns should be constructed from both the descriptive properties and values of a shared object. Thus, schema information gets incorporated within each advertisement. The requirement for semantic mapping can be satisfied in few ways, including the ones discussed in the previous section. It is also possible to reserve a pre-specified number of bits in the advertisement/query pattern, and use these bits as a separate Bloom-filter for storing the ontologies used by the advertisement/query. Query routing mechanism can use this additional information for making semantic laden decisions at each hop.

8.3 Known Solutions to the DPM Problem

An efficient solution to the DPM problem is expected to satisfactory solve the search problem in the three important application domains discussed so far. In this section two solutions to the DPM problem, namely DPMS [3] and Plexus [4], are presented.

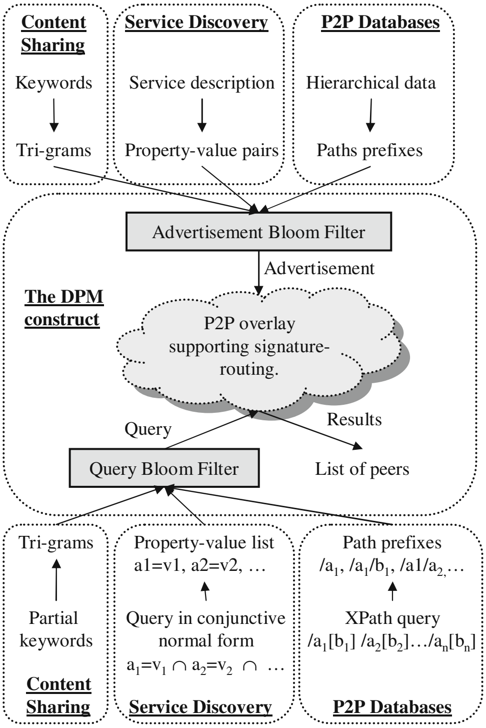


Fig. 11 Mapping different problems to DPM framework

8.3.1 DPMS: Distributed Pattern Matching System

In DPMS a peer can act as a *leaf peer* or *indexing peer*. Leaf peers reside at the bottom level of the indexing hierarchy and act as the document source for the system. An indexing peer, on the other hand, stores indices from other peers (leaf peers or indexing peers). A peer can join different levels of the indexing hierarchy and can simultaneously act in both the roles. Indexing peers get arranged into a lattice-like hierarchy for indexing and disseminating advertised patterns from the leaf peers using repeated aggregation and replication.

The structure of the indexing hierarchy and the amount of replication are controlled by two system-wide parameters, namely replication factor R and branching factor B. Patterns advertised by a leaf peer are propagated to R^l indexing peers at level l. On the other hand, an indexing peer at level l contains patterns from B^l leaf peers. Due to repeated (lossy) aggregation, information content of the aggregates reduces while climbing up the indexing hierarchy.

Indexing peers at level l arrange into R^l groups, numbered from 0 to $(R^l - 1)$ (see Fig. 12). In the ideal case, all the indexing peers in a single group (at any level) collectively cover all the leaf peers in the system. A peer at level l and group

g $(0 \leq g < R^l)$ is responsible for transmitting its aggregated information to R parents at level $(l+1)$. Each parent belongs to a different group in range $[g \times R, (g+1) \times R)$, respectively.

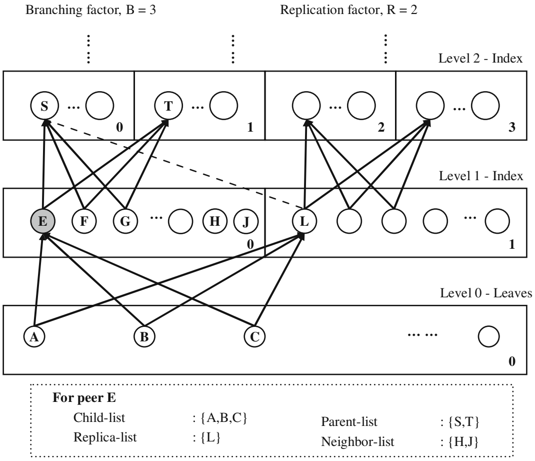


Fig. 12 Index distribution architecture. All the peers interacting with peer *E* are labelled. Group number is printed at the bottom right corner of each box

Peers at level l and group g organize into subgroups (referred to as siblings) of size B to forward their aggregated information to the same set of parents. Thus each group in range $[g \times R, (g+1) \times R)$ at level $(l+1)$ will contain a peer replicating the same index information. This provides redundant paths for query routing and increases tolerance to peer failure.

A don't care based aggregation is used in DPMS. While aggregating to patterns (or aggregate) a don't care ("X") is inserted in the bit-positions at which the constituent patterns disagree. The resulting aggregates retain parts from the constituent patterns or aggregates. A 1-bit (or 0-bit) in such an aggregate indicate that all of the patterns contributing to this aggregate had 1 (or 0) at the corresponding position. However incorporating this extra information (i.e., X's) incur some space overhead, which can be minimized by compressing the aggregates using huffman coding or run length encoding during transmission through the network.

A query is executed in three phases: ascending phase, blind search phase and descending phase. In *ascending phase* the query message climbs the indexing hierarchy until a match is found or it reaches the highest level. If no match is found int he ascending phase the query enters *blind search phase*, where the query message is flooded in one of the groups at the topmost level. If any match exists then it will be discovered in this phase. Finally, in *descending phase* the query message is forwarded to the target leaf peer(s) using the aggregation trail along the indexing hierarchy.

8.3.2 Plexus

Plexus has a *partially decentralized* architecture involving superpeers. It adopts a *structured routing* mechanism derived from the theory of *Linear Covering Codes*.[1] Indexing and routing in Plexus is based on the Hamming distance between the advertised and queried patterns, in contrast to the numeric distance based routing adopted in traditional DHT-approaches. This property makes subset matching capability intrinsic to the underlying routing mechanism. Plexus achieves better resilience to peer failure by utilizing replication and redundant routing paths. Routing efficiency in Plexus scales logarithmically with the number of superpeers.

In Plexus, advertisements and queries are routed to two different sets of peers in such a way that the queried set of peers and the advertised set of peers have at least one peer in common, whenever a query pattern constitute a subset of the 1-bits, as present in an advertised pattern. As explained in Fig. 13, a linear covering code partitions the entire pattern space $\mathbb{F}_2^n$ into Hamming spheres. The codeword at the center of each Hamming sphere is selected as unique representative for that cluster. Now the basic concept is to map a query pattern Q to a set of codewords ($\mathscr{Q}(Q) \subset \mathscr{C}$) and to map an advertised pattern P to another set of codewords ($\mathscr{A}(P) \subset \mathscr{C}$), such that $\mathscr{Q}(Q)$ and $\mathscr{A}(P)$ has *at least one* codeword in common whenever the 1-bits of Q constitute a subset of the 1-bits in P. Mathematically,

$$Q \subseteq P \implies \mathscr{Q}(Q) \cap \mathscr{A}(P) \neq \emptyset \tag{1}$$

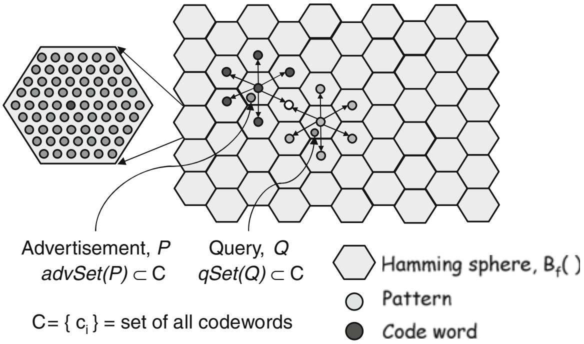
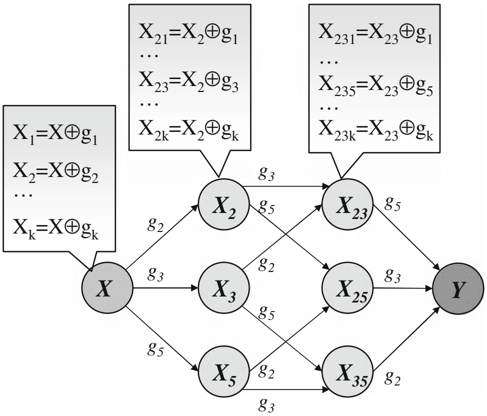


Fig. 13 Hamming distance based indexing in plexus

Any codeword in a linear covering code can be generated by using bit-wise XOR operation of any combination of specially chosen k codewords, labelled

[1] A linear covering code $(n,k,d)f$, over a linear space $\mathbb{F}_2^n$, is a *subspace* $\mathscr{C} \subset \mathbb{F}_2^n$. Each element in $\mathscr{C}$ is called a *codeword*. Here, $|\mathscr{C}| = 2^k$ and d is the minimum Hamming distance between any two codewords. The *covering radius* f is the smallest integer such that every vector $P \in \mathbb{F}_2^n$ is covered by at least one $B_f(c_i)$. Here, $B_f(c_i) = \{P \in \mathbb{F}_2^n | d(P,c_i) \leq f\}$ is the *Hamming sphere* of radius f centered at codeword c_i.

$G = [g_1, g_2, \ldots g_k]$. In other words, any codeword Y can be generated from any other codeword X as follows: $Y = (X \oplus g_{i_1} \oplus g_{i_2} \oplus \ldots \oplus g_{i_t})$, where $g_{i_1}, g_{i_2}, \ldots g_{i_t} \in G$ and $\oplus$ is bitwise XOR operation. For a simple implementation each superpeer in the network is assigned a codeword. A superpeer with codeword X links to k other superpeers with codeword $X_i = X \oplus g_i$, $(1 \leq i \leq k)$. The routing process in Plexus can be best explained by the example in Fig. 14, which shows the possible routes from the superpeer with codeword X to the superpeer with codeword $Y = g_2 \oplus g_3 \oplus g_3$. Superpeer X will forward the message to any of $X_2 (= X \oplus g_2)$, $X_3 (= X \oplus g_3)$ or $X_5 (= X \oplus g_5)$, who are one hop nearer to Y than X. If the message is forwarded to say X_2 then X_2 can route the message to Y via $X_{23} (= X \oplus g_2 \oplus g_3)$ or $X_{25} (= X \oplus g_2 \oplus g_5)$. Number of hops required for routing in this mechanism is logarithmic on the number of superpeers in the network.



$$Y = X \oplus g_2 \oplus g_3 \oplus g_5$$

Fig. 14 Possible paths of routing from peer X to peer Y

8.3.3 Summary

Table 5 summarizes the query semantics, translation functions and routing mechanisms as observed in DPMS and Plexus.

9 Conclusion

Large scale distributed systems including P2P content sharing, service discovery and distributed XML data require flexible, efficient and robust search mechanism due to the volatility in node population and dynamism in advertised objects. Based

Table 5 Components of selected search techniques in service discovery

Distributed pattern matching				
References	Name	Query	Translation	Routing
[3]	DPMS	Subset matching	Feature extraction+Bloom filter	Hierarchical +selective flooding
[4]	Plexus	Subset matching	Feature extraction+Bloom filter	Structured – Linear code based

on routing mechanism contemporary search techniques can be broadly classified as content routing, address routing and signature routing. Content routing techniques preserves query semantics and thus provide better search flexibility; but routing traffic is high for these techniques. Address routing techniques, on the other hand are efficient in routing traffic but offered flexibility in query expressiveness is inadequate. Signature routing can be a good candidate for balancing the trade off between efficiency and flexibility requirements.

The Distributed Pattern Matching (DPM) framework provides an abstract formulation for pattern matching in large scale distributed systems. Keyword search for content-sharing P2P systems, partial Service Descriptions matching for service discovery systems and semantic laden data retrieval for for distributed XML databases can be mapped to the DPM abstraction. DPMS and Plexus are two solutions for the DPM problem. DPMS is a hierarchical signature routing technique, while Plexus is a hamming distance based address routing technique. Routing efficiency in both DPMS and Plexus scales logarithmic on network size. These two solutions as well as future solution to the DPM problem can be tuned to resolve the search problem in the three application domains discussed in this chapter.

References

1. The Gnutella website, http://www.gnutella.com
2. Adjie-Winoto, W., Schwartz, E., Balakrishnan, H., Lilley, J.: The Design and Implementation of an Intentional Naming System. In: Symposium on Operating Systems Principles, pp. 186–201 (1999)
3. Ahmed, R., Boutaba, R.: Distributed pattern matching: A key to flexible and efficient P2P search. IEEE Journal on Selected Areas in Communications (JSAC) **25**(1), 73–83 (2007)
4. Ahmed, R., Boutaba, R.: Plexus: A scalable Peer-to-Peer protocol enabling efficient subset search (2009). IEEE/ACM Transaction on Networking (TON)
5. Ahmed, R., Limam, N., Xiao, J., Iraqi, Y., Boutaba, R.: Resource and service discovery in large-scale multi-domain networks. IEEE Communications Surveys & Tutorials **9**(4) (2007)
6. Amir, A., Porat, E., Lewenstein, M.: Approximate subset matching with don't cares. In: Proc. of Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 305–306 (2001)
7. Androutsellis-Theotokis, S., Spinellis, D.: A survey of Peer-to-Peer content distribution technologies. ACM Computing Surveys **45**(2), 195–205 (2004)

8. Antoniou, G., van Harmelen, F.: Web Ontology Language: OWL. Handbook on Ontologies in Information Systems pp. 76–92 (2003)
9. Aspnes, J., Shah, G.: Skip graphs. In: Proc. of Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 384–393 (2003)
10. Baker, B.S.: A theory of parameterized pattern matching: algorithms and applications. In: Proc. of ACM Symposium on Theory of Computing (STOC), pp. 71–80 (1993)
11. Balazinska, M., Balakrishnan, H., Karger, D.: INS/Twine: A scalable Peer-to-Peer architecture for intentional resource discovery. In: Proc. of International Conference on Pervasive Computing, pp. 195–210. Springer-Verlag (2002)
12. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of ACM **13**(7), 422–426 (1970)
13. Bonifati, A., Matrangolo, U., Cuzzocrea, A., Jain, M.: XPath lookup queries in P2P networks. In: Proc. of the ACM international workshop on Web information and data management (WIDM), pp. 48–55. ACM Press, New York, NY, USA (2004)
14. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.:Web Service Architecture (2004). URL http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/
15. Brookshier, D., Govoni, D., Krishnan, N.: JXTA: Java P2P Programming. SAMS (2002)
16. Burstein, M.H., Hobbs, J.R., Lassila, O., Martin, D., McDermott, D.V., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T.R., Sycara, K.P.: DAML-S: Web Service description for the semantic web. In: Proc. of International Semantic Web Conference on The Semantic Web (ISWC), pp. 348–363. Springer-Verlag, London, UK (2002)
17. Cai, M., Frank, M.: RDFPeers: a scalable distributed RDF repository based on a structured Peer-to-Peer network. In: International World Wide Web Conference (WWW) (2004)
18. Chamberlin, D., Siméon, J., Boag, S., Florescu, D., Fernández, M.F., Robie, J.: XQuery 1.0: An XML query language. W3C recommendation, W3C (2007). http://www.w3.org/TR/2007/REC-xquery-20070123/
19. Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S.: Making Gnutella-like P2P systems scalable. In: Proc. of ACM SIGCOMM, pp. 407–418 (2003)
20. Choon-Hoong, D., Nutanong, S., Buyya, R.: Peer-to-Peer Computing: Evolution of a Disruptive Technology, chap. 2–Peer-to-Peer Networks for Content Sharing, pp. 28–65. Idea Group Inc. (2005)
21. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A distributed anonymous information storage and retrieval system. Lecture Notes in Computer Science (LNCS) **2009**, 46–66 (2001)
22. Cohen, E., Fiat, A., Kaplan, H.: Associative search in Peer-to-Peer networks: Harnessing latent semantics. In: Proc. of IEEE INFOCOM (2003)
23. Cole, R., Harihan, R.: Tree pattern matching and subset matching in randomized $o(n\log^3 m)$ time. In: Proc. of ACM Symposium on Theory of Computing (STOC), pp. 66–75 (1997)
24. Czerwinski, S.E., Zhao, B.Y., Hodes, T.D., Joseph, A.D., Katz, R.H.: An Architecture for a Secure Service Discovery Service. In: Proc. of International Conference on Mobile Computing and Networking (MOBICOM), pp. 24–35 (1999)
25. Decker, S., Schlosser, M., Sintek, M., Nejdl, W.: Hypercup – hypercubes, ontologies and efficient search on P2P networks. In: International Workshop on Agents and Peer-to-Peer Computing (2002)
26. Fuchs, M., Wadler, P., Robie, J., Brown, A.: XML schema: Formal description. W3C working draft, W3C (2001). Http://www.w3.org/TR/2001/WD-xmlschema-formal-20010925/
27. Galanis, L., Wang, Y., Jeffery, S., DeWitt., D.: Locating data sources in large distributed systems. In: Proc. of the VLDB Conference, (2003)
28. Ganesan, P., Sun, Q., Garcia-Molina, H.: Adlib: A self-tuning index for dynamic Peer-to-Peer systems. In: Proc. of the International Conference on Data Engineering (ICDE), pp. 256–257. IEEE Computer Society, Los Alamitos, CA, USA (2005)

29. Garofalakis, J., Panagis, Y., Sakkopoulos, E., Tsakalidis, A.: Web service discovery mechanisms: Looking for a needle in a haystack? In: International Workshop on Web Engineering (2004)
30. Guttman, E., Perkins, C., Veizades, J., Day, M.: Service Location Protocol (SLP), version 2. Tech. rep., IETF, RFC2608, http://www.ietf.org/rfc/rfc2608.txt (1999)
31. Harren, M., Hellerstein, J.M., Huebsch, R., Loo, B.T., Shenker, S., Stoica, I.: Complex queries in DHT-based Peer-to-Peer networks. In: Proc. of International Workshop on Peer-to-Peer Systems (IPTPS), pp. 242–259 (2002)
32. Harvey, N., Jones, M.B., Saroiu, S., Theimer, M., Wolman, A.: SkipNet: A scalable overlay network with practical locality properties. In: Proc. of the USENIX Symposium on Internet Technologies and Systems (USITS) (2003)
33. Herschel, S., Heese, R.: Humboldt Discoverer: A semantic P2P index for PDMS. In: Proc. of the International Workshop Data Integration and the Semantic Web (DISWeb'05) (2005)
34. Howes, T.: The String Representation of LDAP Search Filters. USA (1997). RFC Editor
35. Hu, H., Seneviratne, A.: Autonomic Peer-to-Peer service directory. IEICE/IEEE Joint Special Section on Autonomous Decentralized Systems **E88-D**(12), 2630–2639 (2005)
36. Jin, X., Yiu, W.P.K., Chan, S.H.: Supporting multiple-keyword search in a hybrid structured Peer-to-Peer network. In: Proc. of IEEE International Conference on Communications (ICC), pp. 42–47. Istanbul (2006)
37. Joung, Y., Yang, L., Fang, C.: Keyword search in DHT-based Peer-to-Peer networks. IEEE Journal on Selected Areas in Communications (JSAC) **25**(1), 46–61 (2007)
38. Kay, M., Fernández, M.F., Boag, S., Chamberlin, D., Berglund, A., Siméon, J., Robie, J.: XML path language (XPath) 2.0. W3C recommendation, W3C (2007). Http://www.w3.org/TR/2007/REC-xpath20-20070123/
39. Koloniari, G., Pitoura, E.: Peer-to-Peer management of XML data: issues and research challenges. ACM SIGMOD Record **34**(2), 6–17 (2005)
40. Kosaraju, S.R.: Efficient tree pattern matching. In: Proc. of the 30th IEEE Symposium on the Foundations of Computer Science (FOCS), pp. 178–183 (1989)
41. Lassila, O., Swick, R.R.: Resource description framework (RDF) model and syntax specification. supersed work, W3C (1999). Http://www.w3.org/TR/1999/REC-rdf-syntax-19990222
42. Li, M., Lee, W., Sivasubramaniam, A.: Neighborhood signatures for searching P2P networks. In: Proc. of Seventh International Database Engineering and Applications Symposium (IDEAS), pp. 149–159 (2003)
43. Li, Y., Zou, F., Wu, Z., Ma, F.: PWSD: A scalable web service discovery architecture based on Peer-to-Peer overlay network. In: Proc. APWeb, Lecture Notes Ccomputer Science (LNCS), vol. 3007 (2004)
44. Liu, L., Ryu, K.D., Lee, K.: Supporting efficient keyword-based file search in Peer-to-Peer file sharing systems. In: Proc. of GLOBECOM (2004)
45. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured Peer-to-Peer networks. In: Proc. of the International Conference on Supercomputing (ICS), pp. 84–95 (2002)
46. Maymounkov, P., Mazireres, D.: Kademlia: A Peer-to-Peer information system based on the XOR metric. In: Proc. of International Workshop on Peer-to-Peer Systems (IPTPS), pp. 53–65. Springer-Verlag (2002)
47. Miller, B.A., Nixon, T., Tai, C., Wood, M.D.: Home networking with Universal Plug and Play. IEEE Communications Magazine pp. 104–109 (2001)
48. Montebello, M., Abela, C.: DAML enabled web service and agents in semantic web. In: Workshop on Web, Web Services and Database Systems, Lecture Notes Ccomputer Science (LNCS) (2003)
49. Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I., Loser, A.: Super-peer-based routing strategies for RDF-based Peer-to-Peer networks. Journal of Web Semantics **1**(2), 177–186 (2004)

50. Ng, W.S., Ooi, B.C., Tan, K.L., Zhou, A.: PeerDB: A P2P-based System for Distributed Data Sharing. In: Proc. of the International Conference on Data Engineering (ICDE), pp. 633–644 (2003)
51. Prud'Hommeaux, E., Seaborne, A.: SPARQL query language for RDF. Working Draft WD-rdf-sparql-query-20061004, World Wide Web Consortium (W3C) (2006)
52. Pugh, W.: Skip lists: a probabilistic alternative to balanced trees. Communications of the ACM **33**(6), 668–676 (1990)
53. Rabin, M.: Fingerprinting by random polynomials. Technical report, CRCT TR-15-81, Harvard University (1981)
54. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: Proc. of ACM SIGCOMM, pp. 161–172 (2001)
55. Rhea, S., Kubiatowicz, J.: Probabilistic location and routing. In: Proc. of IEEE INFOCOM (2002)
56. Rompothong, P., Senivongse, T.: A query federation of UDDI registries. In: Proc. of International Symposium on Information and Communication Technologies (ISICT), pp. 578–583 (2003)
57. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale Peer-to-Peer systems. In: Proc. of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware). Heidelberg, Germany (2001)
58. Sagan, H.: Space-filling curves. Springer-Verlag (1994)
59. Salutation Consortium: Salutation architecture specification version 2.0c. http://www.salutation.org (1999)
60. Schlosser, M., Sintek, M., Decker, S., Nejdl, W.: A scalable and ontology-based P2P infrastructure for semantic web services. In: Proc. of International Conference on Peer-to-Peer Computing (P2P) (2002)
61. Schmidt, C., Parashar, M.: Enabling flexible queries with guarantees in P2P systems. IEEE Internet Computing **8**(3), 19–26 (2004)
62. Schmidt, C., Parashar, M.: Peer-to-Peer approach to web service discovery. In: WWW: Internet and web information systems, vol. 7, pp. 211–229 (2004)
63. Sperberg-McQueen, C.M., Bray, T., Maler, E., Paoli, J., Yergeau, F.: Extensible markup language (XML) 1.0 (fourth edition). W3C recommendation, W3C (2006). Http://www.w3.org/TR/2006/REC-xml-20060816
64. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: a scalable Peer-to-Peer lookup protocol for Internet applications. IEEE/ACM Transaction on Networking (TON) **11**(1), 17–32 (2003)
65. Sun Microsystems: Jini Technology Core Platform Specification (2000). URL `http://www.sun.com/jini/specs/`
66. Tang, C., Dwarkadas, S.: Hybrid global-local indexing for efficient Peer-to-Peer information retrieval. In: Proc. of the Symposium on Networked Systems Design and Implementation (NSDI) (2004)
67. Tang, C., Xu, Z., Mahalingam, M.: pSearch: information retrieval in structured overlays. ACM SIGCOMM Computer Communication Review **33**(1), 89–94 (2003)
68. Tsoumakos, D., Roussopoulos, N.: Adaptive probabilistic search for Peer-to-Peer networks. In: Proc. of International Conference on Peer-to-Peer Computing (P2P) (2003)
69. UDDI Consortium: UDDI Technical White Paper (2002). URL `http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf`
70. Uschold, M., Gruninger, M.: Ontologies: Principles, methods and applications. Knowledge Sharing and Review **11**(2) (1996)
71. Yang, B., Garcia-Molina, H.: Improving search in Peer-to-Peer networks. In: Proc. of International Conference on Distributed Computing Systems (ICDCS) (2002)
72. Zhao, B., Huang, L., Stribling, J., Rhea, S., Joseph, A., Kubiatowicz, J.: Tapestry: A resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications (JSAC) **22**(1), 41–53 (2004)

Distributed Semantic Overlay Networks

Christos Doulkeridis, Akrivi Vlachou, Kjetil Nørvåg, and Michalis Vazirgiannis

Abstract Semantic Overlay Networks (SONs) have been recently proposed as a way to organize content in peer-to-peer (P2P) networks. The main objective is to discover peers with similar content and then form thematically focused peer groups. Efficient content retrieval can be performed by having queries selectively forwarded only to relevant groups of peers to the query. As a result, less peers need to be contacted, in order to answer a query. In this context, the challenge is to generate SONs in a decentralized and distributed manner, as the centralized assembly of global information is not feasible. Different approaches for exploiting the generated SONs for content retrieval have been proposed in the literature, which are examined in this chapter, with a particular focus on SON interconnections for efficient search. Several applications, such as P2P document and image retrieval, can be deployed over generated SONs, motivating the need for distributed and truly scalable SON creation. Therefore, recently several research papers focus on SONs as stated in our comprehensive overview of related work in the field of semantic overlay networks. A classification of existing algorithms according to a set of qualitative criteria is also provided. In spite of the rich existing work in the field

Christos Doulkeridis
Department of Informatics, Athens University of Economics and Business (AUEB), 10434 Athens, Greece, e-mail: cdoulk@aueb.gr

Akrivi Vlachou
Department of Informatics, Athens University of Economics and Business (AUEB), 10434 Athens, Greece, e-mail: avlachou@aueb.gr

Kjetil Nørvåg
Department of Computer and Information Science, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway, e-mail: Kjetil.Norvag@idi.ntnu.no

Michalis Vazirgiannis
Department of Informatics, Athens University of Economics and Business (AUEB), 10434 Athens, Greece, e-mail: mvazirg@aueb.gr

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_17, © Springer Science+Business Media, LLC 2010

of SONs, several challenges have not been efficiently addressed yet, therefore, future promising research directions are pointed out and discussed at the end of this chapter.

1 Introduction

As data generation becomes increasingly distributed, either due to user-generated content (multimedia-images/documents) or because of application-specific needs (sensor networks, data streams, etc.), traditional centralized architectures fail to address the new challenges of contemporary data management. The massive amounts of distributed data, such as digital libraries and web accessible text databases, motivate researchers to work towards decentralized infrastructures for efficient data management and retrieval in highly distributed environments.

A promising solution for the design and deployment of distributed, global-scale applications is the exploitation of the peer-to-peer (P2P) paradigm. P2P has emerged as a prominent architecture for searching distributed data repositories, which reside on autonomous and independent sources. The overall challenge is for a large set of cooperative computers to support advanced search mechanisms over vast data collections, thus supporting a wide spectrum of applications.

Contrary to centralized systems and traditional client-server architectures, nodes that participate in a large P2P network store and share data in an autonomous manner. Such nodes can for example be information providers, which do not wish to expose their full data to a client. Therefore, it is challenging to provide efficient and scalable search, in a context of highly distributed content, without necessarily moving the actual contents away from the information providers. Then the problem of lack of global knowledge – in the sense that each peer is aware of only a small portion of the network topology and content – needs to be dealt with.

P2P systems can be distinguished into two main categories: *unstructured* and *structured*. In unstructured P2P, each peer maintains a limited number of connections (also called links) to other neighboring peers in the network. Searching in an unstructured P2P environment usually leads to either flooding queries in the network using a time-to-live (TTL) or query forwarding based on constructed routing indices that give a direction for the search. Examples of such unstructured P2P networks include Gnutella [17] and Freenet [6]. In structured P2P systems, a hash function is used in order to couple keys with objects. Then a distributed hash table (DHT) is used to route key-based queries efficiently to peers that hold the relevant objects. In this way, object access is guaranteed within a bounded number of hops. Examples of popular structured P2P systems are Chord [45], CAN [38], Pastry [40], Tapestry [54].

In general, searching in this context incurs high costs, in terms of consumed network bandwidth and latency. One of the important problems in P2P search is the high number of contacted peers that do not contribute to the final result set. Note that this is also the case for structured P2P networks, which are only efficient for exact search on the indexed key value. Semantic Overlay Networks (SONs) [9] have been proposed as an approach for organizing peers in thematic groups, so that queries can be selectively forwarded to only those peers having content within specific topics. In the case of unstructured P2P systems, SONs enable more efficient query routing to specific peer groups in a deliberate way, instead of blind forwarding. Although this problem is milder in structured P2P systems, as the search cost is logarithmic to the total number of peers, SONs are useful in this context as well, because they can increase the performance of the search [44].

The main topic of this chapter is an overview of existing algorithms and techniques for distributed semantic overlay network generation, in an unsupervised and decentralized manner. We first introduce the notion of semantic overlay networks and we describe the requirements for SON generation in Section 2. In order to illustrate the SON generation process, an approach for unsupervised, distributed and decentralized SON construction, named DESENT [12], is described as representative example in Section 3, which employs distributed clustering of peer contents, respecting the requirements imposed by the distributed nature of the environment [50]. Thereafter, in Section 4, we describe strategies for searching using SONs. We also show how the generated SONs can be exploited in a super-peer architecture, by having each super-peer responsible for a specific SON. Different application scenarios, including P2P web search and P2P image retrieval, are described in Section 5. Then, we provide a survey of existing SON algorithms and techniques in Section 6 and we present a taxonomy of these approaches using meaningful categorization criteria. Finally, in Section 7, we summarize the most important issues related to distributed SON generation and we outline the future research challenges.

2 Semantic Overlay Networks

In this section we describe the notion of semantic overlay networks in detail and we present a list of requirements for SON generation in a distributed P2P setting.

Semantic overlay networks have been originally proposed as an approach for organizing peers in thematic groups with similar contents, so that queries can be selectively forwarded to only those peers having content within specific topics. It should be mentioned that SONs do not necessarily imply use of semantics in the traditional sense (like ontologies), however this is the term first proposed in the literature [9] and it is used as such by all researchers.

An example of semantic overlay networks is illustrated in Fig. 1. In the figure, peers are organized into three different SONs, according to their contents. These are: Sports & Activities, Music & Entertainment and Art & Culture. An important feature is that a peer may belong to more than one SON, as the peer can store data belonging to different thematic groups. For example, one peer has content relevant to both Sports & Activities and Music & Entertainment.

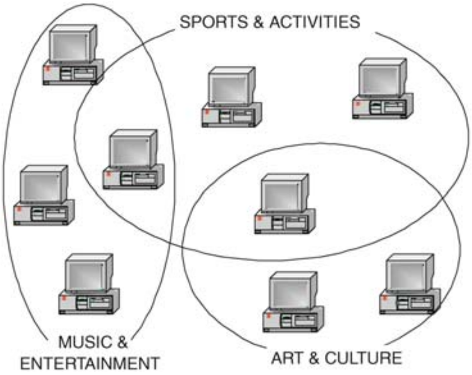


Fig. 1 The notion of semantic overlays in a P2P network

In the following, we present a set of desired goals for semantic overlay networks, in order to provide an assessment of SON quality. Then we proceed to identify and analyze a set of requirements for semantic overlay network generation.

2.1 Aims of SON Generation

The desired features of generated SONs are examined, in order to be able to evaluate and assess their quality. There exist two main measures that assist the assessment: *intra-SON similarity* and *inter-SON similarity*.

2.1.1 Intra-SON Similarity

An individual SON's quality is assessed independently of other SONs by the intra-SON similarity. This measure calculates the similarity of every pair of peers that belong to the SON. High values of intra-SON similarity show that peers contain relevant contents, hence the quality of the SON is high. Low values indicate that peers in the SON have dissimilar contents, therefore the quality of the SONs is not sufficiently high.

2.1.2 Inter-SON Similarity

When examining a set of generated SONs, inter-SON similarity is a suitable quality measure. Essentially, the similarity of every pair of SONs is calculated to determine how similar they are to each other. High values of inter-SON similarity mean that SONs describe common contents, so it is not easy to distinguish them. Low values show indicate that SONs are quite well-separated in terms of their contents, so any given query can be answered by only few SONs.

2.2 Requirements for SON Generation

Although several P2P research papers (see Section 6) adopt the use of semantic overlay networks, they also adopt a set of assumptions that more or less relax the basic constraints imposed by the P2P paradigm. In this section, we go a few steps back, as we identify with the benefit of hindsight from existing approaches the basic requirements for SON generation in a dynamic P2P environment: unsupervised algorithms, scalability, self-organization, autonomy and decentralization. We do not consider this list to be complete, we rather see it as a basic set of requirements that should be enforced, as they increase the value and benefit of any novel SON generation algorithm.

2.2.1 Unsupervised Algorithms

P2P networks in their initial, visionary form are systems characterized by lack of global knowledge. Instead, only local knowledge of content and topology can be safely assumed. However, several approaches make assumptions about the existence of background knowledge, in order to facilitate SON generation. A challenge is to organize the P2P network, assuming minimal pre-existing knowledge.

In principle, clustering algorithms are particularly suitable for SON generation, because they constitute an unsupervised approach. Apart from the input parameters that some clustering algorithms need to execute, distributed clustering of peers' contents assumes no further pre-existing knowledge. Nevertheless, several existing SON generation approaches rely on classification to group similar peers. The difference is that some background knowledge is assumed, usually in the form of a predefined taxonomy or as an already existing labeling scheme. While this assumption sounds reasonable for certain applications, it cannot by any means be generalized and presented as suitable for any P2P system. All in all, a need for unsupervised algorithms and approaches is identified.

2.2.2 Scalability

A unique feature of P2P computing is the unlimited scalability that can be achieved by exploiting the aggregate capabilities of all participating peers. Semantic overlays are proposed as a mechanism that improves the efficiency of search, so any such approach should be scalable. Potential bottlenecks in terms of communication costs (consumed bandwidth, latency, etc.) should be thoroughly studied. In the absence of sufficiently large testbeds, researchers use simulations to test the scalability of proposed systems.

Unstructured P2P networks, such as Gnutella, have problems related to scalability [39]. In particular, the time required to locate content in a large network can be extremely long, with high associated costs. Structured P2P systems solve this issue, by being able to find the answer to a query with logarithmic cost, however their feasibility is still questionable, especially because of the high maintenance cost of keeping the indexes updated. The dynamism of a P2P system, where peers may arbitrarily fail or join the network, poses another threat against ensuring scalable solutions.

Load-balancing is equally important, especially in the cases where the individual peer load may have an aggregate effect, i.e., increase with the size of the network. Any P2P system based on SONs should be able to scale well with the number of peers. Current P2P research focusing on SON generation should put scalability as number one requirement, as this need will become more evident in the future, where the urge for such viable, completely distributed systems is expected to increase.

2.2.3 Self-organization

Informally, the spontaneous activity towards organization of a system is described by the concept of self-organization. The basic mechanism for self-organization is dynamic topology adaptation, as a means to reorganize each peer's neighbors. In this way overlay networks are created on top of the initial P2P network topology. We stress here the essence of self-organization: there is no need for enforcing external observation and maintenance mechanisms. Additionally, it is not necessary for a system administrator to continuously set the values of system parameters or tune the system. Self-organization is one of the most challenging requirements in P2P systems and, at the same time, one of the most difficult to achieve.

2.2.4 Autonomy

Peer autonomy is an important concept in P2P networks, which is indirectly related to other issues like fault-tolerance. Peer autonomy means that each peer can be as independent as possible of the limitations imposed by the P2P protocol, concerning both its behavior and as well as its content. In particular, independence with respect to content means that each peer does not have to replicate its local data or provide explicit indices to its local data to other peers. Moreover, a peer should not be imposed to host indices to data that belongs to other peers. In this sense, unstructured P2P systems respect peer autonomy, in contrast with structured P2P systems. As a consequence, unstructured P2P systems are more resilient to failures, because in general, a peer failure makes only its local content unavailable, while in a DHT-based network, recovery mechanisms must be enforced for consistency. It is important that SON construction algorithms should also respect peer autonomy. However, peer autonomy comes with a cost: it is usually difficult to provide efficient search.

2.2.5 Decentralization

While distribution is inherently related to the P2P concept, the same does not always hold for decentralization. The most evident example of centralized P2P system was Napster, one of the first P2P systems to enable file sharing between participating computers. However, while the actual file exchange was performed in a P2P manner, the index was held in a centralized location. This is not an acceptable approach for dynamic P2P systems, since it presents a single point of failure.

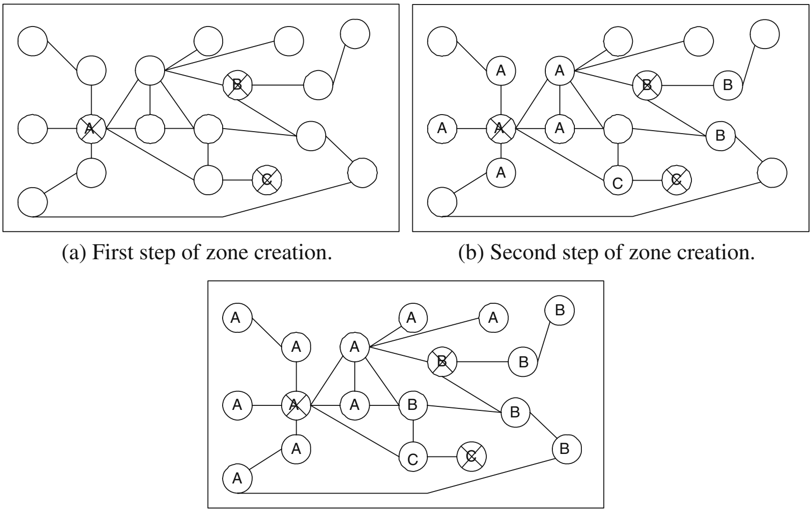
Learning from the shortcomings of such approaches, decentralization should also hold for SON generation, especially in large-scale networks. If operations are centralized, this endangers the completeness of SON generation, with obvious consequences to the correctness of the final overlays. Also, while for small networks a centralized solution may seem appropriate, due to the assembly of global knowledge, where better decisions can be made, however it usually presents problems when applied to large-scale networks. The main reason is communication bottlenecks that often result in non-applicable or infeasible approaches, or in other words algorithms that do not scale.

3 Distributed Creation of Semantic Overlay Networks

In this section we describe how the semantic overlays are created in a decentralized and distributed way using the DESENT approach as described in [12]. The approach is based on creating local topological groups of peers (called *zones*), forming clusters based on data stored on these peers, and then merging zones and clusters recursively until global zones and clusters are obtained. In this approach, clusters and semantic overlays are equivalent, and for the rest of this section we use these terms interchangeably.

3.1 The DESENT Approach to Decentralized and Distributed SON Creation

Having as a starting point the initial unstructured P2P network, some *initiator* peers are selected in a pseudo-random way (*initiator selection phase*). Initiators create local topological zones over their neighboring peers (*zone creation phase*). Then each initiator collects the cluster descriptions of all peers in its zone, and executes a clustering algorithm in order to create new clusters that span the entire zone (*zone clustering phase*). Since the clusters of two (or more) peers may be merged into



(a) First step of zone creation.

(b) Second step of zone creation.

(c) Third step of zone creation.

Fig. 2 Step-wise zone creation given the three initiators A, B, and C

a new cluster, this implies that these peers become members of a SON, and the SON's contents are now represented by a new cluster description. In the subsequent steps, the initiators form the current (unstructured) P2P network, thus playing the role of peers in the initial setup. Therefore, the process described above runs on the initiators, thus new initiators are selected, that create zones and cluster zone contents in a completely similar way as shown above. Hence, zones and clusters are merged recursively until global clusters are obtained.

In order to be able to create zones of approximately equal size throughout the network (S_Z peers in each zone, where a typical zone size is in the order of $S_Z = 50$ peers), a subset of the peers are designated the role of zone initiators that can perform the zone creation process and subsequently initiate and control the clustering process within each zone. The process of choosing initiators is completely distributed, and essentially based on each peer assigning itself the role or not, using a function that is based on a combination of identifier and time. The result is that approximately one out of each S_Z peer assign themselves the role, and these peers are uniformly distributed in the network. One important feature of the initiator selection algorithm is that we obtain different initiators each time the algorithm is run. This tackles the problem of being stuck with faulty initiators as well as reducing the problem of permanent cheaters.

After a peer P_i has discovered that it is an initiator, it uses a probe-based technique to create its zone. An example of zone creation is illustrated in Fig. 2, and as is illustrated an initiator gradually extends its zone until it finds a peer already belonging to another zone. This zone creation algorithm has a low cost wrt. to number

Algorithm 16: Zone-wise clustering

```
1. The initiator of each zone $i$ sends probe messages to all peers $P_j$ in $Z_i$.
2. When a peer $P_j$ receives a probe message it sends its set of feature vectors $\{F\}$ to the
   initiator of the zone.
3. The initiator performs clustering on the received feature vectors. The result is a set of
   clusters represented by a new set of feature vectors $\{F\}$.
4. The initiator selects a representative peer $R$ for each cluster.
5. The result kept at the initiator is a set of cluster descriptions (CDs), one for each cluster $C_i$.
6. Each of the representative peers are informed by the initiator about the assignment and
   receive a copy of the CDs of all clusters in the zone.
7. The representatives then inform peers on their cluster membership by sending them
   $(C_i, F_i, R)$.
```

of messages (in the infrequent case of excessive zone sizes, the initiator can decide to partition its zone, thus sharing its load with other peers). When this algorithm terminates, each initiator has assembled a set of peers Z_i and their capabilities (in terms of resources they possess), each peer knows the initiator responsible for its zone, and each initiator knows the identities of its neighboring initiators. An interesting characteristic of this algorithm is that it ensures that all peers in the network are contacted, as long as they are connected to the network. This is essential, otherwise there may exist peers whose content is never retrieved.

Independently of the actual construction of SONs, local clustering is performed on each peer resulting in a set of clusters. Each cluster is represented by a *feature vector* F_i which is a vector of tuples, each tuple containing a feature (word) f_i and a weight w_i. In order to reduce both computational and communication cost, only the top-k features are kept in F_i. After the zones and their initiators have been determined, global clustering starts by collecting feature vectors from the peers (one feature vector for each cluster on a peer) and creating clusters based on these feature vectors using Algorithm 16. In order to limit the computations that have to be performed in later stages at other peers, when clusters from more than one peer have to be considered, the clustering should result in at most N_C^0 such basic clusters (N_C^0 is controlled by the clustering algorithm). The result of this process is illustrated in Fig. 3 (note that a peer can belong to more than one cluster). Each cluster C_i is described by a cluster description (CD), which consists of the cluster identifier C_i, a feature vector F_i, the set of peers $\{P\}$ belonging to the cluster, and the representative R of the cluster (the purposes of a representative peer is in some sense similar to a super-peer, and is among other things used to represent a cluster at search time), i.e., $\text{CD}_i = (C_i, F_i, \{P\}, R)$. For example, the CD of cluster C_2 in Fig. 3 (assuming A_7 is the cluster representative) would be $\text{CD}_2 = (C_2, F_2, \{A_5, A_7, A_8, A_9\}, A_7)$.

At this point, each initiator has identified the clusters in its zone. These clusters can be employed to reduce the cost and increase the quality of answers to queries involving the peers in one zone. However, in many cases peers in other zones are able to provide more relevant responses to queries. Thus, we need to create an overlay

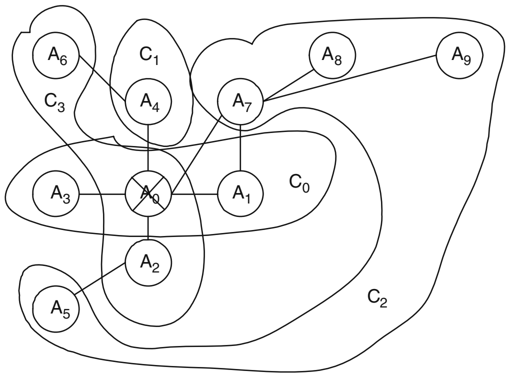


Fig. 3 Intra-zone clustering in zone A resulting in four clusters C_0, C_1, C_2, and C_3

that can help in routing queries to clusters in remote zones. In order to achieve this, we recursively apply merging of zones to larger and larger super-zones, and at the same time merge clusters that are sufficiently similar into super-clusters: first a set of neighboring zones are combined to a super-zone, then neighboring super-zones are combined to a larger super-zone, etc. The result is illustrated in Fig. 4 as a hierarchy of zones and initiators. Note that level-i initiators are a subset of the level-$(i-1)$ initiators.

This creation of the inter-zone cluster overlay is performed as outlined in Algorithm 17, which is based on the overlay topology from the previous level of zone creation: since each initiator maintains knowledge about its neighboring zones (and their initiators), the zones essentially form a zone-to-zone network resembling the P2P network that was the starting point. In general, a level-i zone consists of a number of neighboring level-$(i-1)$ zones, on average $|SZ|$ in each (where SZ denotes a set of zones, and $|SZ|$ the number of zones in the set). This implies that $\frac{1}{|SZ|}$ of the level-$(i-1)$ initiators should be level-i initiators. This is achieved by using the same technique for initiator selection as described for the first level of zones, except that in this case only peers already chosen to be initiators at level-$(i-1)$ in the previous phase are eligible for this role.

Algorithm 17 runs iteratively (creating a new level at each iteration) until only one initiator is left, i.e., when an initiator has no neighbors. The only purpose of the final initiator that is produced by performing the algorithm is to decide the level of the final hierarchy, so it does not perform any clustering operations. The aim is to have in the end at the top level a number of initiators that is large enough to provide load-balancing and resilience to failures, but at the same time low enough to keep the cost of exchanging clustering information between them during the overlay creation to a manageable level. Thus, after Algorithm 17 has terminated, the top-level peer probes level-wise down the tree in order to find the number of peers at each level until it reaches level j with appropriate number min_F of peers. The level-j

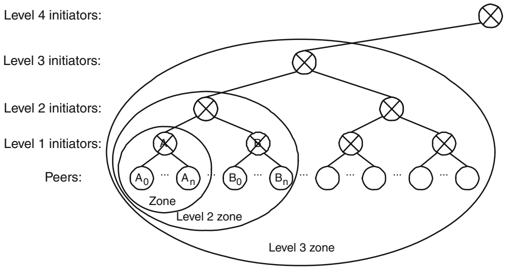


Fig. 4 Hierarchy of zones and initiators

Algorithm 17: Global clustering

1. The level-i initiators create super-zones based on previous-level zones, and in this way also the level-i initiators become aware of their neighboring super-zones.
2. In a similar way to how feature vectors were collected during the basic clustering, the approximately $N_C|SZ|$ CDs created at the previous level are collected by the level-i initiator (where N_C denotes the number of clusters per initiator at the previous level).
3. Clustering is performed again and a set of super-clusters is generated.
4. Each of the newly formed super-clusters is represented by top-k features produced by merging the top-k feature vectors of the individual clusters.
5. A peer inside the super-cluster (not necessarily one of the representatives of the cluster) is chosen as representative for the super-cluster, resulting in a new set of CDs, CD$_i = (C_i, F_i, \{P\}, R)$, where the set of peers {P} contains the representatives of the clusters forming the base of the new super-cluster.
6. The CDs are communicated to the appropriate representatives.
7. The representatives of the merged clusters (the peers in $\{P\}$ in the new CDs) are informed about the merging by the super-cluster representative.

initiators are then informed about the decision and they are given the identifiers of the other initiators at that level, in order to send their CDs to them. Finally, all level-j initiators have knowledge about the clusters in zones covered by the other level-j initiators.

To summarize, the result of the zone- and cluster-creation process is two hierarchies: (1) a hierarchy of peers and (2) a hierarchy of clusters. Starting with individual peers at the bottom level, forming zones around the initiating peer which acts as a zone controller, neighboring zones recursively form super-zones (see Fig. 4). On the top level are peers that effectively form a forest of trees, and where each peer has replicated the cluster information of the other initiators at that level.

Each peer is member of one or more clusters at the bottom level, and each cluster has one of its peers as representative. One or more clusters constitute a super-cluster, which again recursively form new super-clusters. At the top level a number of global clusters exist.

4 Searching in Semantic Overlay Networks

After semantic overlay networks have been successfully generated, the remaining issue is how to exploit them at query time, in order to improve the performance of searching. There exist two challenges in this context: (1) intra-SON search and (2) inter-SON search.

The former is about the way search is performed at peers that belong to a specific SON. For example, when a peer that belongs to a SON about rock music issues a query about "Rolling Stones", the query can be answered from this SON. However, the challenge is to find the appropriate peers to contact inside the SON. The latter refers to searches that cannot be served by the SON(s) that the querying peer belongs to. For instance, in the previous example, the peer issues a query about "Shakespeare", which clearly has to be forwarded to the SON responsible for literature. In this case, the problem is how to route the query to a particular SON, when a potential big list of SONs is available.

In the following, the most popular approach for SON-based search is presented. It consists of exploiting the links between peers with similar content for intra-SON search, and using links between peers with different content for inter-SON query routing. Unfortunately, this approach has several shortcomings, which motivate the need for a more efficient organization of SONs for searching. In this spirit, we demonstrate a super-peer architecture with a super-peer being responsible for a SON. The advantages of this approach are more efficient search relying in super-peer query routing mechanism.

4.1 Traditional SON-Based Search

Most approaches that rely on semantic overlay networks generate two types of links between peers, in order to enable subsequent search. The first type of links is *short* links, which are used to connect peers with similar content and they are used inside a SON. The second type of links is *long* links and they enable search between different SONs. Representative approaches belonging to this category include [18, 23, 35].

Searching is then performed by routing queries over the appropriate links. For search within SONs short links are used, whereas search in different SONs is performed using long links. Usually, searching over short or long links takes the form of flooding, using a TTL value to limit the search cost.

Although the performance of search is increased compared to the basic search mechanism, one problem of this approach is the lack of a selective routing mechanism over the new links. In the case of large P2P networks, this can lead to increased searching costs.

Another problem arises when advanced query processing needs to be supported, instead of file sharing. Usually, in P2P file sharing, replicas of files exist in many peers in the network, and a request for a file can be satisfied by any one of these peers. In contrast, when the query is more complex, as for example finding the sum

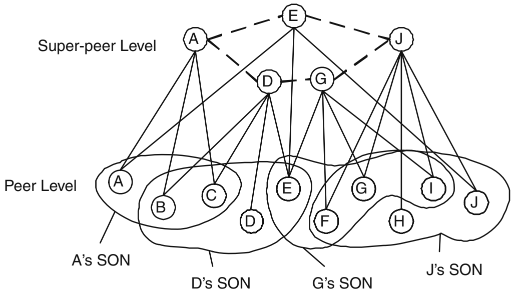


Fig. 5 SONs organized in a super-peer architecture

or average of data stored in peers, the query results will be exact only when all peers holding relevant data are contacted. As a consequence, there is a need for a more efficient mechanism that enables advanced query processing, by facilitating access to all peers that maintain data that contribute to the final result set.

4.2 Super-Peer Organization

As an improvement to the aforementioned problem, peers belonging to a SON are assigned to a super-peer. Thus, SONs are organized in a super-peer architecture, with each super-peer responsible for a SON. For an illustrative example see Fig. 5. While the basic ideas are shortly presented here, the interested reader can find more details in [13].

Intra-SON search is performed by a peer contacting the super-peer responsible for the SON, and then the super-peer forwards the query to peers in the SON. This is performed in an efficient way, as it requires the minimum number of messages for query forwarding.

Inter-SON search is based on query routing at super-peer level. In the case that super-peers are organized in a particular super-peer topology, such as a hypercube (cf. [41]), super-peer routing is efficiently performed using the topology links. In the absence of a specific topology, there exists a naive search technique of flooding the super-peer network. However, even though the number of super-peers is typically at least an order of magnitude smaller than the number of participating peers, such a technique is not scalable. Therefore, a more efficient mechanism is to create routing indices [8] at super-peer level, in order to guide the search to appropriate super-peers.

As super-peer infrastructures [53] harness the merits of both centralized and distributed architectures, we make a design choice of a super-peer architecture for the P2P network interconnection. Moreover, the choice of the super-peer architecture is motivated and driven by our main requirement for scalability. Currently deployed super-peer networks, e.g., eMule (`www.emule-project.net`) and

KaZaA (`www.kazaa.com`), support large number of users, demonstrating the appropriateness of super-peer architectures when scalability is required. In addition, peer autonomy is respected as the actual content remains on the owner peer, and only metadata is indexed and handled by the super-peer network.

5 Applications of Semantic Overlay Networks

In this section we present two application scenarios that exploit SON usage. First, we show how P2P document retrieval can be enhanced by adopting a SON-based approach (Section 5.1). Thereafter, we present the design of a distributed search engine for multimedia content that also capitalizes on SONs (Section 5.2).

5.1 P2P Web Search – Document Retrieval

The advent of the World Wide Web in combination with efficient search engines like Google and Yahoo! has made an enormous amount of information easily available for everybody. However, lately, several researchers have investigated the feasibility of providing web search facilities over a network of cooperative peers [4, 11, 22, 32, 46], contrary to the traditional model of centralized web indexing and searching. Some of the advantages of a completely distributed search engine are:

- Coverage and scalability: Current search engines only cover a small fraction of the documents on the Web, whereas a search engine consisting of thousands or millions of computers can scale better to achieve complete coverage.
- Freshness: Many web pages are rarely accessed by search engines, resulting in outdated search index contents,
- Avoiding information monopoly phenomena: A centralized search engine can control the flow of information, what is indexed, how it is presented to the user and – most importantly – the ranking. Censoring is also an issue as the search engine can filter out material that is considered controversial.
- Precision: one of the major concerns of current search engines is the precision of retrieval. A P2P search engine must adopt mechanisms that improve the precision of searching.
- Low cost: the cost of providing web search facilities employing the P2P paradigm is extremely low, as expensive data-centers and server farms are replaced by the commodity PCs of end-users.

A completely distributed search engine in its basic form consists of several individual peers that cooperate in order to provide the service collectively. Towards this goal, each peer uses part of its resources in order to build the necessary infrastructure for supporting web search. Therefore each peer:

- Collects the available web content, similar to a typical web search engine crawler.
- Indexes the assembled data, similar to inverted indexes.
- Publishes its index to the rest of the network, in order to make its data retrievable.
- Issues keyword-based queries for web document retrieval.

In the following, an application called SOWES[1] [11, 13] is presented, that supports efficient full-text search over an unstructured P2P network, by organizing peers into semantic overlay networks.

SOWES assumes a Gnutella-like unstructured P2P network of N_P peers and each peer stores a (potentially large) set of web documents. The documents stored at a peer may either belong to the peer (in case it is a web site) or may have been crawled from other sites on the web.

We now first describe how peers are organized into SONs, and then we present the searching mechanism for performing keyword-based queries.

5.1.1 SON Creation

Creation of semantic overlays in SOWES is a multi-phase distributed process, which capitalizes on DESENT (see Section 3). First zones of peers are created, then within each zone, SONs are created based on the data of peers. In the subsequent steps zones and clusters are merged recursively until global clusters are obtained. The main difference from DESENT is that the peer hierarchy is only used for SON creation and not for searching. When global SONs have been created, special purpose links are established between peers in SONs, in order to enable searching. Thus, the hierarchy of zones is only used during the SON creation process and finally the search is performed over a flat organization of SONs. In the end, the result is a set of SONs, each represented by a peer which functions similar to a super-peer.

We now describe in more detail the structure of SONs in SOWES; with particular emphasis on SONs interconnections and information retrieval aspects of SOWES. One important point is that when two individual SONs are merged into a new SON, the SON hierarchy is not maintained as in DESENT. Instead, links are created between peers in the two individual SONs ensuring sufficient connectivity, and then only the merged SON is used for further merging.

Similar to DESENT, each SON is a group of peers that is described by a cluster C_i of the peers' contents. Clusters are represented by cluster descriptors (CDs), one for each cluster C_i. In addition to the cluster identifier and feature vector, a random selection of N_r representative peers $\{R\}$ belonging to the cluster is part of the cluster description. The value of N_r is chosen high enough to minimize the probability of all of them disappearing, due to peer failures, during the lifetime of the CD, but low enough to avoid high communication cost when communicating the CDs.

[1] **S**emantic **O**verlays for **We**b **S**earching.

As an example, considering the peers in Fig. 3, the CD of cluster C_2 would be $\mathrm{CD}_2 = (C_2, F_2, \{A_5, A_8\})$, assuming A_5 and A_8 are the randomly chosen peers.

In order to ensure the connectivity of the merged SONs, d links are used between the merged SONs. In this way, the probability that a SON becomes disconnected due to peer failure is eliminated. The d links between the peers of the two clusters are formed by iteratively selecting from each SON, the least connected peers, and then connecting them. The algorithm ensures that there exists a path from each peer to any other peer within each SON.

As described so far, the resulting SONs are sufficiently connected internally. However, in order to support inter-cluster queries, the resulting SONs also need to be sufficiently connected. This is performed during the last step, when there is one zone and it is achieved by creating d connections from each representative peer to representatives of other SONs. This ensures connectivity at cluster representative level and consequently enables query routing among SONs.

Before describing how query routing is performed, we briefly review the connectivity of each peer Q:

- Q maintains L_d connections to other peers that belong in the same global SON, for each global SON that Q belongs to. These connections were created during the SON merging phase.
- Q maintains L_p connections to cluster representatives, for each global SON the peer belongs to.

In addition, cluster representatives of the global SONs maintain L_g connections to other cluster representatives. These connections are used to ensure inter-SON communication, in order to make any SON reachable from any peer and at the same time avoid its potential isolation from the rest of the SONs.

5.1.2 Keyword-Based Search

In this section, we describe how keyword-based search is performed. We first describe in more detail the basic approach to searching, followed by techniques that reduce the cost of query routing. A query for web documents originates from a querying peer Q_P. In an unstructured P2P system, querying is performed by routing the query to appropriate peers and performing the query on each of these peers, and then returning matching results. In our context, processing the query is performed by first determining which clusters might contain relevant data (*inter-cluster routing*), followed by searching one or more of these clusters (*intra-cluster routing*).

Inter-cluster routing refers to query routing at cluster representative level, which aims to identify similar cluster descriptions to the query. In order to limit the amount of costly intra-cluster searching, the inter-cluster routing is performed in two steps.

In the first step, a search for appropriate clusters is performed. A number of techniques can be used to find these clusters. Potential solutions include *random* *jumps* or some *gossiping approach* that allow peers to become familiar with a small set of peers outside their cluster. However, both these techniques and their variants impose a query horizon, i.e., they cannot guarantee that remote peers are reached in very large networks. In order to overcome such limitations, we use the links created among cluster representatives to route queries. In the absence of more sophisticated mechanisms, this routing takes the form of flooding. In this way, we can guarantee that the query contacts all clusters (i.e., their cluster representatives), thus enabling access even to the most distanced peers. Those cluster representatives reached in this way having a similarity to the query that is larger than a certain threshold, return their CD to the originating peer Q_P.

In the second step, Q_P determines which clusters are most appropriate (based on the results of step 1), and forwards the query to these for intra-cluster search ing. In a real system the number of clusters to search is determined by the number of results returned, so that the number of clusters searched can be limited.

Intra-cluster routing refers to query routing within a cluster. Routing always starts from a cluster representative (which has been found during inter-cluster routing as described above), and it is performed as flooding starting from the representative through its L_d connections. Each peer that received the query executes it locally and if it has matching results these are returned to Q_P.

5.2 P2P Image Retrieval

The widespread use of digital image equipment enables end-users to capture and edit their own image content, sometimes of high intellectual or commercial value. The centralized character of Web search raises issues regarding royalties, in the case of protected content, censorship, and to some degree information monopoly. Moreover current tools for image retrieval are limited in query expressiveness and lack semantic capabilities. Image content is only partially covered by web search engines, although it is evident that there is a tremendous wealth of digital images available on computers or other devices around the world. This is probably due to the fact that image content induces further complicated problems regarding the search: current centralized image search facilities do not sufficiently support metadata management, semantics, advanced media querying etc.

In this section, we present a SON-based architecture that adopts the P2P paradigm for indexing, searching and ranking of image content. The ultimate goal of our architecture is to provide a search mechanism for image content relying on image features. The overall system is described in [51] and the algorithms for query processing are introduced in [14]. The application described in this section is a scalable, decentralized and distributed infrastructure for building a search engine for image content capitalizing on P2P technology.

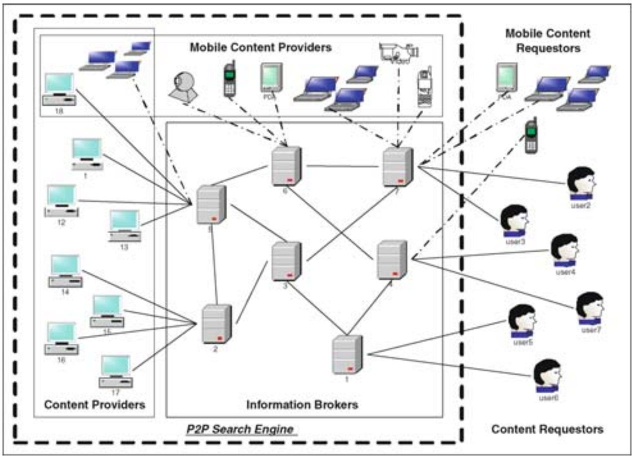


Fig. 6 Architecture of P2P search engine for image content

5.2.1 Architecture

The proposed architecture utilizes a P2P infrastructure for supporting the deployment of a scalable search engine for multimedia content, addressing future user needs and search requirements. A high-level abstraction of the architecture, showing the different participating entities, their interconnections and functionality is shown in Fig. 6.

Collaborating entities in the search engine consist of: (1) *content providers and requestors* and (2) *information brokers*.

Content providers are entities that produce or store images that they would like to publish. *Content requestors* are not necessarily contributing content to the search engine. However their role is important, as they represent the users of the search engine. These roles are not mutually exclusive. Content requestors enjoy a rich repertoire of query and searching facilities and they are provided access to thousands or millions of independent and undiscovered multimedia sources (content providers). Besides stationary content providers, mobile content providers which are able to dynamically capture content, also like to exchange image data and make it widely available.

Information brokers consist of more powerful and less volatile peers in the network. They realize a decentralized indexing service. In addition to the basic form of metadata that is generated by the content providers, information brokers may employ more sophisticated (and thus demanding) algorithms that could not be executed in lightweight peers, due to lack or processing power or lack of more widespread knowledge on the rest of the multimedia content in the network.

Obviously, the afore-described architecture can be realized by means of a superpeer network. Content providers are peers that want to make their content searchable and also be able to search other peers' content, acting as content requestors. On the other hand, information brokers are super-peers that form the backbone of the search engine. In what follows, we describe how SONs can improve the plain super-peer architecture, in terms of search performance.

5.2.2 Peer Organization

One of the most important factors that influences the performance of P2P systems is the underlying data distribution to peers (and super-peers). In the general case, when a peer joins the network, it connects to a super-peer, usually in a random way. However, data on peers is usually clustered into a few thematic areas that reflect the user's interests. It is therefore necessary to detect peers with similar content that belong to the same community. Additionally, this should occur in a self-organizing way, without explicit external intervention. Then, queries can be directed to specific peers only, thus improving query processing performance.

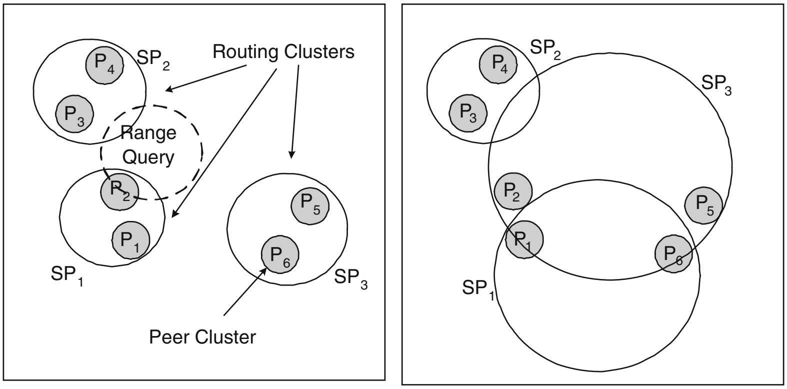
Semantic overlay networks can improve the basic architecture of the distributed search engine for image content. SONs are created in order to form communities that store similar content. After SONs have been formed, a super-peer become responsible for one or more SONs. This is achieved by reassigning peers to super-peers, in such a way that the assignment corresponds to SON membership. The result is that a super-peer indexes peers that store similar content. Consequently, queries can be answered by being directed to a limited set of super-peers, which are responsible for content relevant to the query.

5.2.3 Similarity Search

The SON-based super-peer architecture is a good starting point for constructing an efficient search mechanism. The aim is to support queries like "retrieve all images similar to a given an image" or "retrieve the *k* most similar images to a given one".

Similarity search over image content usually involves having images represented by objects in a high dimensional data space. Features of this data space may include both text-based features (such as key words, annotations) and visual features (such as color, texture, shape, faces). This feature extraction is a semi-automatic process and it takes place on each peer, before joining the search engine. Then similarity search is performed by computing the similarity or distance of high dimensional objects, provided that there exists a commonly accepted similarity or distance function.

SIMPEER [14] is a super-peer system that supports similarity search over high-dimensional data. This system can realize the search mechanism required on top of the architecture described above. After presenting its functionality in short, we show the difference in query processing performance induced by a SON-mechanism.



(a) SON-based routing clusters.

(b) Random routing clusters.

Fig. 7 Super-peer routing clusters and range query

SIMPEER relies on a three-level clustering scheme and supports efficient P2P similarity search in metric spaces. Given a super-peer network, each peer connects to a super-peer and maintains its own data, represented in a high dimensional space. In a construction phase, each peer applies a clustering algorithm on its local data. Thereafter, each super-peer gathers the clusters of its associated peers and applies on them a clustering algorithm resulting in a new cluster set that describe the data indexed by this super-peer. These clusters are broadcasted at the super-peer network, in order to form *routing clusters* at super-peers.

Figure 7a depicts the routing information stored at each super-peer, namely the routing clusters of all other super-peers. The cluster information of peers is stored only at the corresponding super-peer of each peer. At query time, each super-peer decides where to forward a query, based on its routing clusters. Assume a range query, initiated at a super-peer SP_q. First, SP_q examines its routing clusters to find to which of its neighboring super-peers the query should be forwarded to. Each recipient super-peer SP_r checks whether its local peers can provide any results, by inspecting their clusters. If the query overlaps with some clusters, SP_r contacts only the peers responsible for these clusters. Otherwise, SP_r simply forwards the query to its neighboring super-peers, using its routing clusters.

Figure 7 depicts the effects of SONs for our search engine using SIMPEER. In Fig. 7a the peer clusters assigned to a routing cluster are similar, leading to high intra-SON similarity. The inter-SON similarity, i.e., the overlap between routing clusters of different super-peers, is low. In this case SIMPEER has been shown to work well, meaning that only few super-peers and peers that can provide relevant results are queried, resulting in reduced network traffic and response time. However, for the case depicted in Fig. 7b, each peer cluster is assigned randomly to a super-peer, resulting in routing clusters that are not well-separated. Thus, the routing ability of the routing clusters is reduced, while the network traffic and the number of contacted super-peers are increased.

6 Related Work

Semantic Overlay Networks (SONs), have been proposed as an approach for semantically organizing peers, so that queries can be forwarded to only those peers containing documents within specific topics. In the seminal paper [9], SONs are presented as thematic focused groups of peers, which share common interests. In this respect, a P2P network is organized into SONs, in order to enable efficient routing of queries only to relevant peers. The advantage of this approach is that it reduces the routing cost, namely it reduces the flooding cost in the case of unstructured P2P networks and decreases the number of peers that need to be contacted in the case of structured P2P systems. In principle, SONs also enhance the quality of results, an important issue in the case of distributed information retrieval tasks like P2P web search.

In the following, we review existing algorithms and techniques for semantic overlay network generation in P2P systems. A taxonomy of semantic overlay networks is presented in Table 1, by first distinguishing them based on the underlying P2P architecture, namely *unstructured* and *structured*. Then, within each architecture, three basic methods for SON generation are identified: *clustering*, *classification* and *gossip-based*.

	Unstructured	Structured
Clustering	DESENT [12] Distributed K-means [15] Topic-segmented overlays [3] SSW [23] Super-peers [13] Klampanos et al. [19] Semantic peer [26]	pSearch [47] WonGoo [31] SWOP [18]
Classification	Content-based overlays [27] Fairness index [49] iClusterDL [37] iCluster [35]	HSPIR [25] Content-based overlays [27]
Gossiping	REMINDIN' [48] Associative overlays [7] Shortcuts [44] INGA [29] Acquaintances [5] Metric social networks [42] SON reformulation [20] Epidemic protocol [52] Chatty Web [1] Language models [24]	p2pDating [34] GridVine [2]

Table 1 Taxonomy of semantic overlay network generation algorithms

6.1 Unstructured P2P

6.1.1 Clustering

Although several papers describe how to use SON-like structures, little work exists on the issue of to actually create SONs in an unsupervised, decentralized and distributed way in unstructured networks. One important difficulty arises when there is a lack of background knowledge about peer contents. In the following, SON construction methods that use *unsupervised learning* are presented, with most of these approaches relying on *clustering algorithms*.

A distributed (but not decentralized) P2P clustering algorithm is presented in [15], so that for a given query, the querying peer can identify the most promising clusters and route the query precisely to peers containing relevant documents.

In [3], *topic-segmented overlays* are proposed, which partition peers into groups of similar topics. The approach is based on assembling peer representations at one site and perform a clustering that is then communicated to all peers. Cluster updates are treated similarly; the site determines any changes of the global cluster descriptions and informs other peers on demand.

Klampanos et al. [19] propose to cluster peer contents, in order to generate a cluster-based architecture for P2P information retrieval. Individual peer documents are clustered using a hierarchical clustering algorithm and document clusters are evaluated using two metrics: (1) the average standard deviation of term frequency among the documents of each cluster and (2) the number of peer documents within a specific topic, which shows the expertise of a peer. Then a one-pass peer clustering algorithm is used to identify *content-aware groups*, i.e., peer clusters. However this algorithm is not distributed, assuming all knowledge of peer clusters is available at one location.

Semantic small world (SSW) [23] aims at small world network construction by semantic clustering of peer contents. Each peer is represented by the centroid of its largest cluster in the semantic space. This forms the peer's semantic label. Furthermore, each peer maintains short and long range contacts, where short range contacts refer to neighboring peer clusters, while long range contacts refer to peers with data at randomly chosen points in the semantic space.

In [13], SONs are generated using hierarchical clustering of peer contents. After SON construction, each SON is assigned to a super-peer, resulting in a super-peer organization of SONs. This leads to performance improvements, both in terms of retrieval quality and associated search costs.

Recently, a measure for cluster cohesion, called *clustering efficiency*, has been proposed in [36], as a way to evaluate SON quality. In contrast to existing measures, clustering efficiency takes into account the neighborhood of a peer, not only its direct neighbors. Maintenance of neighboring links is based on a periodic rewiring procedure, which triggers the process of discovering similar peers.

Another approach to improve on some of the problems of Gnutella-like systems, is to use a super-peer architecture where a number of peers/clients are connected to a super-peer, which (1) indexes the data stored on its peers, as well as

(2) communicates with other super-peers so that queries that can not be satisfied from the local super-peer can be forwarded to other super-peers. Super-peer approaches benefit from the advantages of both the centralized and the distributed paradigm. They combine the efficiency of centralized search with the intrinsic features of P2P. An interesting study of super-peer networks is presented by Yang and Garcia-Molina [53]. The authors discuss design issues concerning super-peer networks and come up with a set of useful rules of thumb. Edutella [33] is another super-peer approach, based on a hypercube topology [41], for routing queries. Super-peers maintain indices of peer contents and routing indices, and searching is practically achieved through routing at super-peer level. A super-peer architecture can also be used to realize a hierarchical summary index as described in [43]. In [28], clustering policies are proposed to generate semantic clusters in super-peer networks. Particular emphasis is put on managing heterogeneous data schemes.

6.1.2 Classification

One of the problems of unsupervised techniques is that it is generally difficult to identify coherent clusters, without exploiting any background knowledge. As a solution to this, several papers have advocated the use of background knowledge, in order to generate groups of peers with similar content. These methods belong to the category of *supervised learning* and they usually employ *classification algorithms* or assume the existence of a common ontology among peers.

In [49], a P2P architecture where nodes are logically organized into a fixed number of clusters is presented. The main focus is on fairness with respect to the load of individual nodes. For this reason, the *fairness index* is proposed, as a metric for inter-cluster load balancing. Furthermore, intra-cluster load balancing is considered, in order to achieve load balancing among peers that belong to the same cluster.

In [35], the authors present *iCluster*, a system where peers become members of SONs by creating and maintaining two types of links: *short-range* and *long-range* links. Short range links connect peers inside a SON, while long range links are used to interconnect SONs and assist searching. The actual SON creation is based on a periodic rewiring procedure, with each peer trying find other similar peers. An application of this approach in a digital library context is the *iClusterDL* system [37]. Notice that according to the authors SONs can be generated either through document clustering or classification, however we choose to record this work as classification mainly due to the way experiments were performed.

In [26], the authors address the issue of SON creation in a Peer Data Management System (PDMS) where peers have different schemas and they are connected through schema mappings. Each peer is represented by a set of concepts. Each concept will be associated to at most one SON. Heterogeneity is solved using the WordNet as background thesaurus. The authors define a distance function among sets of concepts and they use the BUBBLE framework [16] for clustering. At peer join time, neighbor selection can be performed in two ways: range selection (select the peers in the SON with distance below a threshold) and k-NN selection (the k closest peers semantically). Especially the k-NN approach, requires setting a value for the radius, which requires some non-negligible communication and synchronization with the joining peer. Updates seem to flood the network and their propagation does not have a stopping mechanism.

6.1.3 Gossip-Based

Several papers related to SON construction rely on gossiping protocols for distributing information in the P2P network. Once information about peers' contents is spread in the network, a peer can establish links to relevant peers. The advantage of gossip-based approaches is that they are completely distributed and support self-organization. However, one of their disadvantages is that they can provide no guarantees that relevant peers will be acquainted in finite time, especially in the case of really large and dynamic P2P networks. Obviously, depending on the application, gossip-based approaches can provide an efficient solution for SON generation and subsequent searching.

Associative overlays, proposed in [7], are similar to SONs. The underlying concept is that peers which have previously satisfied queries are more likely candidates for answering a current query. Hence, such peers are organized into overlay networks. The approach has advantages over blind search in unstructured P2P networks, especially for searches for rare items.

A similar notion has been recently proposed. In *Metric Social Networks* [42], peers establish links based on query histories, thus forming overlay networks for improving the efficiency of similarity search. The main idea is to exploit the concept of social networks, with peers creating relationships to other peers according to the similarity of their data. Peer links are generated based on query results, trying to identify peers that contributed significantly to the results of a query.

Cholvi et al. [5] propose the use of *acquaintances* as an extension to Gnutella-like networks to improve searching. Peers with similar contents are linked together, so that searches for a particular topic can be routed to more relevant peers in less time. Semantic communities of peers are created in this way. The basic intuition of this approach is that efficient searching is achieved because requests that initiate from a particular community can be fulfilled within the community with high probability.

In [44], peers that share similar interests create *shortcuts* to each other, thus enriching the original overlay connections. The use of interest-based shortcuts enhances the basic search mechanism employed by Gnutella. Essentially, the creation of interest-based groups of peers is quite similar to the concept of semantic overlay networks.

The use of shortcuts has also been employed for generating semantic social overlay networks [29]. In the system called INGA, peers create and maintain shortcuts to other peers in a lazy manner, by examining past queries both issued by the peer itself and routed through the peer. Shortcuts are maintained at four different level serving different purposes: (1) at the content provider layer, shortcuts are created to remote peers which have successfully answered a query, (2) at the recommender layer, information is maintained about remote peers who have issued a query, (3) at the bootstrapping layer, shortcuts to well connected remote peers are kept, and (4) at the network layer, connections are maintained to peers of the underlying network topology. The authors study the comparative performance of INGA to the approach presented in [44] and their results show that INGA performs better.

Yet another approach where peers collect and maintain information about other peers that answered successfully past queries is presented in [48]. The authors study query evaluation of RDF data distributed over data repositories. Similarly to previous works, the objective of keeping information of successful past query evaluation is to enable future peer selection. This approach has been later extended, in order to support ranking of peers [30].

Aberer et al. [1] present a similar gossip-based technique for creation of semantic links among peers. The main focus of the paper is on producing global semantics from local interactions among peers.

In [24], statistical language models on peers are used to describe peer contents. Then SONs are created by having peers periodically exchange information about their nearest neighbors in the semantic space. Through random meetings with other peers, each peer maintains locally a set of references about interesting peers, which will be used at query time.

Voulgaris et al. [52] propose an epidemic protocol for SON construction. This is essentially a gossiping approach that enables peers to become familiar with other semantically related peers. In this way peer clustering is performed using a semantic proximity function that quantifies the semantic similarity of peers.

In [20], the authors study the problem of keeping SONs updated, without having to reconstruct the overlay network. The approach is based on having a cluster representative for each cluster, which assembles relocation requests and serves them in a second phase. The cluster reformulation problem is modeled in a novel way as a game, where peers decide which clusters to join based on potential gains in the recall of their future queries.

6.2 Structured P2P

6.2.1 Clustering

In the context of structured P2P systems, several research papers have dealt with the issue of improving the basic search provided by the underlying DHT. In general, two are the main goals that should be achieved. First, in document retrieval applications, support for semantic retrieval is required, in contrast to existing exact matching of keywords to documents. Semantic overlay networks are useful in this direction, as peers are organized based on the semantics of their content. Second, improvements to the performance of the underlying structured P2P protocol can be achieved, using SON-based techniques. The objective is to reduce the search cost and the increase the precision of the search, by contacting only carefully selected peers.

Tang et al. [47] propose pSearch, a system for SON creation on top of structured P2P networks. The approach is based on the use of LSI to compute the (multidimensional) semantic key that can provide the coordinates for a document in a CAN network, and then use a nearest-neighbor search technique in order to find appropriate documents. Thus the indices of semantically related documents are assigned to the same or neighboring peers in the overlay.

Improvements to the approach are presented in WonGoo [31]. Mainly the approach tries to improve on the costly update process of pSearch. This is achieved by computing LSI of terms not documents, so as terms in a corpus are not significantly affected when the corpus changes, the LSI of terms do not need to be updated. In terms of the actual SON generation, WonGoo uses similar techniques with pSearch.

However, some of the inherent problems of LSI, like processing cost and choosing number of dimensions, make it difficult to employ this technique in a large-scale dynamic document repository. Furthermore, both [25, 47] assume a central LSI computation, which presumes that all documents or a sample of reasonable size must be assembled at a central location, which is infeasible for a large P2P system, especially when acquisition of global knowledge is impossible.

In [18], the SWOP protocol for construction of small-world P2P overlays has been proposed. Similar to other papers, peers maintain *cluster links* and *long links* to neighboring and distant nodes respectively. In this way, small-world overlay network can improve the performance of the underlying structured P2P, in this case Chord. Furthermore, exploiting the nice properties of small world network, the authors propose an effective object replication algorithm, suitable for handling dynamic query workloads for popular objects, such as the flash crowd scenario.

6.2.2 Classification

There are only few papers that use classification techniques for semantic overlay network creation over structured P2P systems.

Liu et al. [25] propose HSPIR, a hierarchical SON based on CAN [38] and Range Addressable Network [21]. Support for semantics is achieved by using Latent Semantic Indexing (LSI) [10]. A document is represented using the vector space model and LSI and it is classified using support vector machines. A query is classified in order to match a topic and then routed to the appropriate peer responsible for this topic using CAN. Then this peer forwards the query to other peers inside the SON in a hierarchical way, exploiting the range addressable network.

In [27], the authors present a system that exploits existing classifications of peers in taxonomies, in order to build overlays with peers based on taxonomy information. This enables the creation of *Content-based Overlay Networks (CONs)*, which is essentially a content-based grouping of peers. One advantage of CONs over traditional DHT-based systems is that queries that retrieve only few results can be broadened exploiting the information stored in the taxonomy.

6.2.3 Gossip-Based

A similar approach to [5, 44] has been described in [34]. Peers perform random meetings with other peers at regular intervals, thereby progressively establishing connections to *friendly* peers. An advantage of this approach is that it respects peer autonomy, as each peer may independently decide which connections to create and which to avoid. Another nice feature of this family of techniques is adaptivity to changes in the peer's underlying content and interests. The authors present this SON generation method as an improvement of P2P web search over a structured P2P infrastructure [32].

A different notion of SONs, as outlined in [2], is related to schema mappings and peers that are logically interconnected through schema mappings. The *GridVine* system is built on top of a structured P2P network, called P-Grid. Through local schema translations and semantic gossiping, global SONs are created.

6.3 Evaluation

In this section, we present a meaningful evaluation of the aforementioned approaches. The aim is to identify their comparative advantages and disadvantages based on the nice properties that are supported by each approach. A summary of this comparison is presented in Table 2.

A set of quality indices is employed, in order to perform the evaluation. A SON generation method is evaluated based on its adherence to the following properties: distributed, decentralized, unsupervised, scalable, delf-organizing, autonomy. For a more general discussion regarding the requirements for SON generation we refer to [50].

The rationale for classifying a SON generation algorithm is briefly explained in the following. An algorithm is distributed when it is executed at several sites, instead of a single one. Most SON generation algorithms fulfill this criterion, with few exceptions, for example the approach in [19] describes a peer clustering algorithm that is not distributed. Decentralization characterizes algorithms that do not require a centralized location for assembling data or processing algorithms. For instance, pSearch [47] requires the computation of LSI in a centralized manner, therefore it is not decentralized. The distinction between supervised and unsupervised approaches is easy, as it is based on the existence of background knowledge or the use of supervised learning techniques. Scalability is an important requirement and it is not always supported by SON generation methods. An approach is deemed scalable if it can scale up to hundreds of thousands of peers, as opposed to some hundreds or thousands of peers only. For such large-scale networks, it is unclear whether gossip-based approaches can produce results of acceptable quality, as they are usually evaluated at much smaller scale. Self-organization is a necessary property for P2P systems, and approaches that work in a bottom-up manner, starting at peer level and converging at some fixed point without external intervention, usually satisfy this requirement. Peer autonomy is the typical case for SON creation methods in unstructured P2P networks, however it is often violated in structured P2P systems, when peers are assigned specific parts of the global index.

		Distributed	Decentralized	Unsupervised	Scalable	Self-organizing	Autonomy
Unstructured P2P	DESENT [12]	√	√	√	√	√	√
	Distributed K-means [15]	√		√	√	√	
	SSW [23]	√	√	√	√	√	
	Klampanos et al. [19]			√	√	√	√
	Topic-segmented overlays [3]			√		√	√
	Fairness Index [49]	√	√		√	√	√
	Super-peers [13]	√	√	√	√	√	√
	iClusterDL [37]	√	√	√		√	√
	iCluster [35]	√	√	√		√	√
	REMINDIN' [48]	√	√	√		√	√
	Associative overlays [7]	√	√	√		√	√
	Shortcuts [44]	√	√	√		√	√
	INGA [29]	√	√	√		√	√
	Acquaintances [5]	√	√	√		√	√
	Metric social networks [42]	√	√	√		√	√
	SON reformulation [20]	√	√	√		√	√
	Epidemic protocol [52]	√	√	√		√	√
	Chatty Web [1]	√	√	√		√	√
	Language models [24]	√	√	√		√	√
	Semantic peer [26]	√	√	√		√	√
Structured P2P	pSearch [47]	√			√	√	
	HSPIR [25]	√			√	√	
	WonGoo [31]	√			√	√	
	SWOP [18]	√	√	√	√	√	
	p2pDating [34]	√	√	√		√	√
	Content-based overlays [27]	√	√		√		
	GridVine [2]	√	√		√		

Table 2 Evaluation of existing semantic overlay network techniques

7 Summary and Future Trends

In this chapter, distributed semantic overlay networks for peer-to-peer systems have been presented. Having described the requirements for overlay construction, a semantic overlay network generation algorithm with salient features has been presented. Furthermore, searching over semantic overlay networks has been discussed, demonstrating the advantages in terms of improved performance. A number of applications can be deployed exploiting semantic overlay networks and two such examples have been presented, namely P2P web search and P2P image retrieval. Finally, a comprehensive overview of the research related to semantic overlay networks has been conducted.

Future research related to semantic overlay networks looks promising. One research area that has attracted interest lately is social networks. Semantic overlay networks can help to identify social communities in the case that such explicit information is not available. In addition, more advanced tasks like social network mining can be performed using techniques and experiences from semantic overlays.

Another interesting research direction is peer ranking in the context of semantic overlays. Most approaches so far consider peers inside a SON as equal, however there exist several applications that could exploit ranking such peers according to specific criteria. For example, in the case that an application needs to return a few results fast to the user, contacting the highest ranked peers instead of all, would result in savings in response time. Such an application is top-k retrieval, which is quite common in today's web search.

Personalization is an interesting topic that can be studied in the context of semantic overlay networks. In scenarios where users have different preferences or when objective criteria either do not exist or are not helpful, it is not possible to establish commonly accepted semantic overlays. Thus it is interesting to study semantic overlay generation and maintenance in uncertain environments.

In spite of the rich existing work in the field of SONs, several challenges have not been efficiently addressed yet, therefore, distributed SON management is deemed as a future promising research direction.

References

1. Aberer, K., Cudré-Mauroux, P., Hauswirth, M.: The chatty web: emergent semantics through gossiping. In: Procedings of 12th International Conference on World Wide Web (WWW), pp. 197–206 (2003)
2. Aberer, K., Cudré-Mauroux, P., Hauswirth, M., Pelt, T.V.: GridVine: Building internet-scale semantic overlay networks. In: Proceedings of International Conference on Semantic Web (ISWC), pp. 107–121
3. Bawa, M., Manku, G.S., Raghavan, P.: SETS: Search enhanced by topic segmentation. In: Proceedings of the 26th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR), pp. 306–313 (2003)
4. Bender, M., Michel, S., Triantafillou, P., Weikum, G.: Global document frequency estimation in peer-to-peer web search. In: Proceedings of 9th International Workshop on the Web and Databases (WebDB), pp. 62–67 (2006)
5. Cholvi, V., Felber, P., Biersack, E.: Efficient search in unstructured peer-to-peer networks. Tech. rep., Institut EURECOM (2003)
6. Clarke, I., Sandberg, O., Wiley, B., Hong, T.: Freenet: A distributed anonymous information storage and retrieval system. In: Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability, pp. 46–66 (2000)

7. Cohen, E., Fiat, A., Kaplan, H.: Associative search in peer-to-peer networks: Harnessing latent semantics. In: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) (2003)
8. Crespo, A., Garcia-Molina, H.: Routing indices for peer-to-peer systems. In: Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS), pp. 23–34 (2002)
9. Crespo, A., Garcia-Molina, H.: Semantic overlay networks for P2P systems. In: Proceedings of the 3rd International Workshop on Agents and Peer-to-Peer Computing (AP2PC), pp. 1–13 (2004)
10. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. Journal of the American Society of Information Science **41**(6), 391–407 (1990)
11. Doulkeridis, C., Nørvåg, K., Vazirgiannis, M.: The SOWES approach to P2P web search using semantic overlays. In: Proceedings of 15th International Conference on World Wide Web(WWW), pp. 1027–1028 (2006)
12. Doulkeridis, C., Nørvåg, K., Vazirgiannis, M.: DESENT: Decentralized and distributed semantic overlay generation in P2P networks. IEEE Journal on Selected Areas in Communications (J-SAC) **25**(1), 25–34 (2007)
13. Doulkeridis, C., Nørvåg, K., Vazirgiannis, M.: Peer-to-peer similarity search over widely distributed document collections. In: Proceedings of the 6th Workshop on Large-Scale Distributed Systems for Information Retrieval (LSDS-IR) (2008)
14. Doulkeridis, C., Vlachou, A., Kotidis, Y., Vazirgiannis, M.: Peer-to-peer similarity search in metric spaces. In: Proceedings of the 33rd International Conference on Very Large Data Bases(VLDB), pp. 986–997 (2007)
15. Eisenhardt, M., Mueller, W., Henrich, A.: Classifying documents by distributed P2P clustering. GI Jahrestagung pp. 286–291 (2003)
16. Ganti, V., Ramakrishnan, R., Gehrke, J., Powell, A.L., French, J.C.: Clustering large datasets in arbitrary metric spaces. In: Proceedings of the 15th International Conference on Data Engineering (ICDE), pp. 502–511 (1999)
17. Gnutella. `http://en.wikipedia.org/wiki/Gnutella`
18. Hui, K.Y.K., Lui, J.C.S., Yau, D.K.Y.: Small-world overlay P2P networks: Construction, management and handling of dynamic flash crowds. Computer Networks **50**(15), 2727–2746 (2006)
19. Klampanos, I.A., Jose, J.M.: An architecture for information retrieval over semi-collaborating peer-to-peer networks. In: Proceedings of the 2004 ACM Symposium on Applied Computing (SAC), pp. 1078–1083 (2004)
20. Koloniari, G., Pitoura, E.: Recall-based cluster reformulation by selfish peers. In: ICDE Workshops, pp. 200–205 (2008)
21. Kothari, A., Agrawal, D., Gupta, A., Suri, S.: Range addressable network: A P2P cache architecture for data ranges. In: Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P), p. 14 (2003)
22. Li, J., Loo, B., J.M.Hellerstein, Kaashoek, M., Karger, D., Morris, R.: On the feasibility of peer-to-peer web indexing and search. In: Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS), pp. 207–215 (2003)
23. Li, M., Lee, W.C., Sivasubramaniam, A.: Semantic small world: An overlay network for peer-to-peer search. In: Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP), pp. 228–238 (2004)
24. Linari, A., Weikum, G.: Efficient peer-to-peer semantic overlay networks based on statistical language models. In: Proceedings of Workshop on Information Retrieval in Peer-to-Peer Networks (P2PIR) (2006)
25. Liu, F., Li, M., Huang, L.: Distributed information retrieval based on hierarchical semantic overlay network. In: Proceedings of the 3rd International Conference on Grid and Cooperative Computing (GCC), pp. 657–664 (2004)

26. Lodi, S., Penzo, W., Mandreoli, F., Martoglia, R., Sassateli, S.: Semantic peer, here are the neighbors you want! In: Proceedings of 11th International Conference on Extending Database Technology (EDBT), pp. 26–37 (2008)
27. Löser, A.: Taxonomy-based overlay networks for P2P systems. In: Proceedings of the 8th International Database Engineering and Applications Symposium (IDEAS), pp. 407–412 (2004)
28. Löser, A., Naumann, F., Siberski, W., Nejdl, W., Thaden, U.: Semantic overlay clusters within super-peer networks. In: Proceedings of the 1st International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P) (2003)
29. Löser, A., Staab, S., Tempich, C.: Semantic social overlay networks. IEEE Journal on Selected Areas in Communications **25**(1), 5–14 (2007)
30. Löser, A., Tempich, C.: On ranking peers in semantic overlay networks. In: Wissensmanagement, pp. 209–216 (2005)
31. Lv, J., Cheng, X.: WonGoo: A pure peer-to-peer full text information retrieval system based on semantic overlay networks. In: Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications(NCA), pp. 47–54 (2004)
32. Michel, S., Triantafillou, P., Weikum, G.: Minerva∞: A scalable efficient peer-to-peer search engine. In: Proceedings of 6th International Middleware Conference, pp. 60–81 (2005)
33. Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I., Löser, A.: Super-peer-based routing and clustering strategies for RDF-based P2P networks. In: Proceedings of the 12nd International Conference on World Wide Web (WWW), pp. 536–543 (2003)
34. Parreira, J.X., Michel, S., Weikum, G.: p2pDating: Real life inspired semantic overlay networks for web search. In: Proceedings of 2005 Workshop on Heterogeneous and Distributed Information Retrieval (HDIR) (2005)
35. Raftopoulou, P., Petrakis, E.G.M.: iCluster: A self-organizing overlay network for p2p information retrieval. In: Proceedings of the 30th European Conference on IR Research (ECIR), pp. 65–76 (2008)
36. Raftopoulou, P., Petrakis, E.G.M.: A measure for cluster cohesion in semantic overlay networks. In: Proceedings of the 6th Workshop on Large-Scale Distributed Systems for Information Retrieval (LSDS-IR) (2008)
37. Raftopoulou, P., Petrakis, E.G.M., Tryfonopoulos, C., Weikum, G.: Information retrieval and filtering over self-organising digital libraries. In: Proceedings of the 12th European Conference on Research and Advanced Technology for Digital Libraries (ECDL) (2008)
38. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: Proceedings of International Conference on Data Communication (SIGCOMM), pp. 161–172 (2001)
39. Ritter, J.: Why Gnutella can't scale. No, really! `http://www.darkridge.com/~jpr5/doc/gnutella.html` (2001)
40. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Proceedings of the International Middleware Conference, pp. 329–350 (2001)
41. Schlosser, M., Sintek, M., Decker, S., Nejdl, W.: HyperCuP – Hypercubes, ontologies and efficient search on P2P networks. In: Proceedings of International Workshop on Agents and Peer-to-Peer Computing (AP2PC), pp. 112–124 (2002)
42. Sedmidubský, J., Barton, S., Dohnal, V., Zezula, P.: Adaptive approximate similarity searching through metric social networks. In: Proceedings of the 24th International Conference on Data Engineering (ICDE), pp. 1424–1426 (2008)
43. Shen, H.T., Shu, Y., Yu, B.: Efficient semantic-based content search in P2P network. IEEE Transactions on Knowledge and Data Engineering **16**(7), 813–826 (2004)
44. Sripanidkulchai, K., Maggs, B., Zhang, H.: Efficient content location using interest-based locality in peer-to-peer systems. In: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) (2003)

45. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of International Conference on Data Communication (SIGCOMM), pp. 149–160 (2001)
46. Suel, T., Mathur, C., wen Wu, J., Zhang, J., Delis, A., Kharrazi, M., Long, X., Shanmugasundaram, K.: ODISSEA: A peer-to-peer architecture for scalable web search and information retrieval. In: Proceedings of 6th International Workshop on the Web and Databases (WebDB) (2003)
47. Tang, C., Xu, Z., Dwarkadas, S.: Peer-to-peer information retrieval using self-organizing semantic overlay networks. In: Proceedings of International Conference on Data Communication (SIGCOMM), pp. 175–186 (2003)
48. Tempich, C., Staab, S., Wranik, A.: REMINDIN': Semantic query routing in peer-to-peer networks based on social metaphors. In: Proceedings of the 13th International Conference on World Wide Web (WWW), pp. 640–649 (2004)
49. Triantafillou, P., Xiruhaki, C., Koubarakis, M., Ntarmos, N.: Towards high performance peer-to-peer content and resource sharing systems. In: Proceedings of the 1st Biennial Conference on Innovative Data Systems Research (CIDR) (2003)
50. Vazirgiannis, M., Nørvåg, K., Doulkeridis, C.: Peer-to-peer clustering for semantic overlay network generation. In: Proceedings of the 6th International Workshop on Pattern Recognition in Information Systems (PRIS'06) (2006)
51. Vlachou, A., Doulkeridis, C., Mavroeidis, D., Vazirgiannis, M.: Designing a peer-to-peer architecture for distributed image retrieval. In: Proceedings of the 5th International Workshop on Adaptive Multimedial Retrieval: Retrieval, User, and Semantics (AMR), pp. 182–195 (2007)
52. Voulgaris, S., van Steen, M., Iwanicki, K.: Proactive gossip-based management of semantic overlay networks. Concurrency and Computation: Practice and Experience **19**(17), 2299–2311 (2007)
53. Yang, B., Garcia-Molina, H.: Designing a super-peer network. In: Proceedings of the 19th International Conference on Data Engineering (ICDE), pp. 49–60 (2003)
54. Zhao, B., Kubiatowicz, J., Joseph, A.: Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. rep., U.C.Berkeley (2001)

Self-adaptation and Self-organization for Search in Social-Like Peer-to-Peer Networks

Lu Liu, Jie Xu, Duncan Russell, and Zongyang Luo

Abstract For resource discovery in social networks, people can directly contact some acquaintances that potentially have knowledge about the resources they are looking for. However, in current peer-to-peer (P2P) networks, each peer node lacks a "social" network, making it difficult to route queries efficiently. Similarly to social networks where people are connected by their social relationships, two peer nodes can be linked in unstructured P2P networks if users in those nodes are interested in each other's resources. In this chapter, a social model is presented for adaptive and efficient resource discovery in P2P networks by mimicking different human behaviours in social networks.

1 Introduction

As an emerging technology, the innovations of peer-to-peer (P2P) networks offer many interesting avenues of research for scientific communities. As a major design pattern for future systems opposed to the traditional client-server paradigm, research on P2P networks is extremely important, which will possibly significantly alter the

Lu Liu
School of Computing, University of Leeds, Leeds, LS2 9JT, UK,
e-mail: luliu@comp.leeds.ac.uk

Jie Xu
School of Computing, University of Leeds, Leeds, LS2 9JT, UK,
e-mail: jxu@comp.leeds.ac.uk

Duncan Russell
School of Computing, University of Leeds, Leeds, LS2 9JT, UK,
e-mail: duncanr@comp.leeds.ac.uk

Zongyang Luo
School of Computing, University of Leeds, Leeds, LS2 9JT, UK,
e-mail: scszluo@leeds.ac.uk

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_18, © Springer Science+Business Media, LLC 2010

way of day-to-day use of software systems. In the last few years, great achievements have been made on P2P resource sharing and data transfer. However, we will not reap all the benefits of utilising these resources in P2P networks unless we have an efficient way to discover them. Efficient resource discovery remains a fundamental challenge for large-scale P2P networks.

For resource discovery in social networks, people can directly contact some acquaintances that potentially have knowledge about the resources they are looking for. However, in current P2P networks, each peer node lacks a “social” network, making it difficult to route queries efficiently. Similarly to social networks where people are connected by their social relationships, two autonomous peer nodes can be connected in unstructured P2P networks if users in those nodes are interested in each other’s data. The similarity between P2P networks and social networks, where peer nodes can be considered as people and connections can be considered as relationships, leads us to believe that human tactics in social networks are useful for improving the performance of resource discovery [1] by self-organising autonomous peer nodes in unstructured P2P networks.

The theories of small world in social sciences have been used in the design of P2P systems. But most studies of constructing small world in P2P networks are based on the concept of organising peer nodes into groups or clusters (e.g., [2–5]). However, the simple community formation and discovery becomes much more complex due to the lack of a central server. A large communication overhead is required to compensate for the server for group management and resource discovery.

In contrast, social communities are formed naturally by daily intercommunications between people. No additional overhead is required for maintenance of social communities. If P2P networks are self-organised like social networks, a large communication cost for group construction, group maintenance and group discovery can be saved, which can significantly improve overall performance of P2P networks.

To address this problem, an Adaptive and Efficient Social-Like Peer-to-peer (AESLP) model for resource discovery is presented in this chapter, which demonstrates how human strategies in social networks can improve resource discovery in P2P networks, by mimicking different human behaviours in social networks. AESLP could be deployed on the top of any unstructured P2P networks to improve the performance of resource discovery. Here AESLP is discussed by building upon the Gnutella network which is a well-known unstructured P2P network.

Unlike previous models (e.g., [2–5]), we do not intentionally construct peer communities. In contrast, AESLP is a self-organising and self-adaptive P2P system, which gives peer nodes a social network, and lets them meet and connect to each other. In AESLP, peer nodes have the ability to self-organise themselves in a cooperative manner similarly to social networks. Peer nodes that have the same interests will gradually connect to each other and form peer communities spontaneously, which can significantly eliminate communication overhead of previous community-based P2P systems.

2 Background

Sripanidkulchai [6] presented a content location solution in unstructured P2P networks in which peer nodes loosely organise themselves into interest-based structure. When a peer node joins the system, it first searches the network by flooding to locate content. The lookup returns a set of peer nodes that store the content. These peer nodes are potential candidates to be added to a "shortcut list". One peer node is selected at random from the set and added. Subsequent queries will go through the peer nodes in the shortcut list. If a peer cannot find content from the list, it will generate a lookup with Gnutella protocol. From their results, a significant amount of flooding can be avoided which makes Gnutella a more competitive solution.

Sripanidkulchai's model was further extended in study [7] on SWAP infrastructure [8] by classifying the shortcuts into four layers: shortcuts to remote peers which have successfully answered a query; shortcuts to remote peers who have issued a query; shortcuts to well connected remote peers; and shortcuts to peers on an underlying default network. However, this shortcut overlay network is still constructed based on routing by flooding and maintenance overhead is potentially increased with a hierarchical architecture.

Cholvi [9] presented a topology adaptation mechanism to increase the search efficiency. Each peer node created two types of links: neighbour links which connect to a random set of peer nodes and acquaintance links which connect to a set of peer nodes chosen based on common interests, and use these lists to efficiently route queries in the network.

In [10], Bender proposed three routing methods: semantic query routing based on the content similarity between the query and the data held by target peer nodes; social query routing based on social relationships; and spiritual query routing based on behavioural affinity. Their experimental results show that social networks are able to boost search result quality in a P2P network.

Search techniques described above provide useful solutions to the problem of resource discovery in P2P networks based on social phenomena. However, none of these have explored the possibility of building a both self-adaptable and self-organising P2P system. P2P systems are highly dynamic in nature. Each peer node should have self-adaptability to modify its behaviour according to the environmental changes. In AESLP, each node is able to self-adapt searches by using different types of queries according to its own situation and also self-justify forwarding degree (number of receivers) of each query running through according to the probability of successfully finding the requested files. Moreover, different from any other methods discussed above, AESLP enables peer nodes not only to forward queries to the "good" nodes that can potentially offer the requested resources or know who has the requested resources, but also to avoid sending queries to the "bad" nodes that cannot help find any requested files. AESLP mimicks eight different human behaviours in social networks and shows how human behaviours can bring about benefit in improving the performance of resource discovery in P2P networks.

3 AESLP: Adaptive and Efficient Social-Like Peer-to-Peer

In this section, we will generate these eight different social behaviours based the social theories that can be used in the resource discovery in P2P systems and then mimick these social behaviours in a P2P network.

3.1 Knowledge Index Creation and Update

Social behaviour 1: in social networks, people remember and update potentially useful knowledge from social interactions, and then random and diffuse behaviours gradually become highly organised [11]. For example, if a man named Bob successfully borrows an Oxford English dictionary from his friend Alice, this successful interaction will be remembered by Bob. When Bob needs this dictionary again, he will preferentially seek help directly from Alice rather than other people. However, if Alice does not have the Oxford English dictionary and cannot provide any useful information to help Bob find this dictionary, Bob would avoid contacting Alice if he needs the dictionary again in the near future.

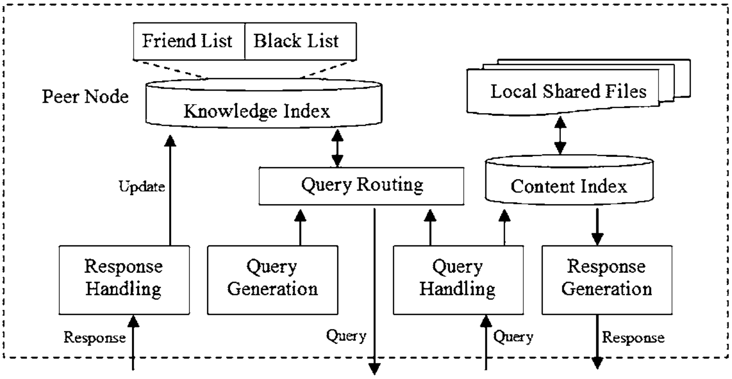
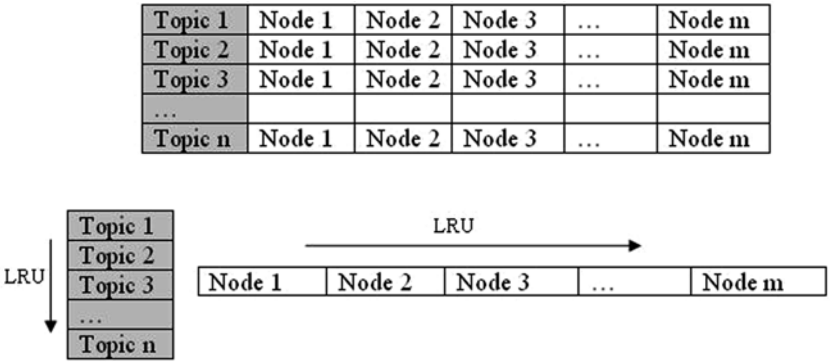


Fig. 1 Main components of an AESLP node

Similarly to people in social networks, each peer node in the AESLP network builds a knowledge index that stores associations between topics and the related peer nodes by the results of searches. Each knowledge index includes two sub-lists (as shown in Fig. 1): a friend list and a black list. In the friend list, peer nodes that have successfully responded to the query are associated with the requested topics. The query originator will put a neighbouring node into the black list associated with the query topic, if no results on this topic are found by the peer node nor its successor(s).

AESLP uses a TTL (Time-to-Live) to prevent infinite propagation. TTL represents the number of times a message can be forwarded before it is discarded. In the AESLP network, a newly joined node will send its first query to a set of randomly selected peer nodes. As shown in Fig. 1, if a search is successful, the query originator updates its friend list to associate the peer nodes that have responded successfully to the query with the requested topic. The new obtained knowledge is stored in the local friend list. In the meantime, the query originator also removes invalid cached knowledge in the local friend list according to the results of searches. The black list is also updated with search results. If the query originator sends a request to a peer node, but no results about the requested topic are found by queries via this peer node, this node will be put into the black list of the query originator associated with the requested topic.

Therefore, peer nodes can learn from the results of previous searches, which makes future searches more focused. When more searches have been done, more knowledge can be collected from search results. If this process continues, each node can cache a great deal of knowledge that is useful to quickly find the peer nodes with the required resources in the future.



Topic 1	Node 1	Node 2	Node 3	…	Node m
Topic 2	Node 1	Node 2	Node 3	…	Node m
Topic 3	Node 1	Node 2	Node 3	…	Node m
…					
Topic n	Node 1	Node 2	Node 3	…	Node m

Fig. 2 Structure of sub-lists in the knowledge index

In AESLP, the information saved in the sub-lists of knowledge index (including the friend list and the black list) are maintained in a two-dimensional map. As shown in Fig. 2, a list can contain a maximum of n topics and each topic is associated with a maximum of m peer nodes, so that a maximum of $n \times m$ pairs of associations between query topics and peer nodes can be stored in a list.

A Least Recently Used (LRU) policy were used to maintain a list without duplicates, where the most recently used topic is at the top and the least recently used at the bottom. The advantage of LRU for removing old index items is that it has constant time and space overhead and can be very efficiently implemented. The least recently used topic will be discarded when the list reaches its maximum n. In the same way, the peer nodes associated with each topic are sorted by time last seen, where the most recently seen node at the head and the least recently seen node at the end. The least recently seen node will be dropped when the maximum size *m* is reached.

Social behaviour 2: in social networks, some events with associated people fade from a person's memory with time [12] and a personal network is adjustable with changing environments [13].

In the AESLP network, peer nodes can update their knowledge about other peer nodes from daily search results. Some old and invalid knowledge is replaced by new obtained knowledge. The invalid knowledge that fails to be updated with daily searches will drop to the bottom of the lists and will be removed by using a LRU policy, when the list reaches a maximum.

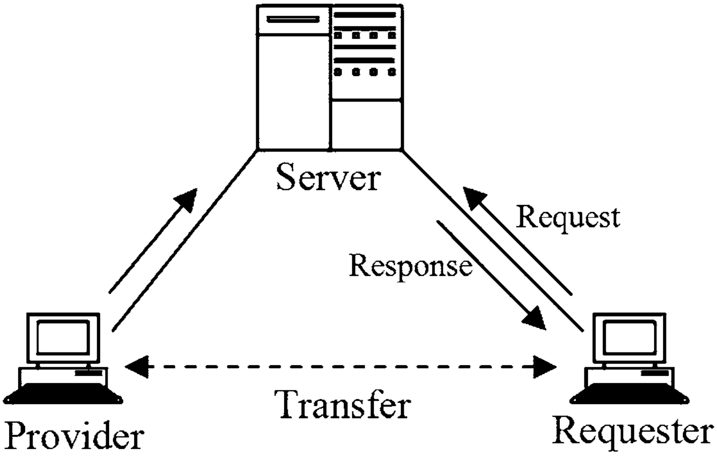


Fig. 3 Example of a knowledge collection

Figure 3 illustrates an example of knowledge collection process of a peer node. Node *A* with an empty knowledge index searches for the files on the topic "radar remote sensing". Since no information is cached, node A will send the query to the randomly selected nodes: node *C*, node *D* and node *E*. Then node *D* further forwards the query to its neighbouring node *F*. In this case, the search is successful at node *F*. Thus, node *F* responds to node *A* with the requested topic "radar remote sensing". Node *A* then associates node *F* with the topic "radar remote sensing" into the local friend list. For a following query on "radar remote sensing", node *A* can find node *F* directly associated with the query from the local friend list and preferentially send the query to node *F* rather than node *D* and node *E*.

As shown in Fig. 3, node *A* updates its black list with search results. Node *E*, a neighbour of node *A*, will be added into the black list associated with the requested topic "radar remote sensing", because no results are found by node *E* nor its successors. But node *D* will not be put into the black list, because node *D*'s successor, node *F*, helps node *D* find the requested files successfully. If node *A* does the same search on “radar remote sensing” again, it will avoid forwarding the query to node *E*, regarding the information in the black list.

3.2 Routing Algorithm for Simple Queries

3.2.1 Query Processing

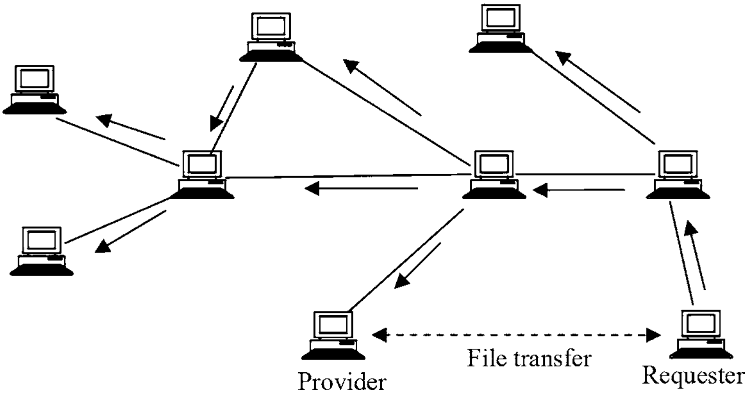


Fig. 4 Query initialisation and processing

When a peer node generates a query, it will search the local friend list to find some associated peer nodes. The query originator will prefer to send the query to these peer nodes found from the friend list and avoid sending the query to associated peer nodes in the black list (Fig. 4). The query originator also attaches a list of “black nodes” associated with the query topic regarding the local black list to help message receivers select peer nodes.

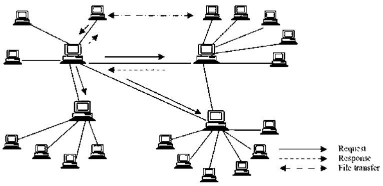


Fig. 5 Flowchart of query handling

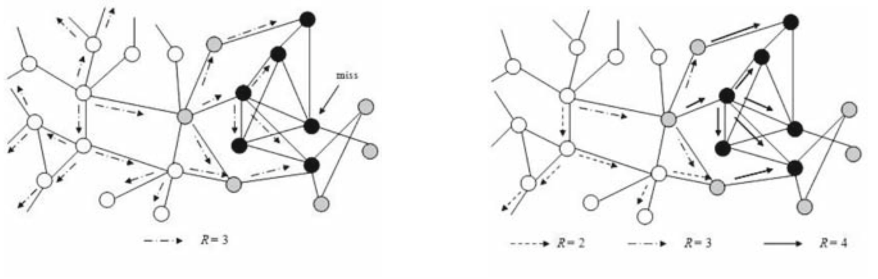
As shown in Fig. 5, when a peer node receives a query, it will first check whether the message has been already received. Redundant messages will be discarded without further processing. Then the peer node will search the local content index to find matched files. If the query needs to be further forwarded ($TTL > 0$), the message receiver will use the local knowledge index to find associated peer nodes using the AESLP routing algorithm and multicast the query to these peer nodes as shown in Fig. 4.

Social behaviour 3: in a social network, communities are self-organised with regard to the common interests. In the AESLP network, because links between peer nodes are built according to the results of searches, a node has more probability to link to other peer nodes with the same interests, which have files of interest to him/her, with a high degree of likelihood. Therefore, peer nodes that have the same interests will be highly connected to each other and form a virtual community spontaneously, which is a similar environment to Watts's model [14] in social networks.

3.2.2 Adaptive Forwarding

Analogously to social networks, AESLP utilises a semantic approach to route queries to a selected subset of neighbouring nodes on the P2P overlay. In addition, AESLP also involves a dedicated strategy to address the network overload problem of existing P2P systems (e.g., Gnutella).

In some existing query routing methods (e.g., [15–17], the number of peer nodes to be forwarded in each hop is set to a static value. A fixed number of peer nodes are selected in each hop neglecting the probability of finding the requested file in these peer nodes. In order to conduct a more efficient search in AESLP, the number of peer nodes to be forwarded is adjustable according to the correlation degree of the selected node to the query between a minimal number FN_{min} and a maximum number FN_{max}. When the correlation degree of a selected peer node is high, there is a high possibility that the selected peer node may share a desired file or has the knowledge about who shares the file. The probability of forwarding the query to this peer node should also be high by defining a high cut-off of node selection. In contrast, if the correlation degree is low, a low cut-off should be set to limit the scope of querying.



(a) a static routing strategy

(b) an adaptive routing strategy

Fig. 6 Self-adaptive routing

Figure 6 shows examples of two routing strategies with a static number of receivers per hop ($R = 3$) and an adaptive number of receivers per hop ($R = 2 \sim 4$), respectively. In Fig. 6, the black dots represent the peer nodes that share the requested resources, while the grey dots show the peer nodes that are highly correlated with the query and know who has the requested resources. As shown in Fig. 6, the adaptive routing strategy achieves better search performance by finding more target nodes with fewer query messages.

3.2.3 Node Selection

Social behaviour 4: for resource discovery in social networks, people usually recall information in memory to find the right people to contact. The persons recalled from memory may directly relate to their requests. For example, Bob wants to borrow an Oxford English dictionary and remembers that he once borrowed it from his friend Alice. Therefore, he can directly contact Alice again for the dictionary.

Social behaviour 5: however, in most circumstances, people cannot find the persons who are directly related to their requests, but people can find some acquaintances that potentially have knowledge about the resources they are looking for, or can provide background information or give advice on how to find the resources [18]. For example, Bob may never have borrowed or cannot clearly remember whether he has ever borrowed an Oxford English dictionary. But Bob believes his friend Alice, who is a linguist, probably has the dictionary or at least she has more knowledge about who has the dictionary. In this case, the Oxford English dictionary is in the area of linguistics and Bob has found that Alice has abundant knowledge on the interest area of linguistics from previous intercommunications. Alice probably does not have the dictionary, but she will use her own knowledge to help Bob find the dictionary with a high likelihood.

The node selection procedure of AESLP is shown in Fig. 7, where n is the number of peer nodes that have already been selected in the first and second phase of node selection. Message receivers only trust the information about the "black nodes" provided by the query originator or by their own local black list concerning the security of information. When a peer node receives a query which needs to be further forwarded, the local routing algorithm will exclude a list of black nodes provided by the query originator from the node selection procedure together with the peer nodes associated with the requested topic in the local black list.

The routing algorithm then starts the three-phase procedure of node selection: searching for the directly related peer nodes, searching for the correlated peer nodes and searching for random peer nodes, from the local friend list. In the first phase, the routing algorithm searches for the peer nodes directly associated with the requested topic from the local friend list and sorts them with the time of receipt. The peer node that is inputted or updated most recently will be selected first. These directly associated peer nodes have the greatest likelihood of finding the requested files. Hence, at most FN_{max} peer nodes will be forwarded.

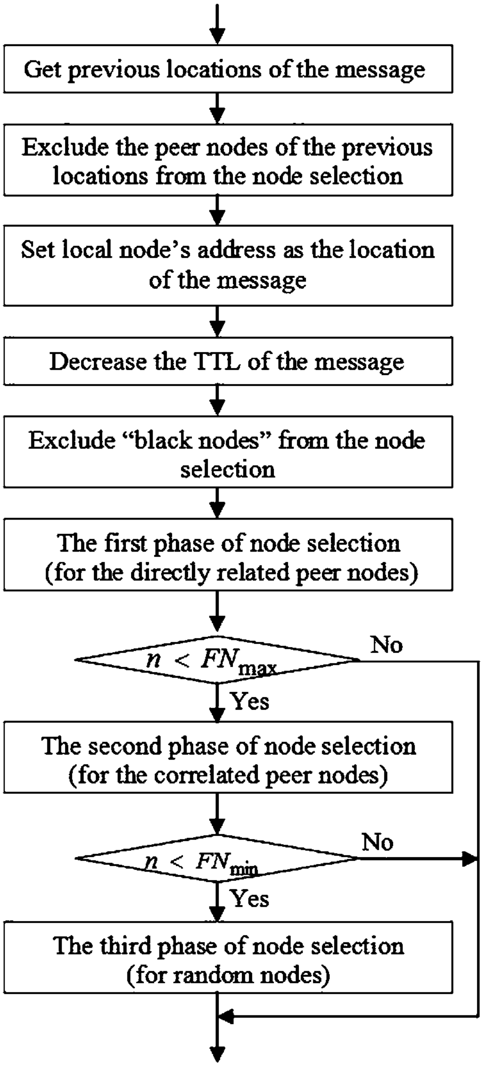

Get previous locations of the message
Exclude the peer nodes of the previous locations from the node selection
Set local node's address as the location of the message
Decrease the TTL of the message
Exclude "black nodes" from the node selection
The first phase of node selection (for the directly related peer nodes)
n < FNmax
No
Yes
The second phase of node selection (for the correlated peer nodes)
n < FNmin
No
Yes
The third phase of node selection (for random nodes)

Fig. 7 Flowchart of the node selection procedure

However, the success probability of finding FN_{max} directly associated nodes in the first phase is very low, especially for new peer nodes with little knowledge. If there are not enough directly associated nodes found in the first phase, the algorithm will move to the second phase that searches for the peer nodes sharing content associated with the interest area of the requested topic in the local friend list. An interest area in AESLP is a semantic area with a set of topics. The corresponding interest area of a specific topic and the other topics in this interest area can be found from the Open Directory Categories [19], which is the most widely distributed database with a hierarchical topic structure. AESLP users use the common topic hierarchy of the Open Directory to formalise a query. When a user generates a query to search files about the topic "Gnutella", the query will be constructed as "Computer: Software: Internet: Client: File Sharing: Gnutella". The closest parent directory "File Sharing" is the interest area of the topic "Gnutella". The other topics in the same area (BitTorrent, Gnutella, FastTrack, Napster, Freenet, Overnet and eDonkey) will be used in the second phase of node selection. Users can also define a category for their own query, if Open Directory cannot provide any satisfactory category for the query.

These peer nodes sharing content associated with the other topics in the same interest area of the requested topic will be sorted according to the degree of correlation to the interest area of the requested topic. The routing algorithm prefers to select the peer nodes with higher degrees of correlation rather than the peer nodes with lower degrees of correlation. If two or more nodes have the same correlation degree, the peer node that responded most recently will be put first.

Social behaviour 6: searching for a piece of information in social networks is most likely a matter of searching the social network for an expert on the topic together with a chain of personal referrals from the searcher to the expert [20].

Social behaviour 7: in social networks, a person does not need to tell everybody he/she is an expert in the areas which has been indicated with his/her social behaviours.

In the AESLP network, it is not necessary for a peer node to declare its interest since that has already been implied by its shared files. If a peer node has a large amount of content in a particular area like an "expert", it is very likely that it will also have other content or knowledge on this area. In our simulations, the correlation degree of a selected node in a particular area is generated by how many topics in the area the peer node is associated with:

$$C = \frac{n_{matches}}{n_{total}} \tag{1}$$

where n_{mathes} is the number of topics in this area that the peer node is associated with and n_{total} is the total number of topics in this area.

Different from any other P2P algorithm, the cut-off criteria R for the second phase are different for different peer nodes between FN_{max} and FN_{min}. The cut-off criterion R of peer node with respect to the query is determined by the correlation degree of the peer node to the interest area of the requested topic with the equation:

$$R = Round(C \times (FN_{max} - FN_{min})) + FN_{min} \tag{2}$$

where the function $Round(x)$ returns the closest integer to the given value x. When the correlation degree of a peer node is very low ($C \approx 0$), there is a low likelihood to find the requested files from the peer node. Therefore, the probability of querying the node should be low with a low cut-off ($R \approx FN_{min}$). In contrast, when the selected node is highly correlated with the area of the requested topic by matching most topics in this area ($C \approx 1$), the cut-off of the node $R \approx FN_{max}$.

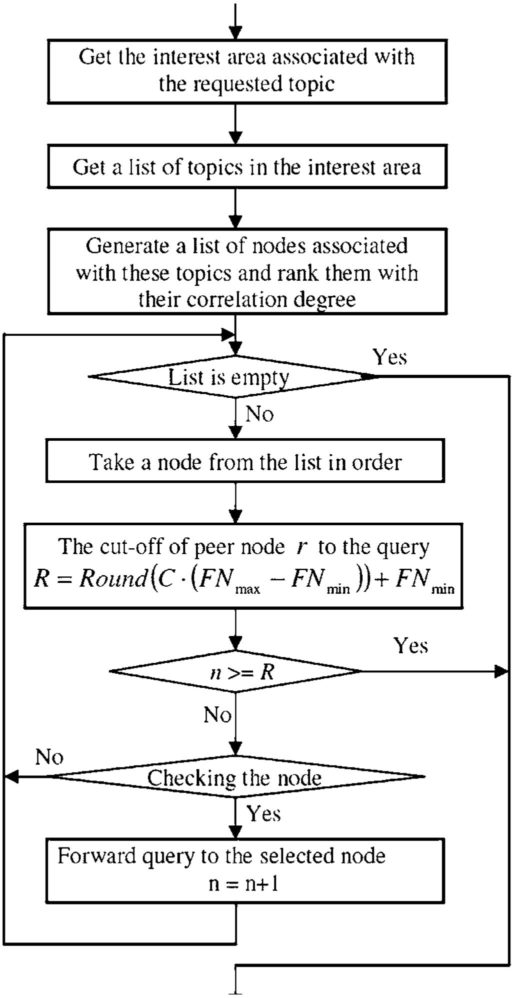

Get the interest area associated with the requested topic
Get a list of topics in the interest area
Generate a list of nodes associated with these topics and rank them with their correlation degree
List is empty
Yes
No
Take a node from the list in order
The cut-off of peer node r to the query
$R = Round(C \cdot (FN_{\max} - FN_{\min})) + FN_{\min}$
$n >= R$
Yes
No
No
Checking the node
Yes
Forward query to the selected node
n = n+1

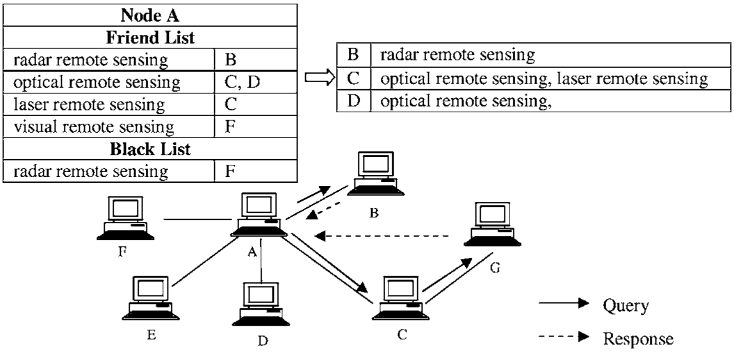
Fig. 8 Flowchart of the second phase of node selection

The flowchart in Fig. 8 shows the second phase of the node selection algorithm. As shown in the flowchart, the peer nodes associated with the interest area of the requested topic are sorted with their correlation degrees. Because peer nodes are taken from the list in order, the cut-offs of these peer nodes R will decrease with the reduced correlation degrees C (according to Equation (2)). But **n** is increased by one for each new node selected. The query will be sent to the selected peer nodes only when the number of selected nodes n is smaller than its cut-off to the query $R(n < R)$.

If $n \geq R$, node selection procedure is completed in the second phase. If all peer nodes associated with the area of the requested topic ($C > 0$) have been taken from the list in the second phase but there are still not enough nodes selected $n < FN_{min}$, the selection procedure will move to the third phase to randomly pick up nodes from the rest of cached nodes irrelevant to the requested topic as well as its interest area ($C = 0$) and forward the query to them, until FN_{min} peer nodes are forwarded ($n = FN_{min}$).

3.2.4 Query Routing Example

Figure 9 illustrates a simple example of query routing with AESLP where $FN_{max} = 5$ and $FN_{min} = 1$. Node A receives a query with the topic "radar remote sensing". Node A will first check the black list to exclude node F which is associated with the requested topic "radar remote sensing" in the black list. In the first round of selection, node B is selected from the local friend list which is directly associated with the requested topic "radar remote sensing". The number of selected nodes n is increased by one: $n = 0 \rightarrow n = 1$.



Node A	
Friend List	
radar remote sensing	B
optical remote sensing	C, D
laser remote sensing	C
visual remote sensing	F
Black List	
radar remote sensing	F

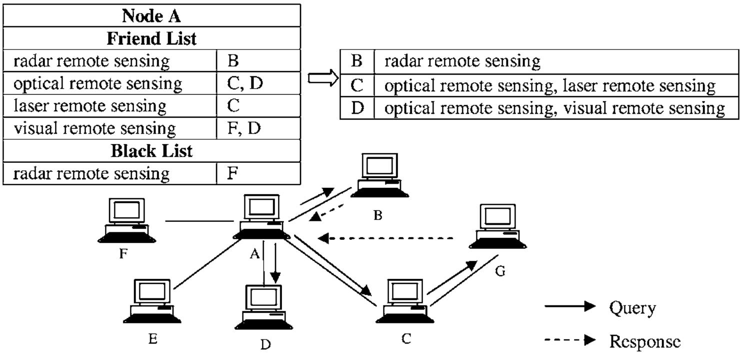
B	radar remote sensing
C	optical remote sensing, laser remote sensing
D	optical remote sensing,

Fig. 9 Query routing example (two nodes selected)

Due to $n < FN_{max}(n = 1, FN_{max} = 5)$, node A further searches for the peer nodes associated with the topics "optical remote sensing", "laser remote sensing", and "visual remote sensing" which are from the same interest area of the requested topic "radar remote sensing". In this case, node A gets node C and node D associated with these topics from the friend list. Since node C is associated with two topics "optical remote sensing" and "laser remote sensing", but node D is associated with only one topic "optical remote sensing" in the area composed by the four topics, node C ($C_C = \frac{2}{4}$) is more correlated with the area of remote sensing than node D ($C_D = \frac{1}{4}$) according to the cached knowledge. Hence, node C will be sorted on the top of node D in the list.

The cut-off of node C is $R_C = \frac{2}{4} \times (5-1) + 1 = 3$ and the cut-off of node D is $R_D = \frac{1}{4} \times (5-1) + 1 = 2$ by using Equation (2). The query will be sent to node C, because the number of selected nodes is smaller than the cut-off of node C: $n < R_C(n = 1, R_C = 3)$. Then $n = 1 \rightarrow n = 2$. The selection procedure will complete and node D is not selected, because $n \geq R_D (n = 2, R_D = 2)$. The actual number of queried nodes is *two* in this case.

Node C may not have the requested files, but it will use its own cached knowledge to propagate the query further and try to find the peer nodes for the query that will have a higher likelihood of success. In this example, node C knows that node G is associated with the requested topic according to its local friend list and the requested file is obtained in node G.



Node A	
Friend List	
radar remote sensing	B
optical remote sensing	C, D
laser remote sensing	C
visual remote sensing	F, D
Black List	
radar remote sensing	F

B	radar remote sensing
C	optical remote sensing, laser remote sensing
D	optical remote sensing, visual remote sensing

Fig. 10 Query routing example (three nodes selected)

Figure 10 illustrates a similar example, where node D is associated with one more topic "visual remote sensing" in the same area of the requested topic. In this case, node D is also selected, because $n < R_D$ (where $n = 2$ and $R_D = \frac{2}{4} \times (5-1) + 1 = 3$)) and the actual number of nodes to be queried is *three* in the second example.

3.3 Routing Algorithm for Multi-Topic Queries

3.3.1 Routing Algorithm for Conjunctive Queries

AESLP also supports queries with conjunctive topics, such as a query on "radar remote sensing" and "mini satellite". For a conjunctive query with multiple topics, a search is successful on a peer node if a requested file is found by matching all query topics. This peer node will respond to the query originator with the requested topics. Moreover, the peer nodes that can find any requested topic of the conjunctive query will also inform the query originator with the matched topic(s), even though they cannot find the requested file completely matching all query topics. This response information will be cached by the query originator for future queries.

Figure 11 shows an example of the knowledge collection process with conjunctive queries. Node A with an empty knowledge index searches for files on "radar remote sensing" and "optical remote sensing", which randomly sends the query

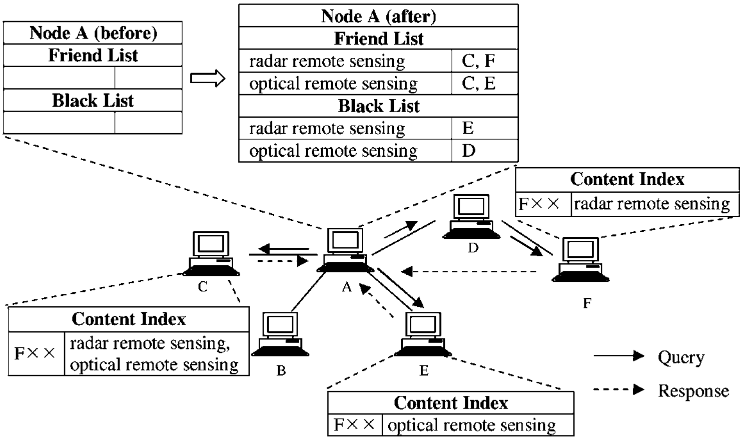


Fig. 11 Example of knowledge collection of a conjunctive query

to node *C*, node *D* and node *E*. The query is successful at node *C* by finding the requested file matching both the requested topics "radar remote sensing" and "optical remote sensing". Thus, node *C* responds to node *A*. Node *A* then adds node *C* associated with the two topics into the local friend list: "radar remote sensing" and "optical remote sensing". Node *E* and node *F* cannot find any matched files, but they can partially satisfy the query by matching one of the requested topics on either "radar remote sensing" or "optical remote sensing". Node *E* and node *F* respond to node *A* which are then associated with a matched topic into the friend list of node *A*. For a following query on "radar remote sensing", node *A* can find at least two peer nodes, node *C* and node *F*, directly associated with the query from the local friend list.

As shown in Fig. 11, node *D*'s successor, node *F*, helps node *D* find the information about one of the requested topics "radar remote sensing", but no results about the other query topic "optical remote sensing" are found in node *D* and its successor(s) (e.g., node *F*). Therefore, node *D* will be put into the black list with the topic "optical remote sensing" only. If node *A* does the same search on "radar remote sensing" and "optical remote sensing" again, the node routing algorithm in node *A* will avoid forwarding the query, regarding information in the black list, to node *D*, because node *D* is associated with one of the requested topics: "optical remote sensing", in the black list.

Similarly to the single-topic query processing, when a node receives a conjunctive query *Q* which needs to be forwarded, the "black nodes" that are provided by the query originator and the "black nodes" that are associated with any requested topic in the local black list will be excluded from the three-phase selection procedure.

In the first phase, the peer nodes that can completely satisfy the conjunctive query will be selected from the local friend list with FN_{max}. If $n < FN_{max}$, the node selection procedure will continue to the second phase to find the peer nodes that are

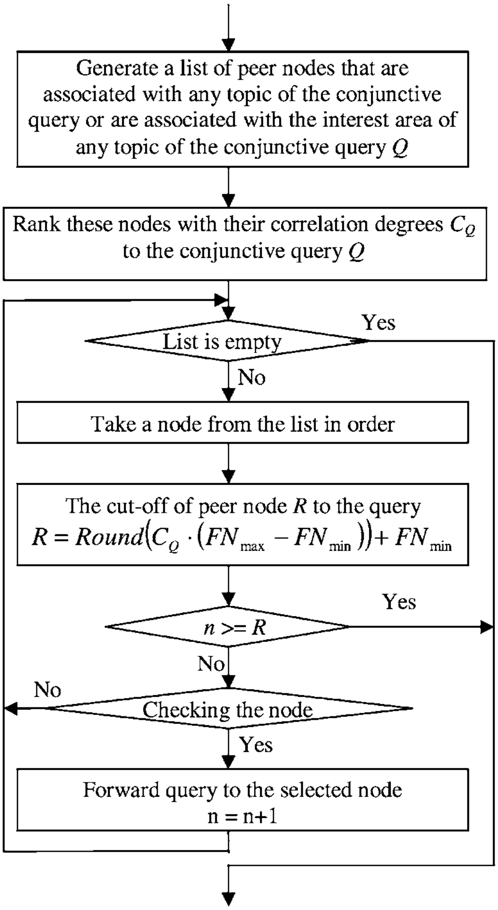

Generate a list of peer nodes that are associated with any topic of the conjunctive query or are associated with the interest area of any topic of the conjunctive query Q
Rank these nodes with their correlation degrees C_Q to the conjunctive query Q
List is empty
Yes
No
Take a node from the list in order
The cut-off of peer node R to the query
$R = Round(C_Q \cdot (FN_{max} - FN_{min})) + FN_{min}$
$n >= R$
Yes
No
Checking the node
No
Yes
Forward query to the selected node
n = n+1

Fig. 12 Flowchart of the second phase of node selection procedure for multi-topic queries

correlated with the query as shown in Fig. 12. Two kinds of peer nodes will be considered: the peer nodes that are associated with any topic of the conjunctive query and the peer nodes that are associated with the interest area of any topic of the conjunctive query.

Analogously to the single-topic query processing, the correlation degree of a peer node to the interest area of a requested topic T is given by Equation (1):

$$C_T = \frac{n_{matches}}{n_{total}}$$

For a conjunctive query with N topics $t_1 \wedge t_2 \wedge t_3 \ldots \wedge t_N$ (where $\wedge$ represents the logical conjunction function: “and”), a peer node is associated with $t_1, t_2, \ldots, t_m$, and is not associated with $t_{m+1}, t_{m+2}, \ldots, t_{m+k}$ but with corresponding correlation degrees $C_{m+1}, C_{m+2}, \ldots, C_{m+k}$(N = m + k).

The correlation degree $C_{indirect}$ of the interest areas that the peer node is not associated with is given by:

$$C_{indirect} = E(C_{t_i}) = \frac{1}{k} \sum_{i=m+1}^{m+k} C_{t_i}$$

We prefer to forward the query to the peer nodes that are directly associated with the most topics of the conjunctive query (with ratio C_{direct}) and simultaneously highly correlated with the interest areas of the requested topics they are not associated to (with ratio $C_{indirect}$). The correlation degree of the node to the conjunctive query Q is given by:

$$\begin{aligned} C_Q &= C_{direct} + (1 - C_{direct}) \times C_{indirect} \\ &= \frac{m}{m+k} + \frac{1}{m+k} \times \sum_{i=m+1}^{m+k} C_{t_i} \ (i = m+1, \ldots, m+k) \end{aligned} \tag{3}$$

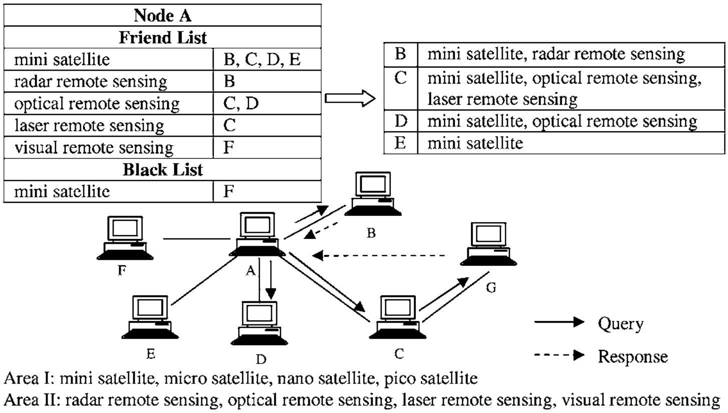
The cut-off criterion R of peer node with respect to the conjunctive query is determined by the correlation degree of the peer node to the conjunctive query Q with the Equation (2):

$$R_Q = Round(C_Q \times (FN_{max} - FN_{min})) + FN_{min}$$

If $n < FN_{min}$, the selection procedure will proceed to the third phase to randomly pick up peer nodes from the rest of cached nodes irrelevant to the requested topics as well as their interest areas ($C = 0$) with cut-off FN_{min}.

Figure 13 shows an example of conjunctive query routing by AESLP where $FN_{max} = 5$ and $FN_{min} = 1$. Suppose node A generates a conjunctive query with two topics: “mini satellite” and “radar remote sensing”. Node A will first check the local black list to exclude node F which is associated with one of the requested topics “mini satellite” in the black list.

In the first round of selection, node B is selected which is directly associated with both the topics “mini satellite”and “radar remote sensing”. Due to $n < FN_{max}(n = 1, FN_{max} = 5)$, node A will further retrieve the peer nodes that are associated with any topic of the conjunctive query and the peer nodes that are associated with the interest area of any topic of the conjunctive query according to the local friend list. In this case, node A finds node C, node D and node E from the local friend list, which are all associated with one of the requested topics: “mini satellite”. Among them, node C is also associated with two topics “optical remote sensing” and “laser remote sensing”in the same area of the other requested topic “radar remote sensing”, node D is associated with one topic “optical remote sensing”, but node E is associated with none in this area. Thus, node C ($C_C = \frac{1}{2} + \frac{1}{2} \times \frac{2}{4} = \frac{3}{4}, R_C = Round(\frac{3}{4} \times (5-1)+1) = 4$) is more correlated with the query than D ($C_D = \frac{1}{2} + \frac{1}{2} \times \frac{1}{4} = \frac{5}{8}, R_D = Round(\frac{5}{8} \times (5-1)+1) = 4$) and node E ($C_E = \frac{1}{2}, R_E = Round(\frac{1}{2} \times (5-1)+1 = 3)$). The request will be sent to node C, because the number of selected nodes to be queried is



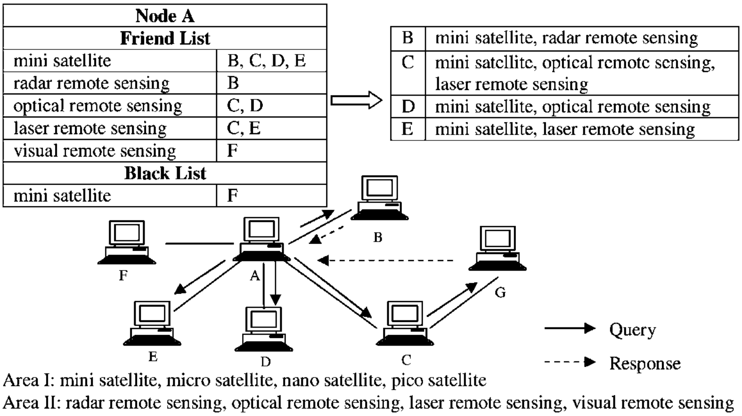
Node A	
Friend List	
mini satellite	B, C, D, E
radar remote sensing	B
optical remote sensing	C, D
laser remote sensing	C
visual remote sensing	F
Black List	
mini satellite	F

B	mini satellite, radar remote sensing
C	mini satellite, optical remote sensing, laser remote sensing
D	mini satellite, optical remote sensing
E	mini satellite

Area I: mini satellite, micro satellite, nano satellite, pico satellite
Area II: radar remote sensing, optical remote sensing, laser remote sensing, visual remote sensing

Fig. 13 Conjunctive query routing example (three nodes selected)

smaller than the cut-off of node C: $n < R_C(n = 1, R_C = 4)$. Then $n+1 \rightarrow n = 2$. Node D is also selected because $n < R_D(n = 2, R_D = 4)$ and $n+1 \rightarrow n = 3$. The selection procedure then stops and node E is not selected because $n \geq R_E(n = 3, R_E = 3)$. The actual number of nodes to be queried is *three* in this case.



Node A	
Friend List	
mini satellite	B, C, D, E
radar remote sensing	B
optical remote sensing	C, D
laser remote sensing	C, E
visual remote sensing	F
Black List	
mini satellite	F

B	mini satellite, radar remote sensing
C	mini satellite, optical remote sensing, laser remote sensing
D	mini satellite, optical remote sensing
E	mini satellite, laser remote sensing

Area I: mini satellite, micro satellite, nano satellite, pico satellite
Area II: radar remote sensing, optical remote sensing, laser remote sensing, visual remote sensing

Fig. 14 Conjunctive query routing example (four nodes selected)

However, if node E is not only associated with the requested topic “mini satellite”, but also with the topic “laser remote sensing” which is in the same area of the other requested topic “radar remote sensing” as in Fig. 14, node E is also selected, because $n < R_E$ (where $n = 3, C_E = \frac{1}{2} + \frac{1}{2} \times \frac{1}{4} = \frac{5}{8}, R_E = Round(\frac{5}{8} \times (5-1)+1) = 4$) and the actual number of nodes to be queried is *four*.

3.3.2 Routing Algorithm for Query Packs

A query pack consists of several conjunctive queries, like (“mini satellite” and “radar remote sensing”) or (“pico satellite” and “visual remote sensing”). To route a query pack, the node selection algorithm first excludes the peer nodes that are associated with any conjunctive query in the local black list and in the black list provided by the query originator. The node selection algorithm then tries to find the peer nodes that can satisfy any conjunctive query in the query pack from the local friend list. A maximum of FN_{max} peer nodes will be selected.

In the second phase, the routing algorithm produces a list of peer nodes that are associated with any topic of the query pack and the peer nodes that are associated with the interest area of any topic of the query pack. For a given peer node, the routing algorithm generates a list of correlation degrees $(C_1, C_2, \ldots, C_M)$ of these peer nodes to all conjunctive queries $(Q_1, Q_2, \ldots, Q_M)$ in the query pack $(Q_1 \vee Q_2 \ldots, Q_M)$ by using Equation (3) (where $\vee$ represents the logical disjunction function “or”). The query pack is successful if any conjunctive query in the query pack can be satisfied. We use the largest correlation degree of a conjunctive query as the correlation degree of the peer node to the query pack $C = MAX(C_1, C_2, \ldots, C_M)$ to generate the cut-off R of the peer node by using Equation (2).

Let’s reuse the example network illustrated in Fig. 13 as an example of message routing of the query pack $(Q_1 \vee Q_2)$, where Q_1 = “mini satellite” $\wedge$ “radar remote sensing” and Q_2 = “pico satellite” $\wedge$ “visual remote sensing”). For node D, the correlation of the first conjunctive query Q_1 is $C_{Q_1} = \frac{1}{2} + \frac{1}{2} \times \frac{1}{4} = \frac{5}{8}$ and the correlation of the second conjunctive query Q_2 is $C_{Q_2} = \frac{1}{2} \times \frac{1}{4} + \frac{1}{2} \times \frac{1}{4} = \frac{1}{4}$. Thus, the correlation degree of node D to the query pack $C = MAX(C_{Q_1}, C_{Q_2}) = \frac{5}{8}$.

These peer nodes are sorted according to their correlation degrees to the query pack. The query pack is forwarded to a given peer node only in case the cut-off R of the peer node is larger than the number n of selected peer nodes, which is the same as the single-topic query and the single conjunctive query processing.

3.4 *Adaptive Query*

In order to efficiently search the network, two kinds of queries are generated at different stages of searches, known as ordinary queries and active queries. For the ordinary queries, the target nodes sharing the desired files will respond to the information related to the requested topic only. For the active queries, the target nodes will not only respond to the requested topic but also inform the query originator of other associated topics it shares in the same interest area.

As illustrated in Fig. 15, when the query originator generates a query with the topic “radar remote sensing”, the target node that shares the desired files will answer the query about “radar remote sensing” as well as the associated topics “optical remote sensing”, “laser remote sensing” and “visual remote sensing”. The newly

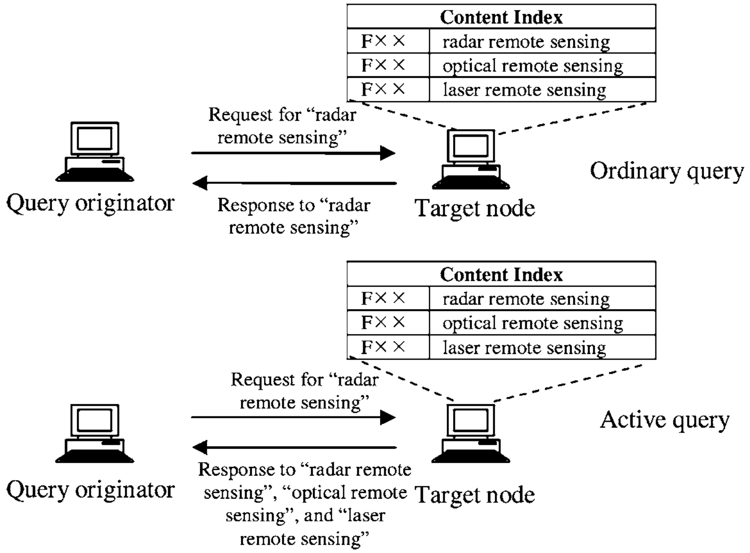

Content Index
F×× radar remote sensing
F×× optical remote sensing
F×× laser remote sensing
Request for "radar remote sensing"
Query originator
Response to "radar remote sensing"
Target node
Ordinary query
Content Index
F×× radar remote sensing
F×× optical remote sensing
F×× laser remote sensing
Request for "radar remote sensing"
Query originator
Response to "radar remote sensing", "optical remote sensing", and "laser remote sensing"
Target node
Active query

Fig. 15 Ordinary query and active query of AESLP

obtained information will be put into the local friend list by the query originator for future queries.

With these active queries, the query originator can gather more pieces of knowledge from each successful query, but extra traffic will be generated for shipping such additional knowledge. The extra traffic could be significant if every node generates all queries in this manner, which is difficult to be handled by bandwidth-limited networks. In contrast, if peer nodes search the network only with ordinary queries, each new node accumulates knowledge slowly by gathering one piece of knowledge from each successful query, especially for those peer nodes which are seldom online or request the network.

To address these issues, AESLP adopts a trade-off solution for bandwidth-limited networks. The recently joined nodes will utilise active queries to quickly accumulate a large amount of useful information regarding their interests. After the cached knowledge reaches a certain threshold ratio r of the maximum size of the friend list $n_{max\_knowledge}$, the peer nodes will use ordinary queries to discover the required files with low traffic cost. If the ratio of cached knowledge $r_{cached} \leq r$, active queries will be used to search the network. Otherwise, ordinary queries will be adopted. Moreover, this process is not only applicable for recently joined nodes, but also enables peer nodes to quickly recover from unpredictable knowledge loss.

Social behaviour 8: the query generation strategy of AESLP is similar to the human strategy in social networks where newly joined persons normally are more active to adapt to new environments. When a person joins to a new society, he/she will not only passively learn knowledge by remembering useful information from daily occasional events happening in the society, but also actively collect potentially useful knowledge consciously. When he/she seeks out help in a new society, communication with the other people who can be of assistance is not normally limited to a strict exchange of assistance; often, he/she would like to know more about the people who can be of assistance in order to expand his/her knowledge of the new society, which may be useful at a later time.

4 Simulations Methodology

In this section, we evaluate the performance of AESLP by simulations in a dynamic environment with a comparison to other relevant methods. The AESLP simulator is programmed using the Java Language. The main components are illustrated in Fig. 16.

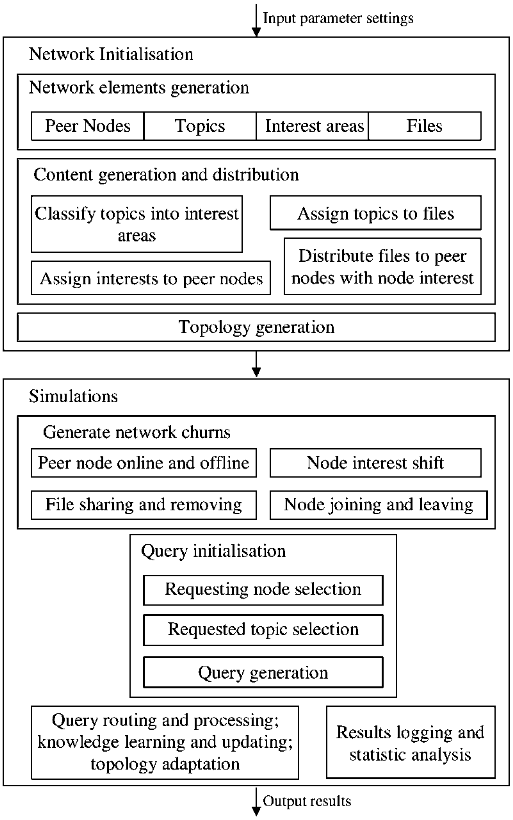

Input parameter settings
Network Initialisation
Network elements generation
Peer Nodes
Topics
Interest areas
Files
Content generation and distribution
Classify topics into interest areas
Assign topics to files
Distribute files to peer nodes with node interest
Assign interests to peer nodes
Topology generation
Simulations
Generate network churns
Peer node online and offline
Node interest shift
File sharing and removing
Node joining and leaving
Query initialisation
Requesting node selection
Requested topic selection
Query generation
Query routing and processing; knowledge learning and updating; topology adaptation
Results logging and statistic analysis
Output results

Fig. 16 Structure of the AESLP simulator

4.1 Content Generation and Distribution

In the simulations of AESLP, each file is shared with a few topic keywords. The topic keyword distribution to files is uneven in P2P file sharing networks, where popular topics are widely distributed to files but unpopular topics only attract little attention by people. This phenomenon affects the performance of resource discovery in P2P networks, where files with popular topics can be easily targeted, but files with unpopular topics are difficult to be discovered. Previous studies observe that the distribution of keywords in files could be approximated by the Zipf's law in the form of $y \sim \frac{1}{x^{\alpha}}$, where y is frequency, x is rank and α is constant. The estimated distribution in the study [20] has been followed in our simulations to generate topic keyword distribution to files. In each simulation, we generate 1280 topics, distribute them to 10,000 files, and each file is randomly assigned three topics.

Previous measurement studies have shown that the distribution of the number of shared files in P2P networks is also unbalanced. Some nodes observed in existing P2P networks tend to download a large number of files, but share few files or none at all [21]. The performance of P2P networks is dependent on the number of shared files each peer node chooses to share. These requested files are more likely to be discovered in the peer nodes sharing a large volume of files. Moreover, these peer nodes also possibly attract more queries from many other peer nodes. In the simulations, we implement the distribution of file sharing in the measurement study [20], where about 42% of peer nodes share 10 files or less including 27% of "free-riders" who share nothing to the network.

The measurement study [22] for the music sharing network in Stanford shows that most peer nodes only share one or a few styles of music that are highly correlated with users' interests. In our simulations, each peer node is randomly assigned a primary interest area and shares a number of files to the network with a probabilistic method: these shared files are mostly relevant to the primary interest area of a node with a probability of 90%, but are occasionally irrelevant to this area. For files relevant to the primary interest area, at least one of the topics of each file should be in the interest area of the hosting node. A total of 32 interest areas are generated and each covers 40 topics.

4.2 Topology Initialisation and Evolution

In order to better observe the evolution of network topology in the simulations, we set up a growing network started with a small-size random network (with 100 peer nodes). In the beginning of each simulation, each peer node randomly connects to four peer nodes bi-directionally to generate a random topology. Since there has been no interaction between peer nodes at the beginning of each simulation, each peer node keeps an empty knowledge index. Each list of the knowledge index can contain a maximum of 60 topics and each topic can be associated a maximum of 60 peer nodes. The threshold ratio r of AESLP is defined as 80%.

Some popular P2P networks are growing very fast on the Internet [23]. However, previous measurement studies (e.g., [24]) have observed that the size of some mature P2P networks stays constant. The phenomenon of quick growth to stability has not been considered by any other P2P simulation, to the best of my knowledge. The simulation network starts from a small set of peer nodes (100 peer nodes). A number of peer nodes (30 peer nodes) join the network each day of the first month in each simulation until the network reaches 1000 peer nodes. Then the network becomes a mature network with 1000 peer nodes.

4.3 Network Churns

In the dynamic Internet environment, network churns are usually caused firstly by peer nodes frequently going online and offline and secondly by content sharing and removing. A high network churn significantly influences the performance of P2P systems, which even leads to the maintenance difficulty of consistent DHTs [25–27] in structured P2P systems. The study [24] measures network churns by using user IDs instead of IP addresses which have been used in previous measurement studies (e.g., [28]). IP address aliasing is a significant issue in the deployed P2P systems (almost 40% of peer nodes use more than one IP address over one day according to their measurements). Therefore, our simulations follow the availability distribution of peer nodes in the study [26], where nearly 50% of peer nodes are present on the network less than 30%.

The study [29] argues that user interest shift is a factor affecting performance of P2P file-sharing networks, especially in today's dynamic information era. To address this issue, 1% of peer nodes randomly shift their interest each day in the simulations. Their major requested topics and additional file sharing will follow the new interests after shifting interest. Content sharing of each peer node changes with time and users' interest, which has seldom been considered by previous P2P simulations. To simulate the dynamics of file sharing, we randomly pick 1% of peer nodes to share an extra 10% files to the network and 1% of peer nodes to remove 10% of shared files from the network every day.

Network churns in these cases could affect the "correctness" of information in the knowledge index. The selected peer nodes that previously had the requested files could be offline from the network at the moment of requesting. Or, the requested files that were previously available on the selected peer nodes could have already been removed from the network.

4.4 Search Network

In each time step of simulations, we randomly choose an online node as the query originator and start a lookup with a query topic. According to the study [22], peer nodes are only interested in a subset of total available content on the network. In order to resemble query behaviours in P2P networks [30, 31], these query topics are generated with a probabilistic method: each topic is randomly selected from the current primary interest area of the query originator with a probability $p = 90\%$, but sometimes from a random area with a probability of $((1-p) = 10\%)$. Each query is tagged by a TTL to limit the life time of a message to four hops in the simulations. Queries are then propagated with the AESLP routing algorithm.

Even though request frequency is variable for different users in different periods, the study [21] observes that each peer node generates an average of two requests each day. This has been implemented in our simulations. One simulation day is defined as: $number\ of\ nodes \times request\ frequency$ time steps, which varies according to the number of peer nodes in the simulation network from 200 (100×2) time steps a day in the network of 100 peer nodes to 2000 (1000×2) time steps each day in the network of 1000 peer nodes. We run simulations to trace the results of about two months (60 days, 92,100 time steps). Each average result is generated from the experimental results of each day.

5 Simulation Results

5.1 Initial Simulations

Due to the complexity of AESLP and numerous customised functions offered by the AESLP simulator, there are many combinations of parameters to experiment with and lots of scenarios to test which could generate far too many graphs to analyse. In this section, we only present an analysis of simulation results from the most pertinent experiments as we see.

Many search methods are emerging in unstructured P2P system in the last decade. Most of existing search methods, such as [32, 33], are obviously different from the methods we are presenting in this chapter. In contrast, NeuroGrid [34] is a well-known method closely related to AESLP, which enables peer nodes to learn the results of previous searches to make future searches more focused. In this section, NeuroGrid will be simulated and analysed as a benchmark to evaluate the performance of AESLP. Additionally, as the first unstructured P2P search method, the Gnutella-like random protocol has been widely used as a benchmark for many follow-up unstructured protocols (e.g., [26]), which is also simulated and compared as a benchmark to evaluate the performance of NeuroGrid.

In this section, the initial simulation results are presented to provide a comparison among the AESLP, two relevant methods: RAN and NEURO, and two derived methods: OSLP and ASLP:

- RAN: a constrained Gnutella routing strategy. Received queries are randomly passed to FN_{min} connected peer nodes in each hop.

- NEURO: NeuroGrid routing strategy with the adaptive number of receivers. In each hop, received queries are passed to a maximum of FN_{max} peer nodes that are directly associated with the requested topic from the local knowledge index. If not enough matches are found ($< FN_{min}$), the algorithm randomly forwards a query to FN_{min} peer nodes from the rest of the connected nodes.
- OSLP: a derived method of AESLP which enables peer nodes to search the network with ordinary queries only. OSLP can be regarded as a special type of AESLP with the threshold ratio $r = 0\%$.
- ASLP: a derived method of AESLP which enables peer nodes to search the network with active queries only. ASLP can also be regarded as a special type of AESLP with the threshold ratio $r = 100\%$.

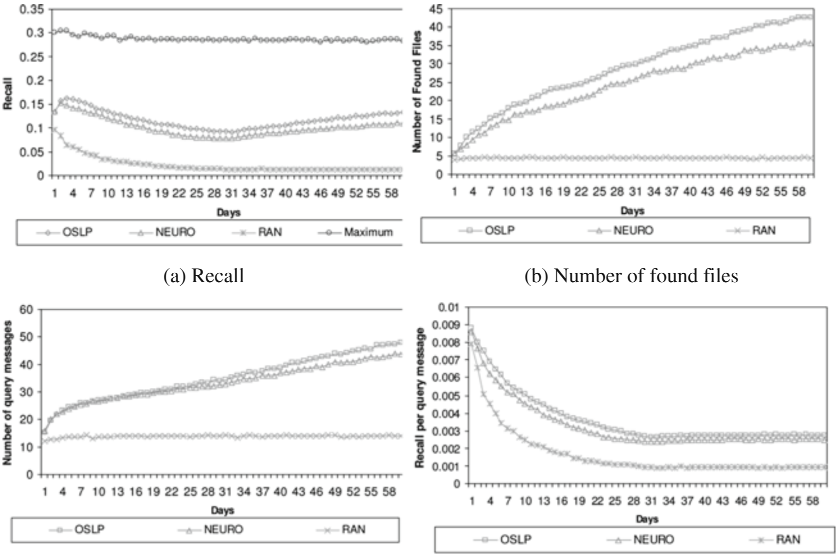
There are no active queries in the NEURO algorithm. In order to do an equivalent comparison, NEURO is compared with one of the derived methods: OSLP (which searches the network with ordinary queries only) to test the performance of the social-like P2P routing protocol. OSLP will be further compared to AESLP and the other derived method: ASLP to show the gains and deficiencies of active queries. As discussed in Section 4, the default parameters are set in Table 1.

Table 1 Default simulation parameters

Parameter	Value
Content parameters	
Files	10,000
Topics	1280
Interests	32
Network parameters	
Nodes	$100 \sim 1000$
Newly joined nodes (per day)	30
Size of lists	60×60
Query parameters	
TTL	4
Query Topic	1
FN_{max}	5
FN_{min}	2
Request structure	90%
Time steps (searches)	92100

5.2 Performance Comparison to Relevant Methods

From the results in Fig. 17a, b, OSLP achieves better performance than NEURO and RAN by finding more files. Maximum possible recall is defined as the ratio of the number of files on online nodes that match to the query to the total number of matched files in the network. In Fig. 17a, the maximum possible recalls are all below 31%, because a large amount of files are available on a large number of offline nodes. Since many new files are added into the network by newly joined peer nodes, recall decreases during the network growing period, while the number of found files keeps increasing.



(a) Recall

(b) Number of found files

(c) Number of query messages

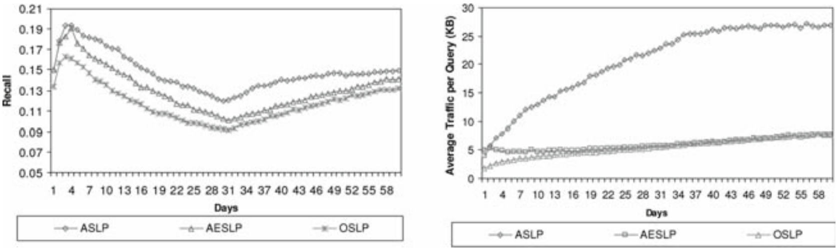
(d) Recall per query message

Fig. 17 Performance comparison

As shown in Fig. 17c, OSLP needs a few more query messages introduced by the second phase of node selection procedure, but the search efficiency of OSLP is still better than NEURO by achieving higher recalls per query message as shown in Fig. 17d. At the early stage of searches, it is very difficult for peer nodes to find the directly associated nodes with the requested topic from the local knowledge index by using either OSLP or NEURO method due to the limited knowledge cached, but OSLP is capable of retrieving the peer nodes which share the associated files with the relevant topics more often. These selected peer nodes which are highly correlated with the interest area of the requested topic have more knowledge about the query than randomly selected peer nodes. Thus, OSLP can find the requested files more efficiently with the same knowledge. More successful searches, in turn, help to build the knowledge index more efficiently. Therefore, OSLP has better search capabilities and better knowledge-collecting capabilities. With these advantages, OSLP achieves better performance than NEURO and RAN.

5.3 Performance Comparison to Derived Methods

The performance of AESLP is further evaluated by comparing to the two derived methods: OSLP ($r = 0\%$) and AESLP ($r = 100\%$). The effect of active queries is analysed in this experiment.



(a) Recall

(b) Average traffic per query

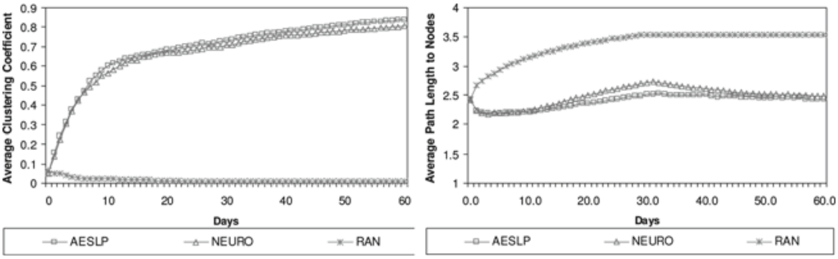
Fig. 18 Performance comparison to derived methods

With active queries, ASLP and AESLP achieve better performance than OSLP as shown in Fig. 18a. Since ASLP utilises high-cost active queries for all searches, ASLP reaches even better performance than AESLP. Moreover, AESLP and ASLP can discover the requested resources more efficiently in the early stage, because they can more rapidly establish the knowledge index than OSLP by gathering more pieces of knowledge from each successful query.

Threshold can be set according to the network bandwidth. ASLP is a special type of AESLP with the threshold ratio $r = 100\%$. Since ASLP achieves the best performance by achieving a higher recall as shown in Fig. 18a, ASLP ($r = 100\%$) is recommended for use in the network with abundant bandwidth. Figure 18b shows the average traffic in bytes generated per query, where we use the message structure of Gnutella protocol and the length of a topic is set as 50 characters. The average traffic in bytes generated per query is the sum of the network traffic generated by not only query messages but also response messages. As shown in Fig. 18b, the traffic generated by each ASLP query is increasing to a huge value by visiting more peer nodes and transferring more knowledge information, which can be a heavy traffic load for the network when many queries occur at the same time. However, the performance of AESLP has been clearly improved with only little more traffic compared to OSLP. Therefore, AESLP is a good trade-off solution for the bandwidth-limited networks which achieves good performance with relatively low traffic.

5.4 Topology Evolution

In the Watts's model [14], a small world network is a kind of networks with a high clustering coefficient and a short average path length to other peer nodes. These two properties of small world networks have been recorded to observe topology evolution in the simulations.



(a) Average clustering coefficient

(b) Average path length

Fig. 19 Topology evolution

Figure 19a, b show the average clustering coefficient and the average path length observed in the simulations. As shown in Fig. 19a, the clustering coefficient of AESLP is greater than other routing methods. Even though the network size increases quickly at the early stage of the simulations, the average path lengths of AESLP and NEURO only increase a little due to the increasing connectivity of network. The average path length of AESLP increases even less. With the active behaviours of newly joined AESLP nodes, these new nodes are more capable of quickly adapting to the new environment, which only slightly affect overall performance.

The average path length of AESLP is only marginally smaller than that of NEURO method. However, by using different routing strategies, search performances are clearly different as shown in Fig. 17. Thus, AESLP is more capable of discovering the short path between the query originators and the target peer nodes. From the results shown in Fig. 19a, b, we can see that the small-world phenomenon appears in the AESLP network with a high clustering coefficient and a short average path length. The visibility of peer nodes is increased by the well-connected nodes which are useful to improve the search performance as shown in Fig. 17.

5.5 Effects of Parameters

In this section, simulations employing different parameters are carried out to show the effects and contributions of simulation parameters (e.g., the size of lists, request structure, the number of receivers in each hop).

5.5.1 Size of Lists

The size of list in the knowledge index is an important parameter, which determines the storage overhead required for each peer node. In this section, we will compare the search performance of AESLP with different sizes of lists. The simulation parameters changed for this experiment are shown in Table 2.

Table 2 Changes to parameters modifying the size of lists

Parameter	Value
Size of lists	$0\times 0; 30\times 30; 60\times 60; 90\ \times 90$

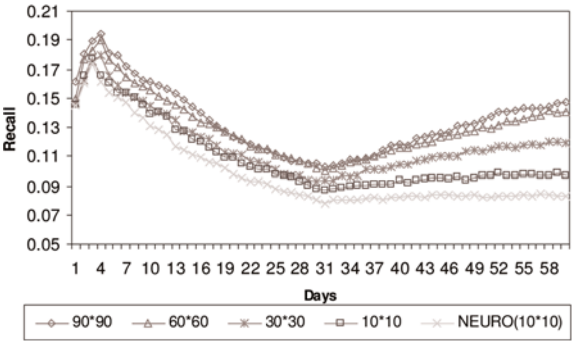


Fig. 20 Recall with different sizes of knowledge index

Figure 20 shows recall by AESLP in the network where each node has a knowledge index with lists containing a maximum of 10×10, 30×30, 60×60 and 90×90 entries (pairs) between topics and associated peer node addresses, respectively. Clearly shown in Fig. 20, the query originators have more difficulty in finding the requested files from the nodes with less knowledge. However, recall only increases a little by changing the size of lists from 60×60 to 90×90 entries. This result suggests that a large knowledge index containing most commonly used topics achieves close performance to a knowledge index with an even larger size. As shown in Fig. 20, in the worst case of 10×10 entries, AESLP still achieves obviously higher performance than NEURO due to its better knowledge collection capability and search capability.

5.5.2 Request Structure

AESLP is simulated with different request structures. By using the AESLP simulator, the requested topic is selected from the primary interest area of the query originator with a probability p, but is from a random area with a probability $(1-p)$. In the case of $p = 90\%$, 90% of the requested topics are randomly selected from the interest area of the query originator. On the contrary, in the case of $p = 0\%$, a purely random topic is chosen as the requested topic which is the worst case since the query originator cannot benefit from the repeated queries in its interest area. Parameters are set as shown in Table 3.

Table 3 Changes to parameters for simulation modifying request structure

Parameter	Value
Request structure	0%; 50%; 90%

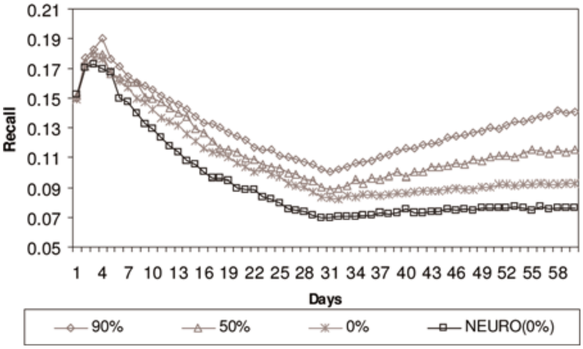


Fig. 21 Recall with different request structures

Figure 21 shows the results of recall by AESLP on the representative samples of p of 0, 50, and 90%, respectively. In the simulations, the request scope is enlarged by setting a smaller p. Since the probability of finding directly associated peer nodes from the knowledge index decreases with smaller p, recall decreases along with p, which means the peer nodes are generally more difficult to target the requested files in the network where users have very wide interests. But the performance of AESLP is still better than that of NEURO even in the worst case of $p = 0\%$ as shown in Fig. 21, because AESLP can still find the peer nodes that potentially have the knowledge about queries even if the directly associated peer nodes cannot be found from the local knowledge index.

5.5.3 Number of Receivers

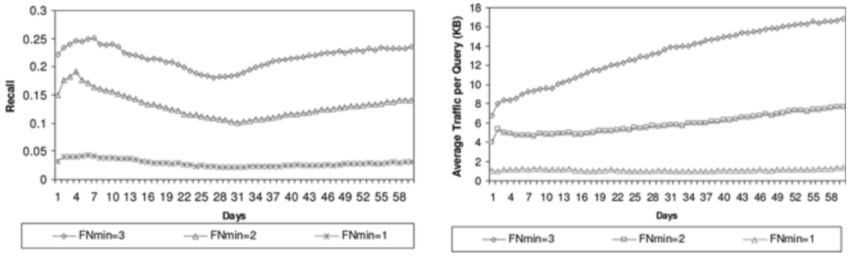
The minimum number of receivers FN_{min} and the maximum number of receivers FN_{max} are important factors to achieve adaptive query forwarding. In this experiment, AESLP is simulated with different FN_{min} and FN_{max} to see the effect that each setting makes.

Effect of the Minimum Number of Receivers

The different minimum numbers of receivers tested in the experiment are shown in Table 4.

Table 4 Changes to parameters for simulation modifying the maximum number of receivers

Parameter	Value
FN_{min}	1; 2; 3



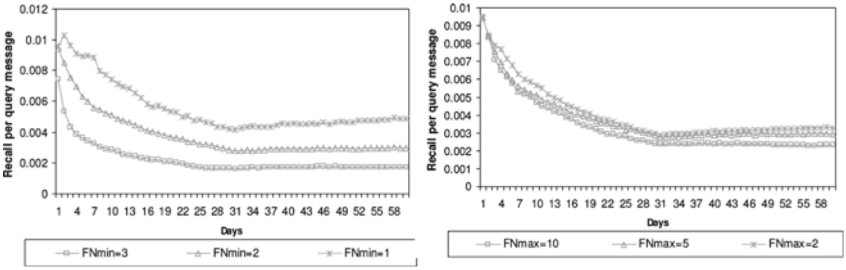
(a) Recall

(b) Average traffic per query

Fig. 22 Alteration of the minimum number of receivers in each hop

As shown in Fig. 22a, b, recall achieved at the end of simulation increases by about 65% by changing FN_{min} from 2 to 3, while the traffic generated per query increases even more significantly by about 120%. This result shows that network traffic is very sensitive to the change of the parameter FN_{min}, where the number of query messages jumps from $O(\sum_{i=1}^{4} 2^i) = O(30)$ to $O(\sum_{i=1}^{4} 3^i) = O(120)$ $(TTL = 4)$. Therefore, flooding would occur in the network by setting a large FN_{min}.

From the results in Fig. a, recall per message decreases clearly with increasing FN_{min}. However, if a very small FN_{min} is defined as $FN_{min} = 1$, a very limited peer nodes could be accessed for answering a query, which seriously affects the speed of knowledge collection. Therefore, recall is very low in the case of $FN_{min} = 1$. From the results discussed above, we suggest that FN_{min} should be carefully defined by application developers who would consider adoption of AESLP in their application. In this case, $FN_{min} = 2$ is a good trade-off to achieve both high search performance and efficiency.



(a) with alteration of the minimum number of receivers

(b) with alteration of the maximum number of receivers in each hop

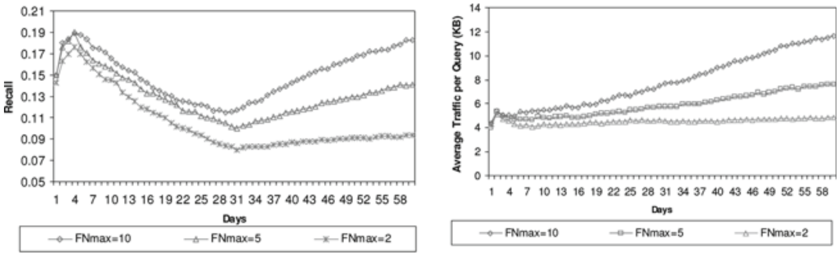
Fig. 23 Recall per message

Effect of the Maximum Number of Receivers

The different maximum numbers of receivers tested in the experiment are changed as shown in Table 5.

Table 5 Changes to parameters for simulation modifying the maximum number of receivers

Parameter	Value
FN_{max}	2; 5; 10



(a) Recall

(b) Average traffic per query

Fig. 24 Alteration of the maximum number of receivers in each hop

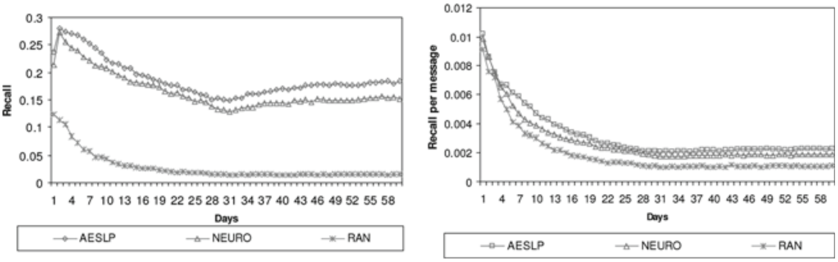
As shown in Fig. 24a, b, recall increases by about 30% while the network traffic increases about 50% by changing the maximum number of receivers in each hop FN_{max} from 5 to 10. Compared to the results with changing the minimum number of receivers FN_{min} shown in Fig. 22b, network traffic is more sensitive to the alternation of the minimum number of receivers FN_{min} rather than the alternation of the maximum number of receivers FN_{max}.

The number of receivers in each hop is adjusted according to the correlation of the selected peer nodes to the query. It is very difficult for a peer node to reach a very high correlation by matching most of topics in a specific area. The number of receivers in each hop is usually much less than the maximum number of receivers FN_{max}, while the number of receivers in each hop must be larger than the minimum number of receivers FN_{min}. Therefore, the total network traffic is more correlated to FN_{min} rather than FN_{max}.

Recall per message is only slightly changed by alternation of FN_{max} (as shown in Fig. 23b) compared to the alteration of FN_{min}, since adaptive lookups are achieved by AESLP. Since a high efficiency is performed by a bigger FN_{max}, application developers could consider defining a bigger FN_{max} to achieve a higher recall rather than a bigger FN_{min}.

5.6 *Query Packs*

We continue to evaluate the performance of AESLP by simulations with query packs. In this experiment, each query pack consists of two conjunctive queries and each conjunctive query contains two query topics, such as ("mini satellite" and "radar remote sensing") or ("pico satellite" and "visual remote sensing"). Because the matching probability decreases significantly to match both of the querying topics, in this experiment, the querying topics are chosen under the assumption that there is at least one possible file existing in the network that can satisfy the query pack.



(a) Recall

(b) Recall per message

Fig. 25 Query packs routing

From the results in Fig. 25a, b, AESLP achieves better performance than NEURO and RAN, which is similar to that of single keyword lookups. Compared to single keyword lookups, recall of each query pack still keeps high. But the number of found files drops to about 16 files by a query pack at the end of simulation, because the number of files matching both two topics is much less than the number of files matching one topic.

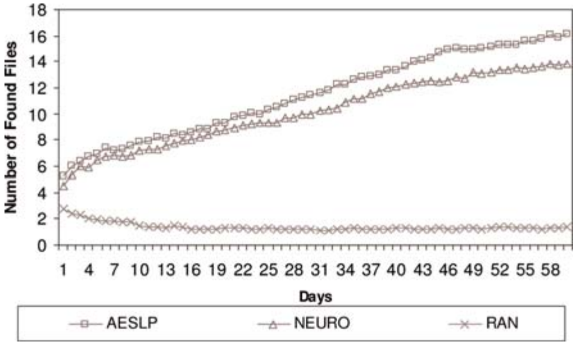


Fig. 26 Number of found files for query packs

Different from single keyword searches, we are surprised to see that the number of found files by RAN decreases at the early stage of searches, while the number of found files by NEURO and AESLP keep increasing as shown in Fig. 26. Because the querying topics are chosen under the assumption that there is at least one possible file existing in the network that can satisfy the query pack, the actual matching probability of RAN decreases with new adding files in the early stage, which leads to the decrement of the number of found files.

6 Conclusions and Future Work

Due to the similarity of social networks and P2P networks, we believe that human strategies in social networks are useful for improving resource discovery by building an efficient social-like peer-to-peer network. In this chapter, we have presented AESLP for resource discovery by self-organising autonomous peers with social strategies.

In the AESLP network, each peer node can learn from the previous search results, both in success and failure, which makes future searches more efficient. No extra communication overhead is required for AESLP to obtain additional information from neighbouring nodes on the P2P overlay. Each peer node maintains a knowledge index (including a friend list and a black list) about resources located in the network. Each peer node works as an autonomous agent which provides local message processing and routing services. AESLP is not only efficient for the previously queried topics but also for the topics that have not been previously queried.

AESLP enables peer nodes not only to forward queries to the “good” nodes that can potentially offer the requested resources or know who has the requested resources, but also to avoid sending queries to the “bad” nodes that cannot help find any requested files.

AESLP is a self-adaptive algorithm. The number of peer nodes to be forwarded in each hop is adaptive according to the correlation degree of the peer node to the query. Moreover, the different types of queries are also utilised in accordance with different search stages, which enables peer nodes to accumulate knowledge efficiently with low communication cost. AESLP is an also self-organising algorithm, peer communities are formed and maintained spontaneously, where peer nodes are self-organised based on their daily intercommunications. Each peer node can automatically detect potential interests of other peer nodes in the network according to their previous behaviours and preferentially links to the peer nodes that have similar interests. Global behaviours then emerge as the result of all the local behaviours that occur. Finally, the peer nodes with similar interests will be highly connected to each other.

In future work we will further optimize and extend the social-like peer-to-peer algorithms to deal with more complex queries and cooperate with semantic languages. Although AESLP is designed for resource discovery in P2P file sharing networks, which is also applicable and will be extended for service discovery in Grid computing environments with numerous service providers. With the cooperation of Grid computing environments, the usage of P2P networks could be broadened from simple file provision to more advanced services, such as sharing redundant computing power for complicated scientific calculation and sharing extra bandwidth for real time video transmission.

References

1. Liu, L., Antonopoulos, N., Mackin, S.: Social Peer-to-Peer for Resource Discovery. 15th Euromicro International Conference on Parallel, Distributed and Network-based Processing, Naples, Italy (2007)
2. Liu, L., Antonopoulos, N., Mackin, S.: Fault-tolerant Peer-to-Peer Search on Small-World Networks. Journal of Future Generation Computer Systems 23 (2007) 921–931
3. Hui, K.Y.K., Lui, J.C.S., Yau, D.K.Y.: Small World Overlay P2P Networks. Workshop on Quality of Service, Montreal, Canada (2004)
4. Salter, J., Antonopoulos, N.: An Optimised 2-Tier P2P Architecture for Contextualised Keyword Searches. Journal of Future Generation Computer Systems 23 (2007) 241–251
5. Triantafillou, P.: PLANES: The Next Step, In Peer-to-Peer Network Architectures. SIGCOMM Workshop on Future Directions in Network Architectures, Karlsruhe, Germany (2003)
6. Sripanidkulchai, K., Maggs, B., Zhang, H.: Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. IEEE Infocom, San Francisco (2003)
7. Loser, A., Staab, S., Tempich, C.: Semantic Social Overlay Networks. Journal of Selected Areas in Communications 25 (2007) 5–14
8. Broekstra, J., Ehrig, M., Haase, P., Harmelen, F.v., Kampman, A., Sabou1, M., Siebes1, R., Staab, S., Stuckenschmidt, H., Tempich, C.: A Metadata Model for Semantics-based Peer-to-Peer Systems. WWW Workshop on Semantics in Peer-to-Peer and Grid Computing, Budapest, Hungary (2003)
9. Cholvi, V., Felber, P., Biersack, E.: Efficient search in unstructured peer-to-peer networks. European Transactions on Telecommunications 15 (2004)
10. Bender, M., Crecelius, T., Kacimi, M., Michel, S., Parreira, J.X., Weikum, G.: Peer-to-Peer Information Search: Semantic, Social, or Spiritual? IEEE Data Engineering Bulletin 30 (2007) 51–60

11. Newcomb, T.M.: Social Psychology : the Study of Human Interaction. - 2nd ed. Revised. Routledge and Kegan Paul, London (1975)
12. Cowan, N., Nugent, L.D., Elliott, E.M., Ponomarev, I., Saults, J.S.: The Role of Attention in the Development of Short-Term Memory: Age Differences in the Verbal Span of Apprehension. Child Development 70 (1999) 1082–1097
13. Fisher, K.M., Lipson, J.I.: Information Processing Interpretation of Errors in College Science Learning. Instructional Science 14 (2004) 49–74
14. Watts, D., Strogatz, S.H.: Collective Dynamics of Small-World Networks. Nature 393 (1998) 440–442
15. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and Replication in Unstructured Peer-to-Peer Networks. ACM SIGMETRICS, Marina Del Rey, CA (2002)
16. Joseph, S.: P2P MetaData Search Layers. International Workshop on Agents and Peer-to-Peer Computing, Melbourne, Australia (2003)
17. Maymounkov, P., Mazi‘eres, D.: Kademlia: A Peer to Peer Information System Based on the XOR Metric. Internation Workshop on Peer-to-Peer Systems, Cambridge, MA (2002)
18. Waloszek, G.: Personal Networks. SAP AG, Product Design Center (2002)
19. Open Directory Categories.
20. Makosiej, P., Sakaryan, G., Unger, H.: Measurement Study of Shared Content and User Request Structure in Peer-to-Peer Gnutella Network. International Conference on Design, Analysis, and Simulation of Distributed Computing System, Arlington, Virginia (2004)
21. Pauli, C., Shepperd, M.: An Empirical Investigation into P2P File-Sharing User Behaviour. Americas Conference on Information Systems, Omaha, Nebraska (2005)
22. Crespo, A., Garcia-Molina, H.: Semantic Overlay Networks for P2P Systems. Stanford University (2002)
23. Distributed Computing Association White Paper. (2003)
24. Bhagwan, R., Savage, S., Voelker, G.M.: Understanding Availability. International Workshop on Peer-to-Peer Systems, Berkeley, CA (2003)
25. Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J.: Handling Churn in a DHT. USENIX Annual Technical Conference, Boston, MA (June 2004)
26. Yang, B., Garcia-Molina, H.: Efficient Search in Peer-to-Peer Networks. International Conference on Distributed Computing Sys-tems, Vienna, Austria (2002)
27. Vuong, S., Li, J.: Efa: an Efficient Content Routing Algorithm in Large Peer-to-Peer Overlay Networks. International Conference on Peer-to-Peer Computing, Linköping, Sweden (2003)
28. Saroiu, S., Gummadi, P.K., Gribble, S.D.: A Measurement Study of Peer-to-Peer File Sharing Systems. International Conference on Multimedia Networking and Computing, Santa Barbara, CA (2002)
29. Ren, Y., Sha, C., Qian, W., Zhou, A., Ooi, B.C., Tan, K.-L.: Explore the "Small World Phenomena" in Pure P2P Information Sharing Systems. International Symposium on Cluster Computing and the Grid, Tokyo, Japan (2003)
30. Koutrika, G., Ioannidis, Y.: Personalization of Queries Based on User Preferences. Preferences, Dagstuhl, Germany (2004)
31. Klemm, A., Lindemann, C., Vernon, M.K., Waldhorst, O.P.: Characterizing the Query Behavior in Peer-to-Peer File Sharing Systems. Internet Measurement Conference, Taormina, Sicily, Italy (2004)
32. Crespo, A., Garcia-Molina, H.: Routing Indices for Peer-to-Peer Systems. International Conference on Distributed Computing Systems, Vienna, Austria (2002)
33. Tsoumakos, D., Roussopoulos, N.: Adaptive Probabilistic Search for Peer-to-Peer Networks. Third International Conference on Peer-to-Peer Computing, Linköping, Sweden (2003)
34. Joseph, S.: NeuroGrid: Semantically Routing Queries in Peer-to-Peer Networks. International Workshop on Peer-to-Peer Computing, Pisa, Italy (2002)

Data Sharing in P2P Systems

Rabab Hayek, Guillaume Raschia, Patrick Valduriez, and Noureddine Mouaddib

Abstract In this chapter, we survey P2P data sharing systems. All along, we focus on the evolution from simple file-sharing systems, with limited functionalities, to Peer Data Management Systems (PDMS) that support advanced applications with more sophisticated data management techniques. Advanced P2P applications are dealing with semantically rich data (e.g., XML documents, relational tables), using a high-level SQL-like query language. We start our survey with an overview over the existing P2P network architectures, and the associated routing protocols. Then, we discuss data indexing techniques based on their distribution degree and the semantics they can capture from the underlying data. We also discuss schema management techniques which allow integrating heterogeneous data. We conclude by discussing the techniques proposed for processing complex queries (e.g., range and join queries). Complex query facilities are necessary for advanced applications which require a high level of search expressiveness. This last part shows the lack of querying techniques that allow for an *approximate query answering*.

Rabab Hayek
LINA, 2 rue de la Houssiniere, 44322, Nantes, France,
e-mail: `rabab.hayek@univ-nantes.fr`

Guillaume Raschia
LINA, 2 rue de la Houssiniere, 44322, Nantes, France,
e-mail: `guillaume.raschia@univ-nantes.fr`

Patrick Valduriez
INRIA, 2 rue de la Houssiniere, 44322, Nantes France,
e-mail: `Patrick.Valduriez@inria.fr`

Noureddine Mouaddib
LINA, 2 rue de la Houssiniere, 44322, Nantes France,
e-mail: `Noureddine.Mouaddib@univ-nantes.fr`

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_19, © Springer Science+Business Media, LLC 2010

1 Introduction

The recent years have witnessed a paradigm shift in the design of internet-scale distributed systems, with widespread proliferation of peer-to-peer technologies. Nowadays, the P2P model is used for diverse applications and services – including content storage and sharing (file-sharing, content distribution, backup storage) and communication (voice, instant messages, multicast) to name a few.

But, what is the P2P paradigm?

From the application perspective, the P2P paradigm is a way to leverage vast amounts of computing power, storage, and connectivity from personal computers distributed around the world [31]. Thus, the P2P model allows distributed systems to scale up on a world wide scale without the need for an expensive infrastructure, like the one it would be incurred by a client-server model.

From the system perspective, the P2P paradigm is about managing autonomous, unreliable resources that connect to the system in order to provide together the desired objectives, which that system is supposed to achieve. The management of such resources should be done without any global information or central control.

In other words, the P2P model overcomes the limitations of centralized and client-server models by introducing symmetry in roles, where each node is both a client and a server. But unlike Grid systems, P2P networks do not arise from the collaboration between established and connected groups of systems. Instead, they are characterized by ad hoc connections between autonomous and dynamic resources. Thus, P2P systems pose new challenges including resource discovery, reliability and availability.

In late 1999, P2P systems gained much attention with Napster's support for music sharing, and then have became a very interesting medium through which users share huge amount of data. Popular examples of P2P data sharing systems (e.g., Gnutella, KaZaa) report millions of users, sharing petabytes of data. However, a key challenge is implementing efficient techniques for search and data retrieval, without which an enormous shared data collection remains useless.

In a P2P data sharing system, users should be able to locate relevant data in a resource-efficient manner. Let us examine the generic architecture of a given peer, as shown by Fig. 1. Queries are submitted through a user interface. Then, they are handled by a data management layer which includes several techniques for supporting an efficient distributed query processing. This data management layer has been enriched all along the evolution of P2P systems from simple file-sharing systems with limited functionalities, to Peer Data Management Systems (PDMSs) which are dealing with semantically rich data.

In early P2P file sharing systems, the data management layer lacks several components. At a given peer, filename-based queries are blindly broadcasted in the network in order to locate the requested files. Besides, all the visited peers locally

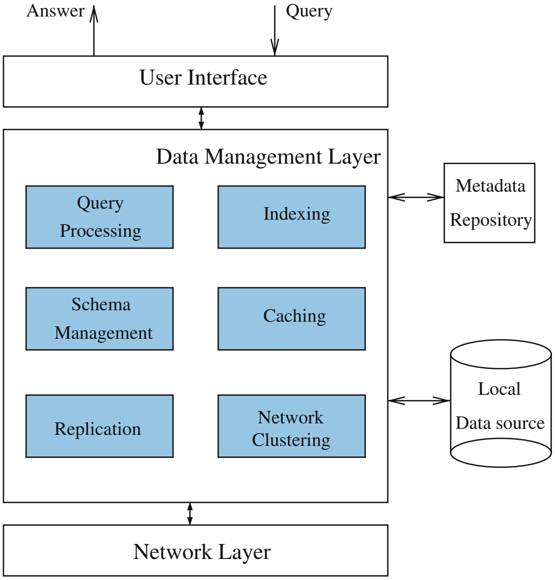


Fig. 1 Peer generic architecture

evaluate the received queries and return results, if any, to be finally merged at the requesting peer. The performance of these systems is quite dependent of the topology of the *underlying network*, and the associated routing protocol. This is discussed in Section 2.

To enhance the search performance, P2P works started to employ *data indexing* techniques (e.g., [6, 82]). At a given peer, the index allows to select a set of relevant peers to which the query is directly sent (i.e., location indexes), or to determine the direction through which relevant peers may be located (i.e., forwarding indexes). Section 3 discusses the P2P indexing schemes.

Parallel works have focused on data *replication* and *caching* techniques in order to improve the availability and the consistency of data, against the dynamic and autonomous nature of P2P systems. In this work, we do not detail these techniques, however good pointers can be found in [11, 15, 46].

In Schema-based P2P systems [93], each peer can provide its own database with its own schema, and may issue queries according to its local schema. In this case, irrespective of using data indexes, peers have to apply *schema management* techniques that provide a common ground for distributed query processing (e.g., [12, 45]). The basic idea is to identify content or structure similarities among peers. *Semantic mappings* are then defined to specify these similarities, and based on the semantic mapping definitions, queries are reformulated for each specific peer. Semantic mappings and other metadata are stored in a specific repository. The schema management techniques are presented in Section 4.

Network clustering has been also proposed as a viable solution to improve query processing in P2P systems (e.g., [7, 55]). Clustering techniques aim to organize the network into groups based on some criteria. A clustering criterion may be a physical network parameter (e.g., latency, bandwidth), peer property/behavior (e.g., connectivity, stability), or application-dependent parameter (e.g., similarity of interests). In this chapter, we do not discuss P2P clustering techniques. However, representative works are [7, 56, 81, 83].

Finally, we note that the data management layer presented in Fig. 1 may include additional components depending on other application requirements, such as trust and security.

All the above techniques have a common objective which is improving the efficiency of locating data. However, they should not restrict the search expressiveness. Certainly, the required level of search expressiveness is related to the data model used by the application. For instance, advanced P2P applications which are dealing with semantically rich data require a higher expressiveness level than key-based lookups or keyword searches. Processing complex queries in P2P systems is discussed in Section 5.

Note that throughout this thesis, the terms “node” and “peer” are used interchangeably to refer to the entities that are connected in a peer-to-peer network.

2 P2P Networks

P2P systems are application-level virtual networks with their own overlay topologies and routing protocols. The overlay topology defines how the nodes are connected to each other, while the routing protocol defines how nodes can exchange messages in order to share information and resources. The network topology and the associated routing protocol have significant influence on application properties such as performance, scalability, and reliability. P2P network overlays can be classified into two main categories: *unstructured* and *structured*, based on their structure. By “*structure*” we refer to the control on overlay creation and data placement.

2.1 Unstructured

Most popular P2P applications operate on unstructured networks. In these networks, peers connect in an ad-hoc fashion and the placement of content is completely unrelated to the overlay topology. Although P2P systems are supposed to operate in a fully decentralized manner (i.e., fully decentralized routing mechanisms), in practice, unstructured networks with various degrees of centralization are encountered. Accordingly, three categories can be identified.

2.1.1 Hybrid Decentralized Architectures

In these networks, a central server facilitates the interaction between nodes by indexing all their shared files (Fig. 2). Whenever a query is submitted, the central server is addressed to identify the nodes storing the requested files. Then, the file exchange may take place directly between two nodes. Certainly, this approach provides a very good search efficiency. However, the central server, which is a single point of failure, renders hybrid decentralized networks inherently unscalable and vulnerable to malicious attacks.

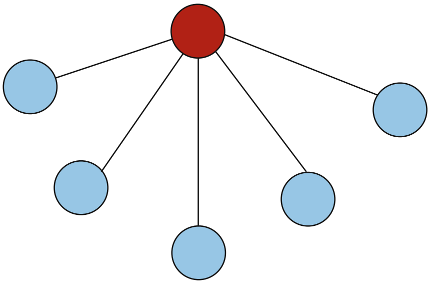


Fig. 2 Hybrid decentralized architecture

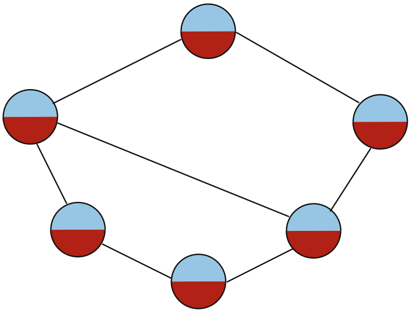


Fig.3 Pure decentralized architecture

The class of P2P systems relying on such hybrid architectures, i.e., including a server (e.g., red node) and peers (e.g., blue nodes), is usually called the first generation of P2P systems (1*GP*). A well-known example is Napster [4].

2.1.2 Pure Decentralized Architectures

In pure decentralized networks, there is a complete symmetry in node roles without any central coordination. Each node is both a client a server, i.e., each node may issue requests and serve/forward requests of other nodes (bi-colored nodes in Fig. 2). Hence, they exhibit high fault tolerance against node dynamicity and failure. However, resources are maintained locally and nodes have only limited knowledge. Thus, guarantees on lookup efficiency and content availability can not be provided. Here, search mechanisms range from brute flooding to more sophisticated mechanisms, such as random walks [11] and routing indices [6]. These mechanisms have direct implications on network scalability.

Representative examples of pure decentralized P2P systems are Gnutella [2], and FreeHaven [79].

2.1.3 Partially Decentralized Architectures

In these networks, there is a differentiation in roles between *supernode* and *leafnode*. Each supernode acts as a proxy for all its neighboring leaves by indexing their data and forwarding queries on their behalf. In practice, several supernodes are designated in the system to avoid all the problems associated to a single server (Fig. 4). Like pure decentralized P2P networks, the set of supernodes can be organized in a P2P fashion and communicate with one another in sophisticated ways. They are dynamically assigned and, if they fail, the network will automatically take action to replace them with others.

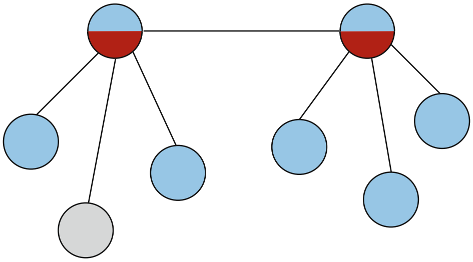


Fig. 4 Partially decentralized architecture

Examples of partially decentralized P2P systems are KaZaa [3], Gnutella2 [1], and Edutella [12]. Note that partially decentralized networks are also referred as *hierarchical* networks, while pure decentralized ones are referred as *flat* networks. Both categories represent the so-called, second P2P generation (2*GP*).

2.2 Structured

In an attempt to remedy the scalability problem of unstructured systems, some works have focused on introducing "structure" into network topologies. The topology overlay is tightly controlled, and the content may be distributed according to specific rules. These works led to the third generation of P2P systems (3*GP*), i.e., structured systems. Aiming basically to act as a decentralized index, structured overlays provide a mapping between content (e.g., file identifier) and location (e.g., node address), in the form of a distributed routing table.

Structured networks consist in partitioning a key space among peers, so that each peer is responsible for a specific key space partition, i.e., it should store all the resources (or pointers) which are mapped into keys, which are in the respective key-space partition. Then a routing algorithm is defined to allow a deterministic search based on key content. A representative class of structured overlays are the *Distributed Hash Tables DHT*s (e.g., [44, 82]). Freenet [43] is often qualified as a loosely structured system because the nodes of its P2P network can produce an estimate (not with certainty) of which node is most likely to store certain data. They use a chain mode propagation approach, where each node makes a local decision about which node to send the request message next.

Table 1 summarizes the P2P categories we outlined, with examples of P2P systems and infrastructures. P2P infrastructures do not constitute working applications, but provide P2P based-services (e.g., location and routing, anonymity, reputation management) and application frameworks. The infrastructures listed here are location and routing infrastructures. Note that according to the centralization criteria, all structured systems and infrastructures rely on pure decentralized topologies where all participants have equal roles.

Structure	Decentralization		
	Hybrid	*Partial*	*Full*
Unstructured	Napster Publius	KaZaa Morpheus Gnutella2 Edutella	Gnutella FreeHaven
Structured infrastructures			Chord CAN Trapestry Pastry
Structured systems			OceanStore Mnemosyne Scan, Past Kademlia

Table 1 A classification of P2P systems and infrastructures based on network structure, and degree of decentralization [13]

2.3 Unstructured vs. Structured: Competition or Complementarity?

An important question is: should the P2P overlay be “Structured” or “Unstructured”? Are the two approaches competing or complementary?

Some have considered unstructured and structured routing algorithms as competing alternatives. When generic key lookups are required, structured routing schemes guarantee locating relevant nodes within a bounded number of hops, based on strong theoretical foundations. The routing unstructured approaches, however, may have large costs or fail to find available data (in particular unpopular data). Despite of the lookup efficiency of structured overlays, several research groups are still leveraging unstructured P2P schemes. In fact, there are two main criticisms for structured systems [98]. First, the strict network structure imposes high overhead for handling node join and leave, although some works have defended performance during churn (e.g., [26]). Second, the lookup efficiency of these systems is limited to exact-match queries. Their ability to implement keyword searches and more complex queries is still an open issue. Therefore, given a P2P application, the best suited network overlay depends on that application functionalities and performance metrics.

Recently, some have started to justify that unstructured and structured approaches are complementary, not competing. The approach presented in [60] improves the unstructured Gnutella network by adding structural components. The motivation behind is that unstructured routing mechanisms are inefficient for data that are not highly replicated across the P2P network, while structured key-based lookup performs efficiently, irrespective of replication. In [21], the authors leverage the idea of cohabiting several P2P overlays on a same network, so that the best overlay could be chosen depending on the application. The distinctive feature of this proposal is that, in the *joint overlay*, the cohabiting overlays share information to reduce their maintenance cost while keeping the same level of performance.

Finally, we agree with the statement saying that the “unstructured vs. structured” taxonomy is becoming less useful, for two reasons. First, almost no network topologies are truly “unstructured”. Unstructured P2P proposals, which used initially blind flooding and random walks, have evolved to exploit inherent structure (e.g., small world and scale-free features), or to incorporate structure through clusters and superpeers. Second, a new class of *schema-based P2P* systems, also called Peer Data Management systems PDMSs, has emerged [93]. Examples of such systems combine approaches from P2P research as well as from the database and semantic web research areas. These systems allow the aggregation and integration of data from autonomous, distributed data sources. They are dealing with heterogeneity of nodes and structure within data.

Following this statement, some studies have adopted database taxonomy rather than networking taxonomy (e.g., [20, 39]) in order to categorize P2P search networks. The structure is implicitly determined by the type of the employed index. In the following section, we discuss the different data indexing schemes that have been proposed in the P2P literature.

3 Data Indexing in P2P Systems

P2P search techniques rely basically on data indexes. A data indexing scheme should take into account the following requirements. First, the creation/maintenance of indexes should not overload either the nodes by an extensive usage of their resources, or the network by a large bandwidth consumption. Second, the mechanism of maintaining indexes should not restrict peer autonomy. Instead, it should recover from node leave and join in a resource-efficient manner.

Obviously, the use of indexes should contribute to enhance the efficiency of searches made in the system. For instance, this efficiency can be quantified by the rate of successful searches (a search is *successful* if it locates, at least, one replica of the requested object), the response time, the number of returned results, the number of hops made to find a first query matching, and the number of messages exchanged in the network, which is an important metric from the system point of view.

The related trade-offs between index update cost, efficiency of the associated search technique, and peer churn are critical to evaluate a P2P indexing scheme.

3.1 Index Types

A P2P index can be local, centralized or distributed according to where it is maintained in the system, and to the distribution of data which it refers to.

3.1.1 Local Index

A node only keeps references to its own data, without obtaining any information about data stored at other nodes. The very early Gnutella design [2] adopted the *local-index* approach. This approach enables rich queries, but also generates huge traffic overhead since the query needs to be flooded widely in the network. Furthermore, any guarantees on search success can not be provided.

Considering that the key part of P2P searching approaches is an efficient routing mechanism, the local-index approaches can be seen as *index-free*, since they do not support query routing with any *forwarding* or *location* hints [97]. A forwarding index allows to reach the requested object within a varying number of hops (with the network size), while a location index allows to reach the target in a single hop. Based on the same reasoning, the search techniques that have been proposed to improve the performance of index-free systems, are referred as *blind* search techniques [86].

Breadth First Search (BFS)

The originally Gnutella algorithm uses flooding (BFS traversal of the underlying graph) for object discovery, and contacts all accessible nodes within a Time-To-Leave (TTL) value (Fig. 5). Small TTL values reduce the network traffic and the load at peers, but also reduce the chances of a successful search.

Modified BFS [89] is a variation of the BFS scheme in which the peers randomly choose only a ratio of their neighbors to forward the query to (Fig. 6). This approach reduces the number of messages needed for query routing at the cost of loosing available query answers, which might be found by the original BFS.

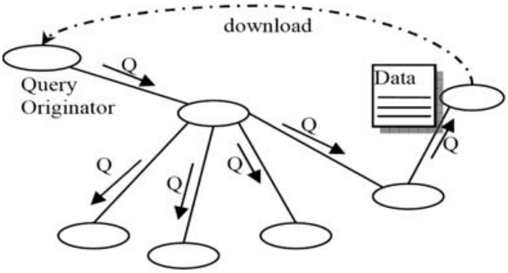


Fig. 5 Example of BFS: the received query is forwarded to all the neighbors

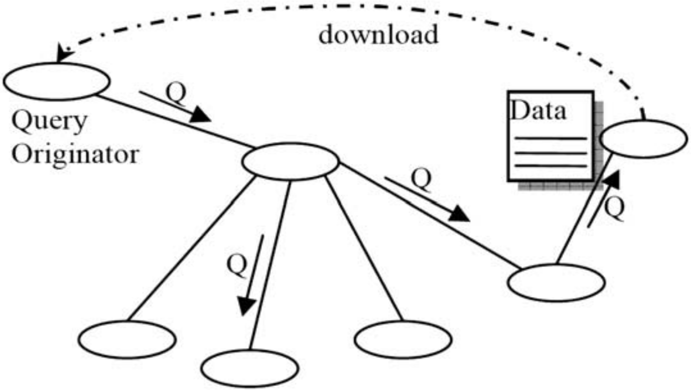


Fig. 6 Example of modified BFS: the received query is forwarded to a randomly selected set of neighbors

Iterative Deepening

In [11], the idea of iterative deepening has been borrowed from artificial intelligence and used in P2P searching. This method is also called *expanding ring*. The querying node periodically issues a sequence of BFS with increasing TTL values. The query terminates when sufficient number of results is found, or the predefined maximum value of TTL is reached. Iterative deepening is tailored to applications where the initial number of results found at peers that are closer to the query originator is important. In this case, it achieves good performance gains compared to the original BFS. In other cases, its overhead and response time may be much higher.

Random Walks

In the *standard random walk* algorithm, the querying node forwards the query message to one randomly chosen neighbor. This neighbor randomly selects one of its neighbors and forwards the query to that neighbor, and so on until there is a query match. This algorithm indeed reduces the network traffic, but massively increases the search latency.

In the *k-walker random walk* algorithm [11], the query is replicated at the originator, so it sends *k* query messages to an equal number of randomly chosen neighbors. Each of these messages follows its own path, having intermediate nodes forward it to a randomly chosen neighbor at each step. These query messages are also known as *walkers*. When the TTL of a walker reaches zero, it is discarded (Fig 7).

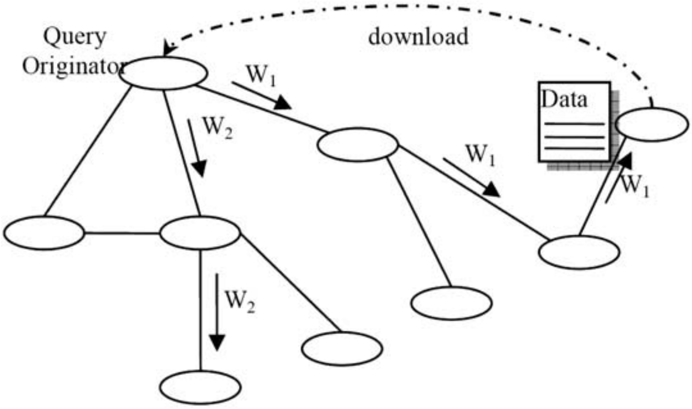


Fig. 7 Example of random walks: each received walk is forwarded to only one neighbor

The algorithm's most important advantage is the significant message reduction it achieves. It produces $k \cdot TTL$ messages in the worst case, a number which seldom depends on the underlying network. It also achieves some kind of local "load balancing", since no nodes are favored in the forwarding process over others. However, the most serious problem of this algorithm is its highly variable performance. Success rates and number of hits vary greatly depending on network topology and the random choices made. Another drawback is its inability to adapt to different query loads.

Adamic et al. [50] addressed the first problem of random walks by recommending that instead of using purely random walks, the search protocol should bias its walks toward high-degree nodes (i.e., nodes with large number of connections). They assume that high-degree nodes are also capable of higher query throughputs. Certainly, the relevance of such assumption is constrained by the design of balancing rules to avoid overloading high-degree nodes, which may not have the capacity to handle a large number of queries.

Finally we note that, in spite of their name, the *Local Indices* proposed in [24] do not belong to this type of indexes. An index is locally maintained at a given node, however, it refers to remote data stored at other nodes.

3.1.2 Centralized Index

The index is centralized at dedicated servers, but the described data is distributed. In fact, the centralized schemes [4] were the first to demonstrate the P2P scalability that comes from separating data index from the data itself. The centralized index is a location (non-forwarding) index that allows to locate relevant data within one hop, which is very efficient. However, the central servers are single points of failure which renders the system inherently unscalable and vulnerable to malicious attack.

The P2P research community has rapidly turned its back on centralized architectures. Furthermore, P2P systems that only use local indexes are becoming rare, since routing the query in a blind manner is still providing a poor trade-off between the traffic overhead and the lookup efficiency. In practice, all current P2P systems are implementing distributed indexes.

3.1.3 Distributed Index

The index refers to data from distributed sources, and is itself distributed across the network. Here, we are talking about the global index, which is (may be virtually) obtained from the set of indexes materialized in the network. A hybrid decentralized approach consists in distributing such global index among some specialized nodes (e.g., supernodes and ultrapeers). A pure decentralized approach distributes the index among all participants, that is, each node in the system maintains a part of that index.

An early P2P proposal for a distributed index was Freenet [43]. Freenet uses a hash function to generate keys, by which the shared files are identified. Each node maintains a dynamic routing table containing the addresses of other nodes and the file keys they are thought to hold. To search for a file, the user sends a request message specifying the *key* and a TTL value. Upon receiving a query message, a node checks its local table for either a match or another node with keys close to the target. If the file is eventually found at a certain node (before exceeding TTL), the query response traverses the successful query path in reverse, adding a new routing table entry (the requested key and the file provider) at each peer. A subsequent request with the same key will be served with this cached entry. The request will be forwarded directly to the node that had previously provided the data. Freenet allows to significantly reduce the traffic overhead in the system. However, it only supports exact-match queries, and only one result is returned. Another limitation is that Freenet takes time to build an efficient index upon the arrival of a new node.

As said before, almost all of the current P2P proposals rely on distributed indexes, which can range from simple forwarding hints to exact object locations. These indexes can be distinguished according to whether they are semantic-free, or they capture data semantics. The semantic index is human-readable. For example, it might associate information with keywords, document names, or database keys. A free-semantic index typically corresponds to the index by a hash mechanism, i.e., the DHT schemes.

3.2 Semantic-Free Index: DHT

Structured systems have emerged mainly in an attempt to address that scalability problem of Gnutella-like systems. They use the Distributed Hash Table (DHT) as a substrate, in which the overlay topology and the data placement are tightly controlled.

Various DHT schemes differ in the topologies, routing protocols, fault tolerance, and resilience to churn. In the following, we present the main geometries (i.e., the topology and the associated routing strategies) used for DHT-based systems, and discuss their search efficiency and their robustness.

3.2.1 Tree

Tree is the first geometry which is used for organizing the peers of a DHT and routing queries among them. In this approach, nodes and objects are assigned unique identifiers (e.g., 160-bit key). The leaf nodes of the binary tree represent the key-space partitions (peer's identifiers). The depth of that tree is $log(n)$, where n is the number of peers. The responsible for a given object key is the peer whose identifier has the highest number of prefix bits which are common with the key. A search is routed toward the requested object based on longest prefix matching at each intermediate peer until reaching the responsible peer. The distance between two peers is then the height of the smallest common subtree. Tapestry [25] uses similar prefix matching in order to forward query messages. To avoid the problem of single

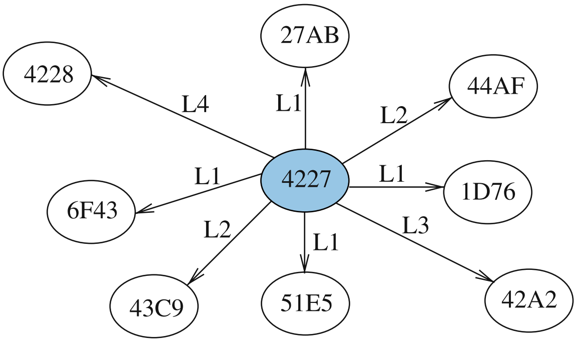


Fig. 8 Tapestry routing mesh from the perspective of a single node. Outgoing neighbor links point to nodes with a common matching prefix. Higher level entries match more digits. Together, these links form the neighbor map [25]

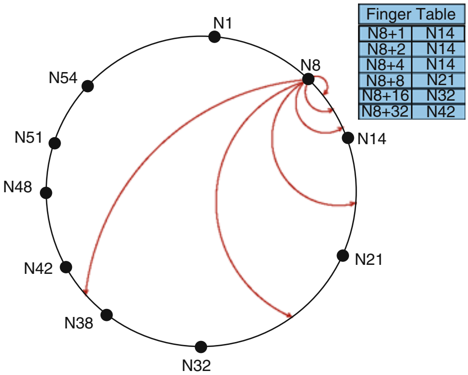
point of failure that root nodes constitute in the Plaxton Tree model, Tapestry assigns multiple roots to each object. Such approach allows reliability at the cost of redundancy.

For each level in a tree topology there are several choices to select routing table entries. To illustrate, each Tapestry node maintains a neighbor map as shown in Fig. 8. The neighbor map has multiple levels, each level l containing pointers to nodes whose identifier must be matched with l bits. For instance, the node in Fig. 8 maintains at the third level of its routing table one pointer to one node matching his identifier with 3 digits.

The tree geometry has good neighbor selection flexibility, i.e., each peer has $2^i - 1$ options in choosing a neighbor at a level i. However, it has no flexibility for message routing: there is only one neighbor which the message must be forwarded to, i.e., this is the neighbor that has the most common prefix bits with the given key. Several applications have been designed on the top of Tapestry, such as OceanStore [47]. Pastry [14] is a scheme similar to Tapestry, however, it differs in the approach to achieving network locality and object replication. It is employed by the PAST large-scale persistent P2P storage utility [15].

3.2.2 Ring

The Ring geometry is based on a one dimensional cyclic space such that the peers are ordered on the circle clockwise with respect to their keys. Chord [44] is the prototypical DHT ring. Chord supports one main operation: find a peer with the given *key*. The keys are assigned both to data and peers by means of a variant of Consistent Hashing [48]. Each key on the key-space is mapped to the peer with the least identifier greater or equal to the key, and this peer is called the key's *successor*. Thus to say, this peer is responsible for the corresponding data. The use of consistent hashing tends to balance load, as each node receives roughly the same number of keys.



Finger Table	
N8+1	N14
N8+2	N14
N8+4	N14
N8+8	N21
N8+16	N32
N8+32	N42

Fig. 9 The *finger table* at node 8 on a Chord ring of 10 nodes, $m = 6$ [44]

In Chord, a peer needs to track the addresses of only m other peers, not all peers such in the original Consistent Hashing proposal. Each peer p maintains a *"finger table"* containing $m = log(n)$ entries such that the ith entry provides the address of the peer whose distance from p clockwise in the circle is $2^i - 1$ *mod n* (see Fig. 9). Hence, any peer can route a given key to its responsible in $logn$ hops because each hop reduces the distance to the destination by half. In Chord, a peer needs to track the addresses of only $m = O(logn)$ other peers, not all peers such as in the original Consistent Hashing proposal.

The correctness of the Chord routing protocol relies on the fact that each peer is aware of its successors. When peers fail, it is possible that a peer does not know its new successor, and that it has no chance to learn about it. To avoid this situation, peers maintain a successor list of size r, which contains the peer's first r successors. When the successor peer does not respond, the peer simply contacts the next peer on its list.

3.2.3 Hypercube

The Hypercube geometry is based on partitioning a d-dimensional space into a set of separate zones and attributing each zone to one peer. Peers have unique identifiers with $log\ n$ bits, where n is the total number of peers Each peer p has $log\ n$ neighbors such that the identifier of the ith neighbor and p differ only in the ith bit. Thus, there is only one different bit between the identifier of p and each of its neighbors. The distance between two peers is the number of bits on which their identifiers differ. Query routing proceeds by greedily forwarding the given key via intermediate peers to the peer that has minimum bit difference with the key. Thus, it is somehow similar to routing on the tree. The difference is that the hypercube allows bit differences to be reduced in any order while with the tree bit differences have to be reduced in strictly left-to-right order.

The number of options for selecting a route between two peers with k bit differences is $(log\ n) \cdot (log\ n - 1) \cdot \cdots \cdot (log\ n - k)$, i.e., the first peer on the route has $log\ n$ choices, and each next peer on the route has one choice less than its predecessor. Thus, in the hypercube, there is great flexibility for route selection. However, each node in the coordinate space does not have any choice over its neighbors coordinates since adjacent coordinate zones in the coordinate space can not change. The high selection flexibility offered by the Hypercube is at the price of poor neighbor selection flexibility.

The routing geometry used in CAN [82] resembles a hypercube geometry. CAN uses a d-dimensional coordinate space which is partitioned into n zones and each zone is occupied by one peer (see Fig. 10). When $d = log\ n$, the neighbor sets in CAN are similar to those of a $log\ n$ dimensional hypercube.

Other DHTs geometries are the *Butterfly* geometry which is used in Viceroy [30], and the XOR geometry which is used by Kademlia [69]. Certainly, two or more geometries can be combined together to provide a hybrid geometry that satisfies better the DHT requirements. To illustrate, Pastry [14] combines the tree and ring geometries in order to achieve more efficiency and flexibility.

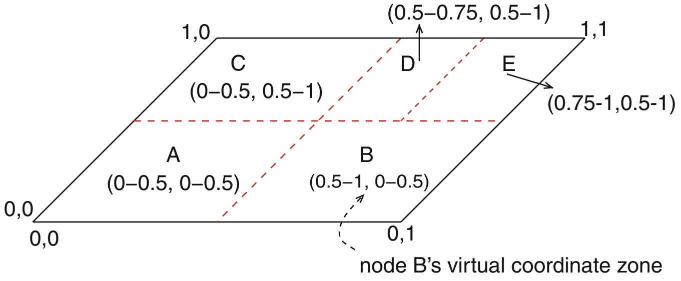


Fig. 10 2-dimensional $[0;1] \times [0;1]$ coordinate space partitioned between 5 CAN nodes [82]

3.3 Semantic Index

The initial unstructured file sharing P2P systems offered a filename-based search facility, while the DHT-based systems offered only a key-based lookup. However, as stated before, the P2P systems should be able to do more than "finding" things, i.e., to capture data semantics and to allow for rich, complex queries. Works on both P2P networks, unstructured and structured, have been started in order to support P2P applications with higher levels of search expressiveness. First enhancements to existing file sharing P2P systems have early provided keyword search facilities. Later, providing large-scale Information Retrieval (IR), e.g., for searching the world wide web, becomes an appealing application for P2P networks. Consequently, the well known IR techniques have been brought into the context of P2P networks, in order to support a decentralized document management (e.g., storing, clustering, indexing) and retrieval.

Recently, the Database and P2P paradigm have meet. The former was slowly moving toward a higher degree of distribution, and thus requiring a new class of scalable, distributed architecture. The latter has started to explore more expressiveness infrastructures in order to extend the representation and query functionalities it can offer to advanced applications. The P2P Data Management Systems (PDMS) are the point where the two paradigms meet.

As the P2P networks are going to be adaptable, i.e., to support a wide range of applications, they need to accommodate many search types. Index engineering has been always at the heart of P2P search methods. In the following, we introduce the various types of semantic indexes employed by current P2P systems. Then, the query capabilities will be discussed in Section 5.

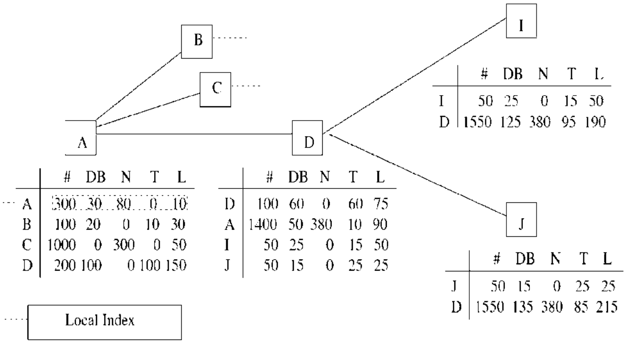
3.3.1 Keyword Lookup

Gnutella [2] provides a simple keyword match. Queries contain a string of keywords and peers answer when they have files whose names contain all that keywords. In its first version, Gnutella was a local-index system. Queries were flooded in the entire network and peers only used their local indexes for filename matches.

As a way to improve the performance of unstructured Gnutella-like systems, the notion of *ultrapeer* was introduced, so that the peer are organized into a hierarchical network overlay. In [16], each peer maintains an index of filename keywords, called the Query Routing Table (QRT), and forwards it to its ultrapeer. Upon receiving a query, the latter sends the query only to leaves which have a match based on their QRTs. Later, there has been a proposal to exploit the network hierarchy in order to build a hierarchical index. Aggregated QRTs are distributed amongst the ultrapeers to improve the query forwarding from an ultrapeer to another.

In other approach, [24] suggested the *local indices*: data structures where each node maintains an index of the data stored at nodes located within a radius r from itself. The query routing is done in a BFS-like way, except that the query is processed only at the peers that are at certain hop distances from the query originator. To minimize the overhead, the hop distance between two consecutive peers that process the query must be $2 \cdot r + 1$. In other words, the query must be processed at peers whose distance from the query originator is $m \cdot (2 \cdot r + 1)$ for $m = 1, 2, \ldots$. This allows querying all data without any overlap. The processing time of this approach is less than that of standard BFS because only a certain number of peers process the query. However, the number of routing messages is comparable to that of standard BFS. In addition, whenever a peer joins/leaves the network or updates its shared data, a flooding with $TTL = r$ is needed in order to update the peers' indices, so the overhead becomes very significant for highly dynamic environments.

Routing Indices [6] have been proposed to support query routing with information about "direction" towards data, rather than providing its actual location. Documents are assumed to fall into a number of topics, and queries request documents on particular topics. Routing Indices (RIs) store information about the approximate number of documents from every topic that can be retrieved through each outgoing link (i.e., not only from that neighbor but from all nodes accessible from it).



	#	DB	N	T	L
A	300	30	80	0	10
B	100	20	0	10	30
C	1000	0	300	0	50
D	200	100	0	100	150

	#	DB	N	T	L
D	100	60	0	60	75
A	1400	50	380	10	90
I	50	25	0	15	50
J	50	15	0	25	25

	#	DB	N	T	L
I	50	25	0	15	50
D	1550	125	380	95	190

	#	DB	N	T	L
J	50	15	0	25	25
D	1550	135	380	85	215

Fig. 11 Example of routing indices [6]

Figure 11 shows an example of a P2P network with RIs built over four topics of interest. The first row of each RI contains the summary of the local index presented before (i.e., radius $r = 2$). In particular, the summary of A's local index shows that A

has 300 documents: 30 about databases, 80 about networks, none about theory, and 10 about languages. The rest of the rows represent a compound RI. In the example, the RI shows that node *A* can access 100 database documents through *D* (60 in *D*, 25 in *I*, and 15 in *J*).

Given a query, the termination condition relates to a minimum number of hits. A node that can not satisfy the query stop condition with its local repository will forward it to the neighbor with the highest "goodness" value. Three different functions which rank the out-links according to the expected number of documents that could be discovered through them are proposed. The routing algorithm backtracks if more results are needed. A limitation of this approach is that RIs require flooding in order to be created and updated, so they are not suitable for highly dynamic networks. Moreover, stored indices can be inaccurate due to topic correlations, over-counts or under-counts in document partitioning and network cycles.

3.3.2 Peer Information Retrieval

The amount of data published in the internet and its amazing growth rate become beyond centralized web search engines. Recently, P2P systems start to represent an interesting alternative to build large-scale, decentralized Information Retrieval systems.

IR systems define representations of both documents and queries. They may only support a boolean retrieval model, in which documents are indexed and a document can match or not a given query. Note that the local and routing indices described in the above section allow for such a retrieval model. Current IR systems are supporting the retrieval model with a ranking function that quantifies the order amongst the documents matching the query. This becomes essential in the context of large document collections, where the resulting number of matching documents can far exceed the number a user could possibly require. To this end, the IR system defines relationships between document and query representations, so that a score can be computed for each matching document, w.r.t. the query at hand.

A P2P system differs from a distributed IR system in that it is typically larger, more dynamic with node lifetimes measured in hours. Furthermore, a P2P system lacks the centralized mediators found in many IR systems that assume the responsibility for selecting document collections, rewriting queries, and merging ranked results [18]. In the following, we first introduce the main IR techniques used for indexing documents. Then, we present the P2P IR systems that have been proposed in the literature.

Inverted Index

The inverted index, or sometimes called inverted file, has became the standard technique in IR. For each term, a list that records which documents the term occurs in is maintained. Each item in the list is conventionally called a *posting*. The list is then called a *postings list* (or inverted list), postings list and all the postings lists taken together are referred to as the postings.

Vector Space Model

The representation of the set of documents and queries as vectors in a common vector space is known as the Vector Space Model (VSM) and is fundamental to support the operation of scoring documents relative to a query. Each component of the vector represents the importance of a *term* in the document or query. The weight of a component is often computed using the *Term Frequency * Inverse Document Frequency* (TF*IDF) scheme.

- *Term Frequency*: the frequency of each term in each document.
- *Inverse Document Frequency*: the document frequency df_t is the number of documents, in a collection of *N* documents, that contain a term t. The Inverse Document Frequency of term *t* is given by: $log(N/df_t)$.

Viewing a collection of *N* documents as a collection of vectors leads to a natural view of a collection as a *term-document matrix*: this is an $M \times N$ matrix whose rows represent the *M* terms (dimensions) of the *N* columns, each of which corresponds to a document.

Latent Semantic Index

Latent Semantic Index (LSI) uses Singular Value Decomposition (SVD) to transform and truncate the term-document matrix computed from VSM. This allow to discover the semantics underlying terms and documents. Intuitively, LSI transforms a high-dimensional document vector into a medium-dimensional semantic vector by projecting the former into a medium-dimensional semantic subspace. The basis of the semantic subspace is computed using SVD. Semantic vectors are normalized and their similarities are measured as in VSM.

Several solutions for text-based retrieval in decentralized environments have been proposed in the literature.

PlanetP [33] is a publish-subscribe service for P2P communities, supporting content ranking search. PlanetP maintains a detailed inverted index describing all documents published by a peer locally (i.e., a local index). In addition, it uses gossiping to replicate a *term-to-peer* index everywhere for communal search and retrieval. This term-to-peer index contains a mapping $t \rightarrow p$ if term *t* is in the local index of peer *p*. PlanetP approximate $TF \cdot IDF$ by dividing the ranking problem into two stages. In first, peers are ranked according to their likehood of having relevant documents. To this end, PlanetP introduces the *Inverse Peer Frequency* (IPF) measure. Similar to IDF, the idea behind is that a term is of less importance if it is present in the index of every peer. Second, PlanetP contacts only the first group of *m* peers from the top of the peer ranked list, to retrieve a relevant set of documents. It stops contacting peers when the top-k document ranking becomes stable, where k is specified by the user. A primary shortcoming of PlanetP is the large amount of metadata that should be maintained, which restricts its scalability.

The PeerSearch system [28] proposes another approach that places documents onto a DHT network according to their semantic vectors produced by Latent Semantic Indexing (LSI) in order to reduce document dimensionality and guarantee solution scalability. However, as semantic vectors have to be defined a priori, the method cannot efficiently handle dynamic scenarios and adapt to changing collections.

A query-driven indexing method has recently been proposed in [37]. However, the solution is based on single-term indexing and does not consider indexing with term combinations. A recent work proposes the AlvisP2P search engine [85], which enables retrieval with multi-keywords from a global document collection available in the P2P network. One of the merits of the proposed approach is that indexing is performed in parallel with retrieval. However, a main limitation is that the quality of the answer obtained for a given query depends on the popularity of the term combinations it contains.

3.3.3 Peer Data Management

While existing architectures for distributed systems have been reaching their maturity (e.g., distributed database systems, data integration systems), the P2P paradigm has emerged as a promising alternative to provide a large-scale decentralized infrastructure for resource sharing. Grible et al. have addressed an important question *how data management can be applied to P2P, and what the database community can learn from and contribute to the P2P area?* [36]. The P2P paradigm has gained much popularity with the first successful file sharing systems (e.g., Gnutella, KaZaa) because of the ease of deployment, and the amplification of the desired system properties as new nodes join (i.e., this is aligned with the definition of the P2P paradigm). However, the semantics provided by these systems is typically weak. So far in this report, we have reviewed P2P systems that support key lookups or keyword search. In order to support advanced applications which are dealing with structured and semantically rich data, P2P systems must provide more sophisticated data access techniques. The overlapping of P2P and database areas has lead to a new class of P2P systems, called Peer Data Management Systems (PDMS) or schema-based P2P systems (see Fig. 12).

In distributed databases, the location of content is generally known, the query optimizations are performed under a central coordination, and answers to queries are expected to be complete. On the other side, the ad-hoc and dynamic membership of participants in P2P systems makes difficult to predict about the location and the quality of resources, and to maintain globally accessible indexes which may become prohibitive as the network size grows.

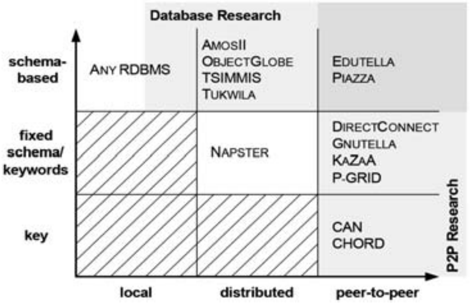


Fig. 12 Schema capabilities and distribution [93]

The work that has been done in PDMSs mainly addresses the information integration issue. In fact, the potential heterogeneity of data schemas makes sharing structured data in P2P systems quite challenging. This issue will be discussed in Section 4. Besides, PDMSs have started to study the design and the implementation of complex query facilities (e.g., join and range queries). This is a fundamental building block of a given PDMS which attempts to be a fully distributed data system, with a high level of query expressiveness. Processing complex queries requires the employment of data access techniques which deal with the structure and semantics within data. Section 5 discusses complex queries in P2P systems.

4 Schema Management in P2P Systems

Semantic heterogeneity is a key problem in large scale data sharing systems [57]. The data sources involved are typically designed independently, and hence use different schemas. To be able to allow meaningful inter-operation between different data sources, the system needs to define schema mappings. *Schema mappings* define the semantic equivalence between relations and attributes in two or more different schemas.

The traditional approach for querying heterogeneous data sources relies on the definition of *mediated schema* between data sources [38] (see Fig. 13). This mediated schema provides a global unified schema for the data in the system. Users submit their queries in terms of the mediated schema, and schema mappings between the mediated schema and the local schemas allow the original query to be reformulated into subqueries executable at the local schemas. There is a wrapper close to each data source that provides translation services between the mediated schema and the local query language [88].

In data integration systems, there are two main approaches for defining the mappings: Global-as-view (GAV) which defines the mediated schema as a view of the local schemas, and Local-as-View (LAV) which describes the local schemas as a view of the mediated schema [54]. In GAV, the autonomy of data sources is higher

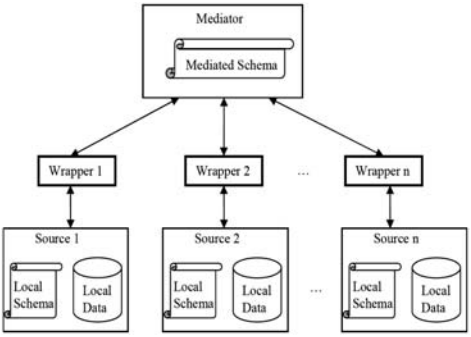


Fig. 13 Schema mapping using a global mediated schema

than LAV because they can define their local schemas as they want. However, if any new source is added to a system that uses the GAV approach, considerable effort may be necessary to update the mediator code. Thus, GAV should be favored when the sources are not likely to change. The advantage of a LAV modeling is that new sources can be added with far less work than in GAV. LAV should be favored when the mediated schema is not likely to change, i.e., the mediated schema is complete enough that all the local schemas can be described as a view of it.

Given the dynamic and autonomous nature of P2P systems, the definition of a unique global mediated schema is impractical. Thus, the main problem is to support decentralized schema mapping so that a query on one peer's schema can be reformulated in a query on another peer's schema. The approaches which are used by P2P systems for defining and creating the mappings between peers' schemas can be classified as follows: pairwise schema mapping, mapping based on machine learning techniques, common agreement mapping, and schema mapping using IR techniques.

4.1 Pairwise Schema Mappings

In this approach, the users define the mapping between their local schemas and the schema of any other schema which is interesting for them. Relying on the transitivity of the defined mappings, the system tries to extract mappings between schemas which have no defined mapping.

Piazza [45] follows this approach (see Fig.14). In Piazza, the data are shared as XML documents, and each peer has a schema, expressed in XMLSchema, which defines the terminology and the structural constraints of the peer. When a new peer (with a new schema) joins the system for the first time, it maps its schema to the schema of some other peers of the network. Each mapping definition begins with an XML template that matches some path or sub-tree of an instance of the target schema, i.e., a prefix of a legal string in the target DTD's grammar. Elements in the template may be annotated with query expressions (in a subset of XQuery) that bind variables to XML nodes in the source.

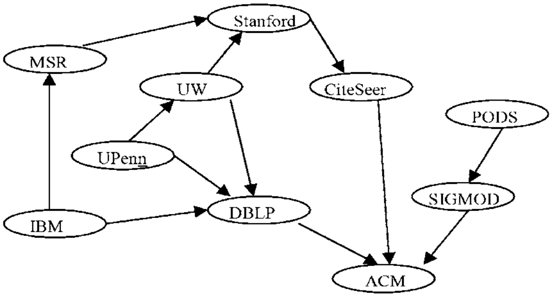


Fig. 14 An example of pairwise schema mapping in piazza

The Local Relational Model (LRM) [66] is another example that follows this approach. LRM assumes that the peers hold relational databases, and each peer knows a set of peers with which it can exchange data and services. This set of peers is called *p*'s *acquaintances*. Each peer must define semantic dependencies and translation rules between its data and the data shared by each of its acquaintances. The defined mappings form a semantic network, which is used for query reformulation in the P2P system.

PGrid also assumes the existence of pairwise mappings between peers, initially constructed by skilled experts [10]. Relying on the transitivity of these mappings and using a gossiping algorithm, PGrid extracts new mappings that relate the schemas of the peers between which there is no predefined schema mapping.

4.2 Mapping Based on Machine Learning Techniques

This approach is usually used when the shared data is defined based on ontologies and taxonomies as proposed in the Semantic Web [5]. It uses machine learning techniques to automatically extract the mappings between the shared schemas. The extracted mappings are stored over the network, in order to be used for processing future queries.

GLUE [8] uses this approach. Given two ontologies, for each concept in one, GLUE finds the most similar concept in the other. It gives well founded probabilistic definitions to several practical similarity measures. It uses multiple learning strategies, each of which exploits a different type of information either in the data instances or in the taxonomic structure of the ontologies. To further improve mapping accuracy, GLUE incorporates commonsense knowledge and domain constraints into the schema mapping process. The basic idea is to provide classifiers for the concepts. To decide the similarity between two concepts A and B, the data of concept B is classified using A's classifier and vice versa. The amount of values that can be successfully classified into A and B represent the similarity between A and B.

4.3 Common Agreement Mapping

In this approach, the peers that have a common interest agree on a common schema description for data sharing. The common schema is usually prepared and maintained by expert users. APPA [76] makes the assumption that peers wishing to cooperate, e.g., for the duration of an experiment, agree on a Common Schema Description (CSD). Given a CSD, a peer schema can be specified using views. This is similar to the LAV approach in data integration systems, except that, in APPA, queries at a peer are expressed in terms of the local views, not the CSD. Another difference between this approach and LAV is that the CSD is not a global schema, i.e., it is common to a limited set of peers with common interest (see Fig. 15). Thus, the CSD makes no problem for the scalability of the system. When a peer decides to share data, it needs to map its local schema to the CSD. In APPA, the mappings between the CSD and each peer's local schema are stored locally at the peer. Given a query Q on the local schema, the peer reformulates Q to a query on the CSD using locally stored mappings.

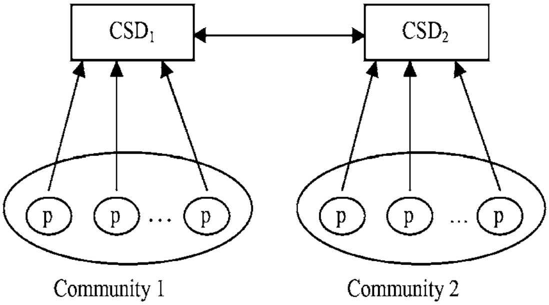


Fig. 15 Common agreement schema mapping in APPA

AutoMed [70] is another system that relies on common agreements for schema mapping. It defines the mappings by using primitive bidirectional transformations defined in terms of a low-level data model.

4.4 Schema Mapping Using IR Techniques

This approach extracts the schema mappings at query execution time using IR techniques by exploring the schema descriptions provided by users. PeerDB [62] follows this approach for query processing in unstructured P2P networks. For each relation which is shared by a peer, the description of the relation and its attributes is maintained at that peer. The descriptions are provided by users upon creation of relations, and serve as a kind of synonymous names of relation names and attributes. When a query is issued, some agents are flooded to the peers to find out potential matches and bring the corresponding meta-data back. By matching keywords from the metadata of the relations, PeerDB is able to find relations that are potentially similar to the query relations. The found relations are presented to the user who has issued the query, and she decides on whether or not to proceed with the execution of the query at the remote peer which owns the relations.

Edutella [12] also follows this approach for schema mapping in super-peer networks. Resources in the Edutella are described using the RDF metadata model, and the descriptions are stored at superpeers. When a user issues a query at a peer p, the query is sent to p's super-peer where the stored schema descriptions are explored and the address of the relevant peers are returned to the user. If the super-peer does not find relevant peers, it sends the query to other super-peers such that they search relevant peers by exploring their stored schema descriptions. In order to explore stored schemas, super-peers use the RDF-QEL query language. RDF-QEL is based on Datalog semantics and thus compatible with all existing query languages, supporting query functionalities which extend the usual relational query languages.

Independently of the approach used to implement the schema mappings, P2P systems attempt to exploit the transitive relationships among peer schemas to perform data sharing and integration [99]. While in traditional distributed systems, schema mappings form a semantic tree, in P2P systems the mappings form a *semantic graph*. By traversing semantic paths of mappings, a query over one peer can obtain relevant data from any reachable peer in the network. Semantic paths are traversed by reformulating queries at a peer into queries on its neighbors.

5 Querying in P2P Systems

The support of a wider range of P2P applications motivates the evolution of current P2P technologies in order to accommodate many search types. As said before, a P2P system should support the operating application with an appropriate level of query expressiveness. Advanced applications which are dealing with semantically rich data require an expressiveness level higher than filename-based or key-based lookup. In the following, we discuss the different techniques used for processing complex queries in P2P systems. These querying techniques can be distinguished according to:

- *Search completeness:* the network is entirely covered by the search mechanism. Relaxing the search completeness leads to *partial lookup*.
- *Result completeness:* the found result set is entirely returned to the user. Relaxing the result completeness leads to *partial answering*.
- *Result granularity:* generally, the returned results are retrieved from, and thus have the same type as, the original queried data (e.g., music files, XML documents, database tuples). Returning results at a different level of granularity (by making data abstraction) leads to *approximate answering*. The term "approximate" may still be ambiguous, due to its wide employment in query processing proposals. However, the following sections tend to give a precise definition of what we are referring to by "approximate answers".

5.1 Partial Lookup

The advantages of P2P data sharing systems, like scalability and decentralization, do not come for free. In large-scale dynamic systems, it is nearly impossible to guarantee a complete search. Let Q be a query issued by a peer p in the system, and P_Q the set of relevant peers, i.e., the peers that store, at least, one query result. Q is said to be a *total-lookup* query if it requires *all* the results available in the system. Here, the set P_Q should be entirely visited. In the case where Q requires *any k* results, Q is said to be a *partial-lookup* query.

The impracticality of an exhaustive flooding, the limited knowledge provided by indexes and the errors they may contain, and the incorrect semantic mappings are reasons among others for considering that all queries in P2P systems are in reality processed as being partial-lookup queries. In other terms, the query Q issued by peer p in a P2P network of size N will be routed in a subnetwork of size N', and thus a subset $P'_Q \subseteq P_Q$ can be targeted. The filename-based, key-based and keyword-based searches have been presented earlier in this chapter. Here, we discuss three types of complex queries: range, multi-attribute and join queries. The importance of these queries has been recognized in many distributed environments (e.g., parallel databases, Grid resource discovery) since they significantly enhance the application ability to precisely express its interests.

5.1.1 Range Queries

Range queries are issued by users to find all the attribute values in a certain range over the stored data.

Several systems have been proposed to support range queries in P2P networks. The query processing in these systems rely on underlying DHTs, or other indexing structures. Some argue that DHTs are not suited to range queries [9]. The hash functions used to map data on peers achieve good load balancing, but do not maintain data proximity, i.e., the hash of two close data may be two far numbers. Despite of this potential shortcoming, there have been some range query proposals based on DHTs.

Instead of using uniform-hashing techniques, Gupta et al. [9] employ locality sensitive hashing to ensure that, with high probability, similar ranges are mapped to the same peer. They propose a family of locality sensitive hash functions, called min-wise independent permutations. The simulation results show good performance of the solution. However, there is the problem of load unbalance for large networks. In [64] the authors extend the CAN protocol using the Hilbert space-filling curve and load balancing mechanisms. Nearby ranges map to nearby CAN zones, and if a range is split into two sub-ranges, then the zones of the sub-ranges partition the zone of the primary range. Thus, the one-dimensional space of data items is mapped to the multi-dimensional CAN zones. Conversely, multi-dimensional data items are mapped to data points in one-dimensional space through the space-filling curve in [27]. Such a construction gives the ability to search across multiple attributes.

Some works rely on Skip list data structure which, unlike DHT, does not require randomizing hash functions and thus can support range queries. SkipNet [63] is a lexicographic order-preserving DHT that allows data items with similar values to be placed on contiguous peers. It uses names rather than hashed identifiers to order peers in the overlay network, and each peer is responsible for a range of strings. This facilitates the execution of range queries. However, it is not efficient because the number of peers to be visited is linear in the query range.

Other proposals for range queries avoid both DHT and Skip list structures. P-Grid [10] is based on a randomized binary prefix tree. One limitation is that P-Grid considers that all nodes in the system have a fixed capacity, and content is heuristically replicated to fill all the node capacity. However, there is no formal characterization of either the imbalance ratio guaranteed, or the data-movement cost incurred.

BATON [40] is a balanced binary search tree with in-level links for efficiency, fault-tolerance, and load-balancing. VBI-tree [41] proposes a virtual binary overlay which is an enhancement of BATON, and focuses on employing multi-dimensional indexes to support more complex range query processing. Common problems to balanced tree overlay structures is that peer joining or leaving can cause a tree structural change, and the update strategies may get prohibitive under a high churn environment.

5.1.2 Join Queries

Distributed data among peers could be seen, in some cases, as a set of large relational tables fragmented horizontally. Running efficient join queries over such massively dispersed fragments is a challenging task. Two research teams have done some initial works on P2P join operations.

In [80], the authors describe a three layer architecture of the PIER system and implement two equi-join algorithms. In their design, a key is constructed from a “namespace” (relation) and a “resourceID” (primary key by default). Queries are multicast to all peers in the two namespaces to be joined. The first algorithm is a version of the symmetric hash join algorithm [WA91]. Each peer in the two namespaces finds the relevant tuples and hashes them to a new query namespace. The resource ID in the new namespace is the concatenation of join attributes. The second algorithm, called “fetch matches”, assumes that one of the relations is already hashed on the join attributes. Each peer in the second namespace finds tuples matching the query and retrieves the corresponding tuples from the first relation. The authors leverage two other techniques, namely the symmetric semi-join rewrite and the Bloom filter rewrite, to reduce the high bandwidth overheads of the symmetric hash join. For an overlay of $10,000$ peers, they evaluated the performance of their algorithms through simulation. The results show good performance of the proposed algorithms. However, for the cases where the join relations have a large number of tuples, this solution is not efficient, especially in terms of communication cost.

In [72], the authors considered multicasting to a large number of peers inefficient. Thus, they propose using a set of dedicated peers called range guards to monitor partitions of join attributes. Join queries are therefore sent only to range guards which decide the peers that should be contacted to execute the query.

5.1.3 Multi-Attributes Queries

There has been some work on multi-attribute P2P queries. The Multi-Attribute Addressable Network (MAAN) [59] is built on top of Chord to provide multi-attribute and range queries. They use a locality preserving hash function to map attribute values to the Chord identifier space, which is designed with the assumption that the data distribution could be known beforehand. Multi-attribute range queries are executed based on single-attribute resolution in $O(logn + n \cdot s_{min})$ routing hops, where n is the number of peers of the DHT and s_{min} is the minimum range selectivity across all attributes. The range selectivity is defined to be the ratio of the query range to the entire attribute domain range. However, the authors notice that there is a query selectivity breakpoint at which flooding becomes better than their scheme. Another drawback of MAAN is that it requires a fixed global schema which is known in advance to all peers. The authors followed up with the RDFPeers system to allow heterogeneity in peers schemas [58]. Each peer contains RDF based data items described as triples $\langle subject, predicate, object \rangle$. The triples are hashed onto MAAN peers. The experimental results show improvement in load balance, but no test for skewed query loads was done.

5.1.4 Fuzzy Queries

Information Need vs Query: In information systems, the *information need* is what the user (or group of users) desires to know from the stored data, to satisfy some intended objective (e.g., data analysis, decision making). However, the *query* is what the user submits to the system in an attempt to get that information need.

Precision vs Accuracy: Let us examine what is the relation between *precise* query statements and the *accuracy* of the returned results according to the information need. Consider the following relational table (Table 2) that maintains some patient records in a given hospital.[1]

Table 2 Patient table

Id	Age	Sex	BMI	Disease
t_1	36	*Female*	17	*Malaria*
t_2	23	*Male*	20	*Malaria*
t_3	45	*Female*	16.5	*Anorexia*
t_4	33	*Female*	23	*Malaria*
t_5	55	*Female*	21	*Rheumatism*
t_6	19	*Male*	18	*Malaria*

Suppose now that a doctor requires information about young patients diagnosed with Malaria (i.e., the information need). In a conventional SQL query, we must decide what are the ages of people considered as young. In the case where such age values fall into the $[21,35]$ range, the SQL query is written as follows:

```
Select all From Patient Where age in [21, 35]
and Disease = Malaria
```

The query above will return two tuples: t_2 and t_4. Unlike other contexts where the *young* term is well defined, such in banking applications where the age of clients precisely decides of the advantages they may benefit from, this term may not have a precise definition in biological contexts. For instance, the tuple t_6 may bring additional information to the doctor, affecting its analysis or decision. From this point of view, we say that the query results are not accurate, although the query has been precisely stated. One way to include tuple t_6 in the result set is to expand the scope of the selection predicate in order to encompass more data. Thus, the previous query is modified as follows:

```
Select all From Patient Where age in [18, 35]
and Disease = Malaria
```

Although it selects more tuples, the query still fails to find tuples lying just outside the explicit range of the selection predicate (e.g., tuple t_1). This is due to the crisp boundaries of the search range. Furthermore, there is no measure of inclusion, i.e., there is no way to know which tuples are strongly satisfying the information need and which are weakly satisfying it. Introducing fuzziness into user queries is a

[1] Body Mass Index (BMI): patient's body weight divided by the square of the height.

viable solution for that problem, i.e., introducing some imprecision in query statements may in some cases improve accuracy.

A fuzzy set is a class with *unsharp* boundaries. The grade of membership of an object in a fuzzy set is a number in the unit interval or, more generally, a point in a partially ordered set [51].

The application of gradual predicates, such as the YOUNG predicate presented in Fig. 16, results in associating membership degrees to tuples in a PATIENT relational table. For example, a tuple whose attribute value *t.age* is equal to 21 will be associated with a membership degree of 0.5 according to the YOUNG predicate. Hence, tuples can "partially" belong to the result set depending on how well they fit the information need.

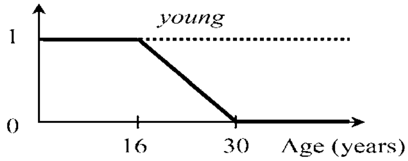


Fig. 16 Gradual predicate on attribute AGE

In [77], fuzzy techniques have been used in the design of P2P reputation systems based on collecting and aggregation peers' opinions. Characterizing peer's reputation by either "bad" or "good" based on some defined threshold is not adequate, as it would characterize in the same way a positive reputation produced by the collection of only positive opinions by many users and a reputation built with a limited number of heterogeneous opinions that produce a value immediately above the threshold; the same reasoning can be applied to negative reputations.

A recent work tends to introduce fuzziness into the BestPeer platform [94]. In [68], the authors propose FuzzyPeer, a generic P2P system which supports similarity queries. An image retrieval application is implemented as a case study. Fuzzy queries like "find the top-*k* images which are similar to a given sample" are very common in such applications because it is difficult for humans to express precisely an image's content in keywords or using precise attribute values. The authors investigate the problem of resolving similarity queries. The approach that consists in setting a similarity threshold and accepting objects only above this value is rejected. In fact, choosing the threshold value is not trivial given that the interpretation of an image depends on the user's perception of the domain. The approach proposed in [68] is based on the following observation: if two queries are similar, the top-*k* answers for the first one may contain (with high probability) some of the answers for the second query. In FuzzyPeer some of the queries are paused (i.e., they are not propagated further) and stay resident inside a set of peers. We use the term *frozen* for such queries. The frozen queries are answered by the stream of results that passes through the peers, and was initiated by the remaining running queries. Then, the authors propose distributed optimization algorithm in order to improve the scalability and the throughput of the system.

5.2 Partial Answering

As seen before, a first repercussion of the scale of P2P systems on query processing is that all queries can *partially* search the network. Another issue is that the amount of available data in P2P systems is dramatically increasing. More specifically, it becomes difficult to retrieve a few data items within a large structured data set in current PDMSs. Consider that a user issues the following query Q: *select hotels in Nice where price ¡ 100(euros) and proximity ¡ 8(km)*. The set R_Q of results returned by the set of relevant peers P'_Q may include a number of hotels that is so far from the one required by the user. Therefore, rank-aware queries like top-k and skyline queries started to emerge in order to provide a partial result subset R'_Q, with the k results having the highest grades of membership to the result set R_Q, i.e., $R'_Q \subseteq R_Q$. Indeed, the user is interested in the most relevant available results, which may be specified in the query as follows: *select hotels with cheap price, and yet close to the beach*. The degree of relevance (score) of the results to the query is determined by a scoring function.

Ranking results in a distributed manner is difficult because ranking is global: *all* results (matching a query) have to be ranked w.r.t. each other. In a completely distributed system, the results returned for identical queries should ideally be the same, which is not an issue in a centralized implementation. In a large-scale P2P system, the lack of a central location to aggregate global knowledge makes the problem of ranking challenging.

5.2.1 Top-k Queries

Given a dataset D and a scoring function f, a top-k query retrieves the k data items in D with the highest scores according to f. The scoring function is specified by the user according to its criteria of interests.

In unstructured P2P systems, one possible approach for processing top-k queries is to route the query to all peers, retrieve all available answers, score them using the scoring function, and return to the user the k highest scored answers. However, this approach is not efficient in terms of response time and communication cost. Top-k is a popular aspect of IR. As mentioned before, PlanetP [33] supports content ranking search in Peer IR systems. The top-k query processing algorithm works as follows. Given a query Q, the query originator computes a relevance ranking of peers with respect to Q, contacts them one by one from top to bottom of ranking and asks them to return a set of their top-scored document names together with their scores. To compute the relevance of peers, a global fully replicated index is used that contains term-to-peer mappings. This algorithm has very good performance in moderate-scale systems. However, in a large P2P system, keeping up-to-date the replicated index is a major problem that hurts scalability.

In the context of APPA, a fully distributed solution is proposed to execute top-*k* queries in unstructured P2P systems [74]. The solution involves a family of algorithms that are simple but effective. It executes top-*k* queries in completely distributed fashion and does not depend on the existence of certain peers. It also addresses the volatility of peers during query execution and deals with situations where some peers leave the system before finishing query processing.

In [92], the authors leverage the usage of super-peer networks, and propose an algorithm for distributed processing of top *k* queries on the top of Edutella [12]. In Edutella, a small percentage of nodes are super-peers and are assumed to be highly available with very good computing capacity. The super-peers are responsible for top-*k* query processing and other peers only execute the queries locally and score their resources. A limitation of this framework is that it assumes a global shared schema as well as consistent ranking methods employed at peers.

As for other complex queries, processing top-*k* queries in DHTs is quite challenging. A solution is to store all tuples of each relation by using the same key (e.g., relation's name), so that all tuples are stored at the same peer. Then, top-*k* query processing can be performed at that central peer using well-known centralized algorithms. However, the central peer becomes a bottleneck and single point of failure. In the context of APPA, a recent work has proposed a novel solution for Top-*k* query processing in DHT systems [75]. The solution is based on the TA algorithm [FLN03, GKB00, NR99] which is widely used in distributed systems. The solution is based on a data storage mechanism that stores the shared data in the DHT in a fully distributed fashion, and avoids skewed distribution of data among peers.

5.2.2 Skyline Queries

Top-*k* queries are sometimes difficult to define, especially if multiple aspects (i.e., scoring functions) have to be optimized. It is often not clear how to weight these aspects in order to obtain a global rank. Given such a multi-preference criteria, the concept of skyline queries provide a viable solution by finding a set of data points that are not *dominated* by any other points in a given data set. A point dominates another point if it is no worse in all concerning dimensions and better in at least one dimension according to user preferences. Objects belonging to skyline are precisely those objects that could be the best under some monotonic scoring functions. Most existing studies have focused mainly on centralized systems, and resolving skyline queries in a distributed environment such as a P2P network is still an emerging topic.

Reference [73] is the first attempt on progressive processing of skyline queries on a P2P network such as CAN [82]. The authors present a recursive region partitioning and a dynamic region encoding method to enforce a partial order over the CAN's zones, so that all the participating machines can be correctly pipelined for query execution. During the query propagation, data spaces are dynamically pruned and query results are progressively generated. Therefore, users do not have to wait for query termination to receive partial results, substantially reducing the query response time. However, this work focuses only on *constrained skyline queries* [32] where users are only interested in finding the skyline points among a subset of data items that satisfies multiple *hard* constraints. Besides, it suffers from workload imbalance caused by skewed query ranges.

A more recent work [84] has proposed an efficient solution for skyline query processing in the context of BestPeer. BestPeer [94] is a P2P platform that supports both structured and unstructured overlays. The solution proposed in [84] is called *Skyline Space Partitioning* (SSP), and is implemented in the BestPeer's structured network, called BATON [40]. It supports processing *unconstrained* skyline queries, which search skyline points in the whole data space. This work deals with the issue of imbalanced query load.

5.3 What About Approximate Answering?

To fix the ideas previously presented, and to eliminate any ambiguity, we precise here that "*approximate answering*",

- *Is not only about approximating the search space:* In Section 5.1, the fact of relaxing search completeness, due either to the limited coverage of routing protocols or to the inaccuracy of data indexes, has been referred as *"partial lookup"*.
- *Is not only about introducing flexibility into user's queries:* By flexible queries we refer to queries that may contain keywords, wildcards, ranges, or include user's preferences (e.g., top-k, skyline queries) or user's perception of the queried domain (e.g., fuzzy queries). This flexibility certainly supports users with more facilities to express their interests.
- *Is not only about approximating query evaluation techniques:* Query evaluation techniques, which are initially defined in centralized environments, can be only approximated in the context of P2P systems. Examples are [49] in which the notion of *relaxed skyline* is introduced, and [42] in which the well-known TA algorithm [87] is extended to adapt to P2P scenarios. For more illustration, the top-k answers returned to a user in a P2P system do not exactly match the set of top-k answers which would be obtained if all data were available and processed under a central coordination. This is considered as a natural repercussion of the nature of P2P networks on any computation method requiring some global information.
- *It is about returning approximate results, represented at a different level of abstraction:* As P2P systems start getting deployed in e-business and scientific environments, the vast amount of data within P2P databases poses a different challenge that has not been intensively researched until recently. In collaborative and decision support applications, a user may prefer an *approximate* but fast answer. Approximate answers do not belong to the original result set R_Q. However, they provide data descriptions $\tilde{R_Q}$, which may be queried or used as an alternative dataset for other operations input, including querying, browsing, or data mining.

Aggregation Queries

Aggregation queries have the potential of finding applications in decision support, data analysis and data mining. For example, millions of peers across the world may be cooperating on a grand experiment in astronomy, and astronomers may be interested in asking decision support queries that require the aggregation of vast amounts of data covering thousands of peers [17].

Consider a single table *T* that is horizontally partitioned and distributed over a P2P system. An aggregation query can be defined as follows:

```
Select Agg-Op(col) From T Where selection-condition
```

The *Agg-Op* may be any aggregation operator such as SUM, COUNT, AVG, MAX, and MIN. *Col* may be any numeric column of *T*, or even an expression involving multiple columns, and the *selection-condition* decides which tuples should be involved in the aggregation. Recently, traditional databases and decision support systems have witnessed the development of new *Approximate Query Processing techniques* AQP (e.g., [19, 95]) for aggregation queries. These techniques are mainly based on sampling, histograms, and wavelets.

Initial works have aimed to support aggregation queries in P2P systems by introducing OLAP techniques which employ materialized views over data ([61, 67]). However, the distribution and management of such views seems to be very difficult in such dynamic and decentralized environments. A recent work has investigated the feasibility of online sampling techniques for AQP in P2P systems [17]. The authors abandon trying to pick uniform random samples, which are nearly impossible to obtain in P2P systems. Instead, they have proposed to work with *skewed samples* while being able to accurately estimate the skew during sampling process.

Note that aggregation queries are not flexible, i.e., they are precisely formulated with specific operators. However, the aggregate values returned to the user provide information about tendencies within data. For example, the data cube [35], which is the most popular data model used for OLAP systems, generalizes the *GROUP BY* operation to *N* dimensions. Pre-computed aggregate values are stored in the cube cells and then, the OLAP system provides tools to navigate within these cells. This allows, for example, to examine the total number of sales of a given product in the last week of the current year, which have been reported in all cities of France (see Fig. 17).

Fuzzy Summaries

As seen before, fuzziness can be introduced into the user interface to allow more flexibility in query formulation. Fuzzy queries may be interpreted in a quantitative preference framework, provided that: (1) a membership function gives a similarity value of tuples to elementary query requirements (the fuzzy or gradual predicates) and (2) fuzzy aggregation computes an overall score that allows ranking items in the result set. However, we believe that it could not be the users' very first intention

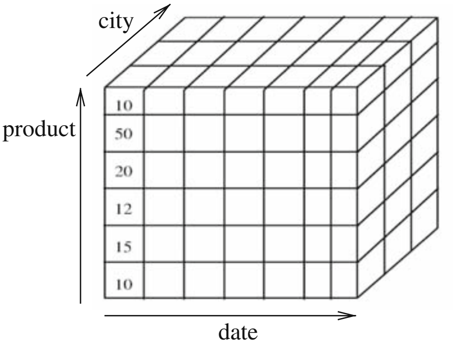


Fig. 17 Data cube

when they deal with such fuzzy queries. The simple fact that they need to define membership functions to compute attribute-oriented scores is somehow less natural than explicitly formulating preferences into query [90].

The literature also offers studies of how to express concepts or needs through constructs such as operators or linguistic variables [52]. One of the main challenges of extending query languages is to enrich query formulation without drastically reducing the performance of the query evaluation process. Linguistic summaries, studied by Yager et al. in [96], serve that concern by expressing the content of a set of data. The new expression is a description of the data using linguistic terms. Many works, some prior to Yager's, fall into the domain of linguistic summaries.

Quantified summaries approaches [71, 78] use fuzzy quantifiers in addition to linguistic terms to describe the data. For instance, in SummarySQL [78], evaluating "*summary most from* PATIENTS *where age is young*" provides a degree of validity for the proposition "*most* PATIENTS *are young*". Linguistic summaries also comprise fuzzy rules-based summaries. Such summaries are discovered by searching associations and relations between attribute values [23] or by exploiting fuzzy functional dependencies [22, 29]. They produce, in the case of gradual rules of Bosc et al. [23], propositions such as "*the more age is old, the more patient day is high*[2]". It is also possible to summarize records by repeatedly generalizing linguistic descriptions. This approach uses techniques from automatic learning and classification. Its output is a tree of descriptions. Lee and Kim's "*is-a*" hierarchies [53] and the SaintEtiQ model [34] are instances of this approach.

At the end of this chapter, we lighten the importance of querying such fuzzy summaries in centralized as well as in distributed P2P environments. First of all, these database summaries are a means of significantly reducing the volume of input for processes that require access to the database. The response time benefits from the downsizing. However, this response time gain is made clearly at the expense of a loss of precision in the answer (i.e., this what we are calling *approximate*

[2] Patient day: number of days spent in a hospital.

answering). This is of no importance when only a rough answer is required. Besides, imprecision can be sometimes a requirement. This is the case for instance when querying a medical database for anonymous, statistical information. Indeed, precise information can violate medical confidentiality. The loss of precision is also of no importance when a request only aims at determining the absence of information in a database. This is the case when one wants to know if a database is likely to answer the query.

Existing techniques have been proposed for querying fuzzy summaries in centralized environments [65, 91], however, such techniques have not been studied in P2P environments yet. The next chapter proposes a solution for managing fuzzy summaries in P2P systems to support DB applications with approximate query facilities.

References

1. http://www.gnutella2.com
2. http://www.gnutella.com
3. http://www.kazaa.com
4. http://www.napster.com
5. http://www.w3.org/2001/sw/
6. A.Crespo, H.G.Molina: Routing indices for peer-to-peer systems. In: Proc. of the 28 tn Conference on Distributed Computing Systems (2002)
7. A.Crespo, H.G.Molina: Semantic overlay networks for p2p systems. Tech. rep., Computer Science Department, Stanford University (2002)
8. A.Doan, J.Madhavan, R.Dhamankar, P.Domingos, A.Halevy: Learning to match ontologies on the semantic web. The VLDB Journal **12**(4), 303–319 (2003)
9. A.Gupta, D.Agrawal, A.El-Abbadi: Approximate range selection queries in peer-to-peer systems. In: CIDR (2003)
10. K.Aberer et al.: P-grid: a self-organizing structured p2p system. SIGMOD Rec. **32**(3), 29–33 (2003)
11. Q.Lv et al.: Search and replication in unstructured peer-to-peer networks. In: ACM Int. conference on Supercomputing (2002)
12. W.Nejdl et al.: Edutella: a p2p networking infrastructure based on rdf. In: WWW'02 (2002)
13. S.Androutsellis-Theotokis, D.Spinellis: A survey of peer-to-peer content distribution technologies. ACM Comput. Surv. **36**(4), 335–371 (2004)
14. A.Rowstron, P.Druschel: Pastry: Scalable decentralized object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware) (2001)
15. A.Rowstron, P.Druschel: Storage management and caching in PAST, a large–scale, persistent peer-to-peer storage utility. In: Proc.SOSP (2001)
16. A.Singla, C.Rohrs: Ultrapeers: another step towards gnutella scalability. Tech. rep. (2002)
17. B.Arai, G.Das, D.Gunopulos, V.Kalogeraki: Approximating aggregation queries in peer-to-peer networks. In: ICDE (2006)
18. M.Bawa, G.S.Manku, P.Raghavan: Sets: search enhanced by topic segmentation. In: SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, pp. 306–313 (2003)
19. B.Babcock, S.Chaudhuri, G.Das: Dynamic sample selection for approximate query processing. In: SIGMOD (2003)

20. B.Cooper, H-G.Molina: Ad hoc, self-supervising peer-to-peer search networks. ACM Trans. Inf. Syst. **23**(2), 169–200 (2005)
21. B.Maniymaran, M.Bertier, A-M.Kermarrec: Build one, get one free: Leveraging the coexistence of multiple p2p overlay networks. In: Proc of the 27th International Conference on Distributed Computing Systems ICDCS, p. 33. IEEE Computer Society, Washington, DC, USA (2007)
22. P.Bosc, D.Dubois, H.Prade: Fuzzy functional dependencies and redundancy elimination. JASIS **49**(3), 217–235 (1998)
23. P.Bosc, O.Pivert, L.Ughetto: On data summaries based on gradual rules. In: Fuzzy Days, pp. 512–521 (1999)
24. B.Yang, H-G.Molina: Improving search in peer-to-peer networks. In: Proc of the 22 nd International Conference on Distributed Computing Systems (ICDCS) (2002)
25. B.Zhao, J.Kubiatowicz, A.Joseph: Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. rep., Computer Science Division, U. C.Berkeley (2001)
26. B.Zhao, L.Huang, J.Stribling, S.Rhea, A.Joseph, J.Kubiatowicz: Tapestry: A resilient global-scale overlay for service deployment. IEEE J. Selected Areas Commun. **22**, 41–53 (2004)
27. C.Schmidt, M.Parashar: Enabling flexible queries with guarantees in p2p systems. IEEE Int. Comput. **08**(3), 19–26 (2004)
28. C.Tang, Z.Xu, M.Mahalingam: Peersearch: Efficient information retrieval in peer-to-peer networks. Tech. Rep. HPL-2002-198, HP Labs (2002)
29. J.C.Cubero, J.M.Medina, O.Pons, M.A.V.Miranda: Data summarization in relational databases through fuzzy dependencies. Inf. Sci. **121**(3–4), 233–270 (1999)
30. D.Malkhi, M.Naor, D.Ratajczak: Viceroy: a scalable and dynamic emulation of the butterfly. In: Proc of the twenty-first annual symposium on Principles of distributed computing, pp. 183–192 (2002)
31. D.Milojicic, *et al*: Peer-to-peer computing. Tech. rep., HP labs (2002)
32. D.Papadias, Y.Tao, G.Fu, B.Seeger: An optimal and progressive algorithm for skyline queries. In: ACM SIGMOD, pp. 467–478 (2003)
33. F.Cuenca-Acuna, C.Peery, R.Martin, T.Nguyen: Planetp: Using gossiping to build content addressable peer-to-peer information sharing communities. In: HPDC-12 (2003)
34. G.Raschia, N.Mouaddib: A fuzzy set-based approach to database summarization. Fuzzy sets and systems 129(2) pp. 137–162 (2002)
35. J.Gray, S.Chaudhuri, A.Bosworth, A.Layman, D.Reichart, M.Venkatrao, F.Pellow, H.Pirahesh: Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. J. Data Mining Knowl. Discov. **1**(1), 29–53 (1997)
36. S.Gribble, A.Halevy, Z.Ives, M.Rodrig, D.Suciu: What can databases do for peer-to-peer? In: WebDB Workshop on Databases and the Web (2001)
37. G.Skobeltsyn, T.Luu, Žarko, I.P., M.Rajman, K.Aberer: Query-driven indexing for scalable peer-to-peer text retrieval. Infoscale p. 14 (2007)
38. G.Wiederhold: Mediators in the architecture of future information systems. IEEE Computer **25**, 38–49 (1992)
39. J.M.Hellerstein: Toward network data independence. SIGMOD Rec **32**, 200–203 (2003)
40. H.Jagadish, B.Ooi, Q.Vu: Baton: A balanced tree structure for peer-to-peer networks. In: VLDB (2005)
41. H.Jagadish, B.Ooi, Q.Vu, R.Zhang, A.Zhou: Vbi-tree: A peer-to-peer framework for supporting multi-dimensional indexing schemes. In: ICDE, p. 34 (2006)
42. I.Chrysakis, D.Plexousakis, I.Chrysakis, D.Plexousakis: Semantic query routing and distributed top-k query processing in peer-to-peer networks. Tech. rep., Department of Computer Science, University of Crete (2006)
43. I.Clarke, S.Miller, T.Hong, O.Sandberg, B.Wiley: Protecting free expression online with freenet. IEEE Int. Comput. **6**(1), 40–49 (2002)
44. I.Stoica, R.Morris, D.Karger, M.F.Kaashoek, H.Balakrishnan: Chord: A scalabale peer-to-peer lookup service for internet applications. In: Proc ACM SIGCOMM (2001)

45. I.Tartinov et al.: The Piazza peer data management project. In: SIGMOD (2003)
46. S.Iyer, A.Rowstron, P.Druschel: Squirrel: a decentralized peer-to-peer web cache. In: PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing, pp. 213–222 (2002)
47. J.Kubiatowicz, D.Bindel, Y.Chen, S.Czerwinski, P.Eaton, D.Geels, R.Gummadi, S.Rhea, H.Weatherspoon, C.Wells, B.Zhao: Oceanstore: an architecture for global-scale persistent storage. SIGOPS Oper. Syst. Rev. **34**(5), 190–201 (2000)
48. D.Karger, E.Lehman, T.Leighton, R.Panigrahy, M.Levine, D.Lewin: Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web. In: STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, pp. 654–663 (1997)
49. K.Hose, C.Lemke, K.Sattler: Processing relaxed skylines in pdms using distributed data summaries. In: CIKM, pp. 425–434 (2006)
50. L.Adamic et al.: Search in power law networks. Phys. Rev. E **64**, 46,135–46,143 (2001)
51. L.A.Zadeh: Fuzzy sets. Inf. Control **8**, 338–353 (1965)
52. L.A.Zadeh: Concept of a linguistic variable and its application to approximate reasoning-I. Inf. Syst. **8**, 199–249 (1975)
53. D.H.Lee, M.H.Kim: Database summarization using fuzzy ISA hierarchies. IEEE Trans. on Systems, Man and Cybernetics-Part B: Cybernetics **27**, 68–78 (1997)
54. M.Lenzerini: Data integration: a theoretical perspective. In: PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 233–246 (2002)
55. Y.Li, L.Lao, J.H.Cui: Sdc: A distributed clustering protocol for peer-to-peer networks. In: Networking, pp. 1234–1239 (2006)
56. L.Ramaswamy, B.Gedik, L.Liu: A distributed approach to node clustering in decentralized peer-to-peer networks. IEEE Trans. Parallel Distributed Syst. **16**(9), 814–829 (2005)
57. J.Madhavan, P.A.Bernstein, A.Doan, A.Halevy: Corpus-based schema matching. In: ICDE '05: Proceedings of the 21st International Conference on Data Engineering, pp. 57–68 (2005)
58. M.Cai, M.Frank: Rdfpeers: a scalable distributed rdf repository based on a structured peer-to-peer network. In: WWW, pp. 650–657 (2004)
59. M.Cai, M.Frank, J.Chen, P.Szekely: Maan: A multi-attribute addressable network for grid information services. In: GRID (2003)
60. M.Castro, M.Costa, A.Rowstron: Should we build gnutella on a structured overlay? SIGCOMM Comput. Commun. Rev. **34**(1), 131–136 (2004)
61. M.Espil, A.Vaisman: Aggregate queries in peer-to-peer olap. In: DOLAP (2004)
62. W.Ng, B.Ooi, K-L.Tan, A.Zhou: Peerdb: A p2p-based system for distributed data sharing. In: ICDE, pp. 633–644 (2003)
63. N.Harvey, M.Jones, S.Saroiu, M.Theimer, A.Wolman: Skipnet: A scalable overlay network with practical locality properties. In: USENIX Symposium on Internet Technologies and Systems (2003)
64. O.Sahin A.Gupta, D.Agrawal, A.El-Abbadi: Query processing over peer-to-peer data sharing systems. Tech. rep., University of California, Santa Barbara (2002)
65. H.Y.Paik, N.Mouaddib, B.Benatallah, F.Toumani, M.Hassan: Building and querying e-catalog networks using p2p and data summarisation techniques. J. Intell. Inf. Syst. **26**(1), 7–24 (2006)
66. P.Bernstein, F.Giunchiglia, A.Kementsietsidis, J.Mylopoulos, L.Serafini, I.Zaihrayeu: Data management for peer-to-peer computing: A vision. In: Proc. of the 5th International Workshop on the Web and Databases (WebDB) (2002)
67. P.Kalnis, W.Ng, B.Ooi, D.Papadias, K.Tan: An adaptive peer-to-peer network for distributed caching of olap results. In: SIGMOD (2002)
68. P.Kalnis, W.Ng, B.Ooi, K.Tan: Answering similarity queries in peer-to-peer networks. Inf. Syst. **31**(1), 57–72 (2006)
69. P.Maymounkov, D.Mazieres: Kademlia: A peer-to-peer information system based on the xor metric. In: Int. Workshop on Peer-to-Peer Systems (IPTPS), pp. 53–65 (2002)

70. P.McBrien, A.Poulovassilis: Defining peer-to-peer data integration using both as view rules. In: DBISP2P, pp. 91–107 (2003)
71. H.Prade, C.Testemale: Generalizing database relational algebra for the treatment of incomplete/uncertain information and vague queries. Inf. Sci. **34**(2), 115–143 (1984)
72. P.Triantafillou, T.Pitoura: Towards a unifying framework for complex query processing over structured peer-to-peer data networks. In: DBISP2P, pp. 169–183 (2003)
73. P.Wu, C.Zhang, Y.Feng, B.Zhao, D.Agrawal, A.El-Abbadi: Parallelizing skyline queries for scalable distribution. In: EDBT, pp. 112–130 (2006)
74. R.Akbarinia, E.Pacitti, P.Valduriez: Reducing network traffic in unstructured p2p systems using top-k queries. Distrib. Parallel Databases **19**(2–3), 67–86 (2006)
75. R.Akbarinia, E.Pacitti, P.Valduriez: Processing top-k queries in distributed hash tables. In: Euro-Par, pp. 489–502 (2007)
76. R.Akbarinia, V.Martins, E.Pacitti, P.Valduriez: Design and implementation of appa. In: Global Data Management (Eds. R. Baldoni, G. Cortese and F. Davide). IOS press (2006)
77. R.Aringhieri, E.Damiani, S.Vimercati, S.Paraboschi, P.Samarati: Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems. J. Am. Soc. Inf. Sci. Technol. **57**(4) (2006)
78. D.Rasmussen, R.R.Yager: SummarySQL – a fuzzy tool for data mining. Intell. Data Anal. **1**, 49–58 (1997)
79. R.Dingledine, M.Freedman, D.Molnar: The free haven project: distributed anonymous storage service. In: International workshop on Designing privacy enhancing technologies, pp. 67–95. Springer-Verlag New York, Inc., New York, NY, USA (2001)
80. R.Huebsch, J.Hellerstein, N.Lanham, B.Thau, L.Shenker, I.Stoica: Querying the internet with pier. In: VLDB (2003)
81. S.Ratnasamy, M.Handley, R.Karp, S.Shenker: Topologically-aware overlay construction and server selection. In: Proceedings of IEEE INFOCOM'02 (2002)
82. S.Ratnasamy, P.Francis, M.Handley, R.M.Karp, S.Shenker: A scalable content – addressable network. In: SIGCOMM (2001)
83. K.Sripanidkulchai, B.M.Maggs, H.Zhang: Efficient content location using interest-based locality in peer-to-peer systems. In: INFOCOM (2003)
84. S.Wang, B.Ooi, A.Tung, L.Xu: Efficient skyline query processing on peer-to-peer networks. In: ICDE (2007)
85. T.Luu, G.Skobeltsyn, F.Klemm, M.Puh, Žarko, I.P., M.Rajman, K.Aberer: Alvisp2p: Scalable peer-to-peer text retrieval in a structured p2p network. In: Proc VLDB (2008)
86. D.Tsoumakos, N.Roussopoulos: A comparison of peer-to-peer search methods. In: Int.Workshop on the Web and Databases (WebDB), pp. 61–66 (2003)
87. U.Guntzer, W.Balke, W.Kieβling: Optimizing multi-feature queries for image databases. In: VLDB (2000)
88. J.D.Ullman: Information integration using logical views. In: ICDT '97: Proceedings of the 6th International Conference on Database Theory, pp. 19–40 (1997)
89. V.Kalogeraki, D.Gunopulos, D.Yazti: A local search mechanism for peer-to-peer networks. In: Proc CIKM. USA (2002)
90. A.Voglozin, G.Raschia, L.Ughetto, N.Mouaddib: Handbook of Research on Fuzzy Information Processing in Databases, vol. 1, chap. From User Requirements to Evaluation Strategies of Flexible Queries in Databases, pp. 115–142 (2008)
91. W.A.Voglozin, G.Raschia, L.Ughetto, N.Mouaddib: Querying the SAINTETIQ summaries–a first attempt. In: Int.Conf.On Flexible Query Answering Systems (FQAS) (2004)
92. W.Balke, W.Nejdl, W.Siberski, U.Thaden: Progressive distributed top-k retrieval in peer-to-peer networks. In: ICDE (2005)
93. W.Nejdl, W.Siberski: Design issues and challenges for rdf- and schema-based peer-to-peer systems. SIGMOD Record **32**, 2003 (2003)
94. W.Ng, B.Ooi, K.Tan: Bestpeer: A self-configurable peer-to-peer system. In: ICDE (2002)
95. X.Li, Y.J.Kim, R.Govindan, W.Hong: Multidimensional range queries in sensor networks. In: SENSYS (2003)

96. R.R.Yager: On linguistic summaries of data. In: Knowledge Discovery in Databases, pp. 347–366. MIT Press (1991)
97. B.Yang, P.Vinograd, H.Garcia-Molina: Evaluating guess and non-forwarding peer-to-peer search. In: ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04), pp. 209–218 (2004)
98. Y.Chawathe, S.Ratnasamy, L.Breslau, N.Lanham, S.Shenker: Making gnutella-like p2p systems scalable. In: In Proc. ACM SIGCOMM (2003)
99. Y.Halevy, G.Ives, D.Suciu, I.Tatarinov: Schema mediation for large-scale semantic data sharing. VLDB J. **14**(1), 68–83 (2005)

Managing Linguistic Data Summaries in Advanced P2P Applications

Rabab Hayek, Guillaume Raschia, Patrick Valduriez, and Noureddine Mouaddib

Abstract As the amount of stored data increases, data localization techniques become no longer sufficient in P2P systems. A practical approach is to rely on compact database summaries rather than raw database records, whose access is costly in large P2P systems. In this chapter, we describe a solution for managing linguistic data summaries in advanced P2P applications which are dealing with semantically rich data. The produced summaries are synthetic, multidimensional views over relational tables. The novelty of this proposal relies on the double summary exploitation in distributed P2P systems. First, as semantic indexes, they support locating relevant nodes based on their data descriptions. Second, due to their intelligibility, these summaries can be directly queried and thus approximately answer a query without the need for exploring original data. The proposed solution consists first in defining a summary model for hierarchical P2P systems. Second, appropriate algorithms for summary creation and maintenance are presented. A query processing mechanism, which relies on summary querying, is then proposed to demonstrate the benefits that might be obtained from summary exploitation.

Rabab Hayek
LINA, 2 rue de la Houssiniere, 44322, Nantes, France,
e-mail: rabab.hayek@univ-nantes.fr

Guillaume Raschia
LINA, 2 rue de la Houssiniere, 44322, Nantes, France,
e-mail: guillaume.raschia@univ-nantes.fr

Patrick Valduriez
INRIA, 2 rue de la Houssiniere, 44322, Nantes, France,
e-mail: Patrick.Valduriez@inria.fr

Noureddine Mouaddib
LINA, 2 rue de la Houssiniere, 44322, Nantes France,
e-mail: Noureddine.Mouaddib@univ-nantes.fr

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_20, © Springer Science+Business Media, LLC 2010

1 Introduction

As revealed by the large number of P2P research papers and surveys (including the previous chapter), data localization has been the main issue addressed in the P2P community. However, advanced data management techniques, which allow more than locating data, are strongly required to support database applications. These applications are facing the *information explosion* problem: a huge amount of information is generated and stored each day into data sources (e.g., biological, astronomical database applications). This ever increasing amount of available data, in addition to the high distribution of connected resources, makes search techniques no more sufficient for supporting P2P database applications. To illustrate, in a scientific collaborative application, a doctor may require information about patients diagnosed with some disease, without being interested in individual patient records. Besides, in today's decision-support applications, users may prefer an approximate but fast answer, instead of waiting a long time for an exact one.

To address the problem of managing voluminous databases, the data summarization paradigm has emerged into the database field. The objective of data summarization is to synthesize the information which is disseminated within large datasets, in order to provide the *essential*. Obviously, data summarization is of higher importance in P2P systems. In these systems, the participants share a global database which is equivalent to the aggregation of all their local databases, and thus may become much more voluminous. Therefore, *Reasoning on compact data descriptions that can return approximate answers like "dead Malaria patients are typically children and old" to queries like "age of dead Malaria patients", is much more efficient than retrieving raw records, which may be very costly to access in highly distributed P2P databases.*

The authors in [21] propose a new solution for managing linguistic data summaries in P2P systems. The produced summaries are synthetic, multidimensional views over relational tables. The novelty of this proposal relies on the double exploitation of summaries in distributed P2P systems. First, as semantic indexes, they support locating relevant nodes based on their data descriptions. Second, due to their intelligibility, these summaries can be directly queried to approximately answer a query without the need for exploring original data, which might be highly distributed in P2P systems.

This work makes the following contributions. Section 2 first proposes a summarization process that exhibits many salient features making from it an interesting process to be integrated into P2P environments. In Section 3, a summary model is defined in the context of hierarchical P2P systems. The network is organized into *domains*, where a domain is defined as being the set of a supernode and its associated leaf nodes. In a given domain, peers cooperate to maintain a global summary over their shared data. Section 4 discusses the gains that might be obtained in query processing from the use of data summaries. In Section 5, the performance of the summary proposal is evaluatedthrough a cost model and a simulation model.

Simulation results show that the cost of query routing is significantly reduced without incurring high costs of summary updating. Section 6 discusses related works, and Section 7 concludes.

2 Summarization Process

In this section, we first briefly discuss the existing approaches of data summarization according to some required characteristics. Then, we define the input data and describe the summarization process of the approach adopted in [21]. Finally, the structure of distributed summaries in P2P systems is formally defined.

2.1 Data Summarization

The data summarization paradigm has been extensively studied in centralized environments to address the problem of managing voluminous data sets. In the literature, many approaches have been proposed, each satisfying the requirements of specific applications. Here, we discuss the characteristics that should control the choice of a data summarization process in order to satisfy the purposes discussed in our introduction.

- *Compression:* the summarization process should provide compact versions of the underlying data. However, we are not referring here to (syntactic) compression techniques. These techniques consider a data source as a large byte string and thus use compression algorithms such as Huffman or Lempel-Ziv coding. They were mainly proposed to deal with throughput and space storage constraints. In fact, we are concerned with compression (or data reduction) techniques that can provide fast and approximate answers. This is very efficient in the context where obtaining an exact answer is a time-consuming process, and an approximate answer may give information that are sufficiently satisfying. Examples of such techniques are discrete wavelet transform and linear regression used in signal processing [19], sampling and histograms [14] used in statistics. Index trees such as B^+Tree [8] and K-d-Tree [6] could also be considered as data reduction techniques in the database field. These indexes do not provide approximate answers, however, they allow to optimize and accelerate the access within a large data set.

- *Intelligibility:* the summarization process should synthesize the information disseminated within large datasets in order to provide the *essential*. In fact, the data summarization techniques are data reduction techniques that are supposed to provide intelligible representations of the underlying data. Three categories

of database summarization techniques have been proposed in the literature. The first one focuses on aggregate computation. Examples are OLAP and multidimensional databases which allow an end-user to query, visualize and access part of the database using cubes of aggregate values computed from raw data [5]. The second category, so-called semantic compression (SC), deals with intentional characterization of groups of individuals to provide higher-level models such as decision trees or association rules. Examples are [12, 24] which explicitly refer to semantic compression of structured data.
The last category is interested in metadata-based semantic compression (MDBSC) approaches. These approaches use metadata to guide the compression process. The produced summaries are highly comprehensible since their descriptions rely on user-defined vocabularies. The main difference between SC and MDBSC is that the latter provide data descriptions which precisely fit the user perception of the domain, whereas the former aims to identify hidden patterns from data. One representative of the MDBSC approaches is the Attribute-Oriented Induction process (AOI). It provides reduced versions of database relations using *is-a* hierarchies, i.e., a concept tree built over each attribute domain [11].

- *Robustness:* the summarization process should handle the vagueness and the imprecision inherent to natural language. The theory of fuzzy sets provides a formal framework associated with a symbolic/numerical interface using linguistic variables [17] and fuzzy partitions [23]. Hence, the linguistic summaries have the advantages of being robust and formulated in a user-friendly language (i.e., linguistic labels).

- *Scalability:* the summarization process should be scalable in terms of the amount of processed data. It should be able to treat voluminous databases with low time complexity, and controlled memory consumption. Besides, an important issue is the ability of the process to be parallelized and distributed among multiple processors or computers.

In [21], the approach used for data summarization is based on SaintEtiQ [22]: an online linguistic approach for summarizing databases. The SAINTETIQ model aims at apprehending the information from a database in a synthetic manner. This is done through generating linguistic, multidimensional summaries that are arranged and incrementally maintained in a hierarchy. The hierarchies of SAINTETIQ summaries differ from those obtained by the *is-a* approach in that they rely on one set of linguistic terms, without level assignment. Indeed, the SAINTETIQ model proceeds first in an abstraction of data by the use of a user-defined vocabulary, and then in performing a classification that produces data descriptions at different levels of granularity (i.e., levels of abstraction). The SAINTETIQ process exhibits many interesting features and is proposed to be efficiently integrated into P2P environments, as it is revealed in the rest of this section.

2.2 Input Data

The summarization process takes as input the original data to be summarized, and the *"Background Knowledge"* BK which guides the process by providing information about the user's perception of the domain.

2.2.1 Data Model

The data to be summarized come from relational databases. As such, the data are organized into records, with a schema $R(A_1, A_2, \ldots, A_n)$. Each attribute A_i is defined on an attribute domain D_i, which may be numeric or symbolic. Thus, each tuple t consists of n attribute values from domains D_1 to D_n. It is given by:

$$t = \langle t.A_1, t.A_2, \ldots, t.A_n \rangle$$

A constraint on these data is that it should be complete: any value $t.A_i$ is necessarily known, elementary, precise, and certain. In other terms, all records with a null value are dismissed.

2.2.2 Background Knowledge

A unique feature of the summarization system is its extensive use of a *Background Knowledge (BK)*, which relies on linguistic variables and fuzzy partitions.

Consider the following relational database of a given hospital, which is reduced to a single *Patient* relation (Table 1).[1] Figure 1 shows a linguistic variable defined on the attribute AGE where descriptor YOUNG ADULT is defined as being plainly satisfactory to describe values between 19 and 37 and less satisfactory as the age is out of this range. Similarly, Fig. 2 provides the linguistic variable defined on attribute BMI.[2] Thus, linguistic variables come with linguistic terms (i.e., descriptors) used to characterize domain values and, by extension, database tuples. For a continuous domain D_i, the linguistic variable is a fuzzy partition of the attribute domain. For a discrete domain D_i (DISEASE and SEX in our example), the BK element is a fuzzy set of nominal values. In short, the BK supports the summarization process with means to match attribute domain values with the summary expression vocabulary.

Note that the BK, given by users or experts of the data domain, concerns the attributes which are considered as pertinent to the summarization process. In our example, we have excluded the *patient day* attribute. However, when the descriptions of that attribute values might be useful for the hospital application (i.e., requiring information about the *length of stay* of patients), a corresponding fuzzy partition should be also provided.

[1] Patient day: number of days spent in the hospital.

[2] BMI is the patient's body weight divided by the square of its height.

Table 1 Raw data

Id	Age	Sex	BMI	Patient days	Disease
t_1	16	*Female*	16	350	*Anorexia*
t_2	60	*Male*	32	4	*High blood pressure*
t_3	18	*Female*	17	280	*Anorexia*
t_4	17	*Female*	18	230	*Anorexia*
t_5	54	*Female*	26	14	*Osteoporosis*

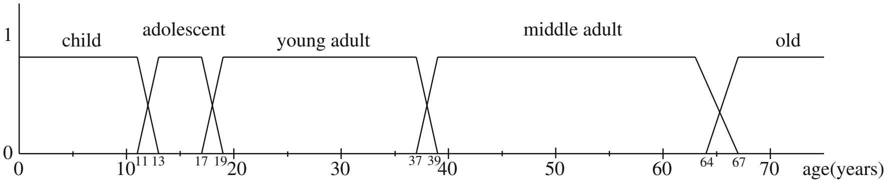


Fig. 1 Fuzzy linguistic partition on `age`

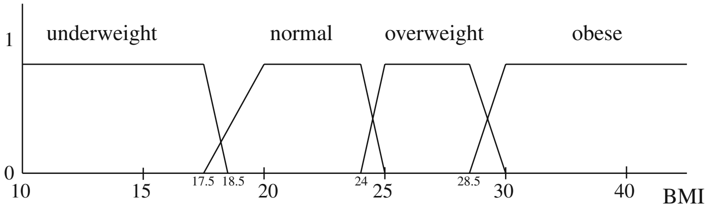


Fig. 2 Fuzzy linguistic partition on `BMI`

2.3 Process Architecture

A service oriented architecture has been designed for the summarization process in order to incrementally build data summaries. By "incremental", we mean that the database tuples are processed one by one, and the final hierarchy of summaries is obtained by repeating this summarization process for each tuple in the table at hand. The architecture is organized into two separate services: online mapping and summarization.

2.3.1 Mapping Service

The mapping service takes as input the original relational records and performs an abstraction of the data using the BK. A representation of the data under the form of fuzzy sets is obtained. Basically, the mapping operation replaces the original attribute values of every record in the table by a set of linguistic descriptors defined in the BK. For instance, with the linguistic variable defined on the attribute AGE (Fig. 1), a value $t.AGE = 18$ years is mapped to $\{0.5/adolescent, 0.5/young\ adult\}$

where 0.5 is a membership grade that tells how well the label *young adult* describes the value 18. Extending this mapping to all the attributes of a relation could be seen as locating the overlapping cells in a grid-based multidimensional space which maps records of the original table. The fuzzy grid is provided by the BK and corresponds to the user's perception of the domain.

In our example, tuples of Table 1 are mapped into four distinct grid-cells denoted by c_1, c_2, c_3, c_4, and c_5 in Table 2. The fuzziness in the vocabulary definition of *BK* permits to express any single value with more than one fuzzy descriptor and thus avoid threshold effect thanks to the smooth transition between different categories. For instance, the tuple t_3 of Table 1 is mapped to the two grid cells c_1 and c_2 since the value of the attribute *age* is mapped to the two linguistic terms: *adolescent* and *young adult*. Similarly, the tuple t_4 is mapped to c_1 and c_3 since the value of the attribute *BMI* is mapped to $\{0.7/underweight, 0.3/normal\}$. The tuple count column gives the proportion of records that belongs to the cell, and 0.5/*young adult* says that *young adult* fits the data only with a degree of 0.5. The degree of a label is the maximum of membership grades to that label, computed over all the tuples in the corresponding grid cell. For instance, the degree of *adolescent* in c_1 is equal to one, which is the maximum grade value of the two tuples t_1 and t_3.

Table 2 Grid-cells mapping

Id	Age	Sex	BMI	Disease	Tuple count
c_1	*Adolescent*	*Female*	*Underweight*	*Anorexia*	2.2
c_2	0.5/*Young adult*	*Female*	*Underweight*	*Anorexia*	0.5
c_3	*Adolescent*	*Female*	0.3/*Normal*	*Anorexia*	0.3
c_4	*Middle adult*	*Male*	*Obese*	*High blood pressure*	1
c_5	*Middle adult*	*Female*	*Overweight*	*Osteoporosis*	1

Therefore, the BK leads to the point where tuples become indistinguishable and then are grouped into grid-cells such that there are finally many more records than cells. Every new (coarser) tuple stores a record count and attribute-dependent measures (min, max, mean, standard deviation, etc.). It is then called a *summary*.

2.3.2 Summarization Service

The summarization service is the last and the most sophisticated step of the SAINTETIQ system. It takes *grid-cells* as input and outputs a collection of summaries hierarchically arranged from the most generalized one (the root) to the most specialized ones (the leaves) [22]. Summaries are clusters of grid-cells, defining hyperrectangles in the multidimensional space. In the basic process, leaves are grid-cells themselves and the clustering task is performed on K cells rather than N tuples ($K << N$).

From the mapping step, cells are introduced continuously in the hierarchy with a top-down approach inspired of D.H. Fisher's Cobweb [16], a conceptual clustering algorithm. Then, they are incorporated into best fitting nodes descending the tree. Three more operators could be apply, depending on partition's score, that are *create*, *merge* and *split* nodes. They allow developing the tree and updating its current state. Figure 3 represents the summary hierarchy built from the cells c_1, c_2, c_3, c_4, and c_5.

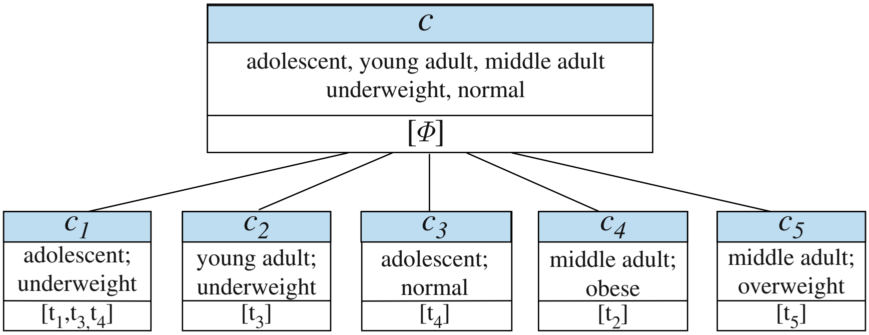


Fig. 3 Example of SaintEtiQ hierarchy

The SAINTETIQ process has been successfully integrated to an existing DataBase Management System (DBMS) thanks to the design of a service-oriented architecture where service calls are made through the SOAP protocol. On the other hand, the *eXtensible Markup Language* XML is the format adopted for exchanging data with the DBMS, as well as for summary representation all along the summarization process (i.e., raw data, BK, grid cells and the final summaries are represented and exchanged under the form of XML documents).

2.3.3 Scalability Issues

Memory consumption and time complexity are the two main factors that need to be taken care of in order to guaranty the capacity of the summary system to handle massive datasets. First, the time complexity of the SAINTETIQ process is in $O(n)$, where n is the number of candidate tuples to incorporate into a hierarchy of summaries. However, the number of candidate tuples that are produced by the mapping service is dependent only on the fuzziness of the BK definition. A crisp BK will produce exactly as many candidate tuples as there are original tuples. Besides, an important feature is that in the summarization process, raw data have to be parsed only once, and this is performed with a low time cost. Second, the system requires low memory consumption for performing the summary construction algorithm as well as for storing the produced summaries. Moreover, a cache manager is in charge of summary caching in memory and it can be bounded to a given memory requirement. Usually, less than a hundred of summaries are needed in the cache since this number covers the two or three top levels of even a wide hierarchy. Least recently used summaries are discarded when a required summary is not found in the cache.

On the other hand, the parallelization of the summary system is a key feature to ensure smooth scalability. The implementation of the system is based on the Message-Oriented Programming paradigm. Each sub-system is autonomous and collaborates with the others through disconnected asynchronous method invocations. It is among the least demanding approaches in terms of availability and centralization. The autonomy of summary components allows for a distributed computing of the summary process. Once a component completes the treatment and evaluates the best operator for the hierarchy modification, if needed, a similar method is successively called on children nodes. The cache manager is able to handle several lists of summaries residing on different computers [22].

2.4 Distributed Summary Representation

In this section, we introduce basic definitions related to the summarization process.

Definition 1. *Summary* Let $E = \langle A_1, \dots, A_n \rangle$ be a n-dimensional space equipped with a grid that defines basic n-dimensional areas called cells in E. Let R be a relation defined on the cartesian product of domains D_{A_i} of dimensions A_i in E. Summary z of relation R is the bounding box of the cluster of cells populated by records of R.

The above definition is constructive since it proposes to build generalized summaries (hyper-rectangles) from cells that are specialized ones. In fact, it is equivalent to performing an *addition* on cells:

$$z = c_1 + c_2 + \dots + c_p$$

where $c_i \in L_z$, the set of p cells (summaries) covered by z.

A summary z is then an *intentional description* associated with a set of tuples R_z as its *extent* and a set of cells L_z that are populated by records of R_z.

Thus, summaries are areas of E with hyper-rectangle shapes provided by BK. They are nodes of the summary tree built by the SAINTETIQ system.

Definition 2. *Summary Tree* A summary tree is a collection S of summaries connected by $\preccurlyeq$, the following partial order:

$$\forall z, z' \in \mathscr{Z}, \quad z \preccurlyeq z' \Longleftrightarrow R_z \subseteq R_{z'}$$

The above link between two summaries provides a generalization/specialization relationship. And assuming that summaries are hyper-rectangles in a multidimensional space, the partial ordering defines *nested summaries* from the larger one to the single cells. General trends in the data could be identified in the very first levels of the tree whereas precise information has to be looked at near the leaves.

In [21], a summary tree is also considered as an indexing structure over distributed data in a P2P system. Thus, a new dimension has been added to the definition of a summary node z: a *peer-extent* P_z, which provides the set of peers having data described by z.

Definition 3. *Peer-extent* Let z be a summary in a given hierarchy of summaries S, and P the set of all peers who participated to the construction of S. The *peer-extent* P_z of the summary z is the subset of peers owning, at least, one record of its extent R_z: $P_z = \left\{p \in P \mid R_z \cap R_p \neq \emptyset\right\}$, where R_p is the view over the database of node p, used to build summaries.

Due to the above definition, the notion of *data-oriented* summary in a given database is extended to a *source-oriented* summary in a given P2P network. In other words, summaries can be used as database indexes (e.g., referring to relevant tuples), as well as semantic indexes in a distributed system (e.g., referring to relevant nodes).

The summary hierarchy S will be characterized by its *Coverage* in the P2P system; that is, the fraction of data sources described by S. Relative to the hierarchy S, *Partner Peer* is a peer whose data is described by at least a summary of S.

Definition 4. *Partner peers* The set of *Partner peers* P_S of a summary hierarchy S is the union of peer-extents of all its summary nodes in S: $P_S = \{\cup_{z \in S} P_z\}$.

For simplicity, in the following we designate by "summary" a hierarchy of summaries maintained in a P2P system, unless otherwise specified.

3 Summary Model for Hierarchical P2P Networks

In this section, we first define the problem of managing data summaries in P2P systems. Second, we describe the architecture of the summary model adopted for hierarchical networks. Then, we present the solutions proposed for summary creation and maintenance.

3.1 Problem Statement

Given a P2P network, we consider the two following assumptions.

- Each peer p owns some tuples (R_p) in a global, horizontally partitioned relation R.
- Users that are willing to cooperate agree on a Background Knowledge BK, which represents their common perception of the domain.

Thus, here the problem of semantic heterogeneity among peers is not addressed, since it is a separate P2P issue on its own. Besides, this work mainly targets collaborative database applications where the participants are supposed to work on "related" data. In such a context, the number of participants is also supposed to be limited, and thus the assumption of a common *BK* seems not to be a strength constraint. An example of such *BK* in a medical collaboration is the Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT) [2], which provides a common language that enables a consistent way of capturing, sharing and aggregating health data across specialties and sites of care. On the other hand, the used summaries are data structures that respect the original data schemas [22]. Hence, we can assume that the techniques that have been proposed to deal with information integration in P2P systems (e.g., [15, 20]) can be used here to overcome the heterogeneity of both data and summary representations, in the context of heterogeneous data.

Let $G = (V,E)$ be the graph corresponding to a P2P network of size N, where V is the set of nodes (i.e., $|V| = N$), and E is the set of links between nodes. The ultimate goal is to maintain a global summary that completely describes the global relation R. However, as stated before, the relation R is horizontally partitioned and distributed among autonomous peers. Hence, the problem can be defined as follows. Given that each peer p_i locally maintains a Local Summary LS_i, we aim to construct the global summary GS_c such that:

$$GS_c = \cup_{i=1}^{N}(LS_i)$$

The local summaries are obtained by integrating the summarization process previously defined into each peer's DBMS. The operator $\cup$ designates the summary merging operation which will be discussed later.

The autonomous and dynamic nature of P2P networks makes the convergence to GS_c quite challenging. It is difficult to build and to keep this summary consistent relative to the current data instances it describes. So, the problem can be redefined as follows.

Given the set of materialized local summaries $\{LS_i, 1 \leq i \leq N\}$, we require to build/materialize the set of global summaries $\{GS_j, 1 \leq j \leq N_G\}$ such that:

- $GS_j = \cup_{i=1}^{l}(S_i)$, where S_i is a local or global summary. Here, the latter is defined as being the merging result of, at least, two summaries (i.e., $l \geq 2$).
- The set of materialized local/global summaries (each having its set of partner peers P_{GS_j}), and the set of links ($E_s \subset E$) between nodes belonging to different sets of partner peers (i.e., links connecting different summaries), provide together an approximation of the *virtual* summary GS_c.

$$GS_c \approx (\{LS_i, 1 \leq i \leq N_L\}, \{GS_j, 1 \leq j \leq N_G\}, E_s) \qquad (1)$$

  N_L is the number of local summaries, which is the number of peers that have not participated to any existing global summary GS_j. While N_G is the number of global summaries built in the network (i.e., $|\cup_{j=1}^{N_G}(P_{GS_j})| + N_L = N$).
- A "good" trade-off should be achieved between the cost of updating the set of materialized summaries and the benefits obtained from exploiting these summaries in query processing.

3.2 Model Architecture

Data indexes are maintained in P2P systems using one of the following approaches. A *centralized* approach maintains a global index over all the data shared in the network, and thus provides a centralized-search facility. A *hybrid decentralized* approach distributes indexes among some specialized nodes (e.g., supernodes), while a *pure decentralized* approach distributes indexes among all the participants in the network (e.g., structured DHTs, Routing Indices). Each of these approaches provides a different trade-off between the cost of maintaining the indexes and the benefits obtained for queries. In [21], the second approach is adopted since it is the only one that exploits peer heterogeneity, which is a central key to allow P2P systems scaling up without compromising their decentralized nature [25]. Besides, some also argue that the super-peer networks [7] are the most suited for content-based search [13, 27] since their inherent hierarchical structure allows to build a similar hierarchy over the shared data and metadata.

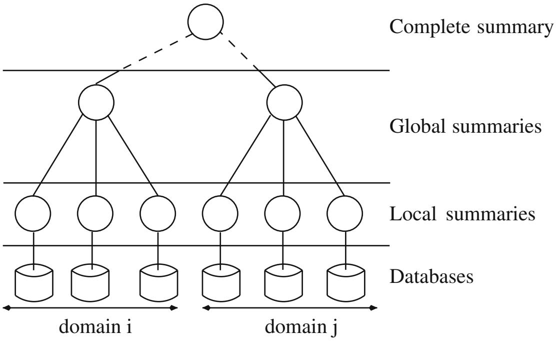


Fig. 4 Model architecture for hierarchical P2P networks

Let us examine the architecture of the summary model which is presented in Fig. 4. It is clear that the hierarchy among global summaries follows the hierarchical structure of the underlying network (two summary levels). The network is organized into *domains*, where a domain is defined as being the set of a supernode and its associated leaf nodes. In a given domain, peers cooperate to maintain a global summary over their shared data. The set of global materialized summaries and links between the corresponding domains, provide an approximation of the summary GS_c (Equation 1).

3.3 Summary Management

In this section, we present the algorithms for summary construction and maintenance in a given domain. First, the algorithms are presented in a static context, where all participants remain connected, and then they are extended to cope with peer dynamicity.

3.3.1 Summary Construction

Each global summary *GS* is associated with a *Cooperation List* (*CL*) that provides information about its partner peers. An element of *CL* is composed of two fields. A partner identifier *PeerID*, and a 2-bit freshness value *v* that provides information about the description freshness as well as the availability of the corresponding database.

- value 0 (initial value): descriptions are fresh relative to the original data,
- value 1: descriptions need to be refreshed,
- value 2: original data are not available. This value is used while addressing peer volatility in Section 3.3.3.

Algorithm 18 shows the messages exchanged between peers in order to build a global summary *GS*. The algorithm starts at a superpeer (referred later as *Summary Peer SP*) who broadcasts a SUMPEER message that contains its identifier, to indicate its ability to host summaries. Since *SP* is supposed to have high connectivity, a small value of TTL (Time-To-Live) is sufficient to cover a large number of peers (e.g., $TTL = 2$).

The message contains also a hop value *h*, initialized to 0, which is used to compute the distances between *SP* and the visited peers. A peer *p* who received a first SUMPEER message, maintains information about the corresponding summary peer *SP* (i.e., Line 13). Then, *p* sends to *SP* a LOCALSUM message that contains its local summary *LS*, and thus becomes a partner peer in the *SP*'s domain. Upon receiving this last message, *SP* merges *LS* to its current global summary *GS*, and adds a new element to *CL*.

However, a peer *p* who is already a partner may receive a new SUMPEER message. In such a case, only if the new summary peer is nearer than the old one (based on latency), it chooses to drop its old partnership through a DROP message (i.e., Line 11), and it proceeds to participate to a new domain. We now suppose that a peer *p* does not belong to any domain (i.e., *p* is not a partner peer), and wants to participate to a global summary construction. Using a selective walk, it can rapidly find a summary peer *SP* (i.e., FIND message). The information about *SP*, which is maintained at each of its partners, makes the selective walk even shorter. Once a partner or a summary peer is reached, the FIND message is stopped (i.e., Line 27).

3.3.2 Summary Maintenance

An important issue for any indexing technique is to efficiently maintain the indexes against data changes.

Algorithm 18: *Global Summary Construction*

```
// Definition of different types of messages
SUMPEER=$\langle sender\rangle\ \langle id, h, TTL\rangle$; FIND=$\langle sender\rangle\ \langle h, TTL\rangle$
LOCALSUM=$\langle sender\rangle\ \langle LS\rangle$; DROP=$\langle sender\rangle\ \langle\rangle$
// Treatment of messages
Switch msg.type
// Receiving information about a summary peer
Case (SumPeer):
    $msg.h^{++}$; $msg.TTL^{--}$
    if (this.SumPeer=null) or (this.SumPeer.h $>$ msg.h) then
        if (this.IsPartner) then
            Send DROP message to this.SumPeer.id
        end if
        this.SumPeer:= $\langle msg.id, msg.h\rangle$
        LOCALSUM:= new msg (this.LS); Send LOCALSUM to msg.sender
        IsPartner:= True
    end if
    if $msg.TTL > 0$ then
        Send msg to all neighbors
    end if
end Case
// Searching for a summary peer
Case (Find):
    $msg.TTL^{--}$; $msg.h^{++}$
    if (this.Is_SumPeer) then
        PEERSUM:= new msg (this.id, msg.h, 1); Send PEERSUM to msg.sender
    else
        if (this.SumPeer $\neq$ null) then
            PEERSUM:= new msg (this.SumPeer.id, (msg.h + this.SumPeer.h), 1)
            Send PEERSUM to msg.sender
        else
            if ($msg.TTL > 0$) then
                $p'$:= highest degree peer in $N(p)$; Send msg to $p'$
            end if
        end if
    end if
end Case
// arrival of a new partner
Case (LocalSum):
    CoopList.add (msg.sender, 0); GlobalSum:= merge (GlobalSum, msg.LS)
end Case
// departure of a partner
Case (Drop):
    CoopList.remove (msg.sender)
end Case
```

For a local summary, it has been demonstrated that the summarization process guarantees an incremental maintenance (using a *push* mode for exchanging data with the DBMS), while performing with a low complexity. This allows for an online data processing without the need for an overall summary computation.

In this section, a strategy for maintaining *global summary data* is proposed: a global summary *GS* which has been obtained by merging the local summaries of the set of peers P_{GS}, and its associated cooperation list *CL*. The objective is to keep *GS* consistent with the current instances of the local summaries.

The latters are supposed to be consistent with the current instances of the original data sources. The maintenance strategy uses both *push* and *pull* techniques, in order to minimize the number of messages exchanged in the system.

Push: Cooperation List Update

Each peer *p* in P_{GS} is responsible for refreshing its own element in the cooperation list *CL*. The partner *p* monitors the modification rate issued on its local summary *LS*. When *LS* is considered as enough modified, the peer *p* sets its freshness value *v* to 1, through a push message. The value 1 indicates that the local summary version being merged while constructing *GS* does not correspond any more to the current instance of the database.

An important feature is that the frequency of push messages depends on modifications issued on local summaries, rather than on the underlying databases. It has been demonstrated in [22] that, after a given process time, a summary becomes very stable. As more tuples are processed, the need to adapt the hierarchy decreases and hopefully, once all existing attribute combinations have been processed, incorporating new tuple consists only in sorting it in a tree. A summary modification may be detected by observing the appearance/disappearance of descriptors in the summary intention.

Pull: Summary Update

The summary peer *SP*, in its turn, monitors the fraction of old descriptions in *GS*, i.e., the number of ones in *CL*. Upon each push message sent to update a freshness value, this fraction value is checked. If it exceeds a threshold value α, the summary update mechanism will be then triggered. Note that the threshold α is our parameter by which the freshness degree of *GS* will be controlled. In order to update *GS*, all the partner peers will be pulled to merge their current local summaries into a *GS*'s version. The algorithm is described as follows.

SP initiates a summary update message Sum_Update which is then propagated from a partner to another. This message contains a new summary *NewGS* (initially empty), and a list of peer identifiers *PeerIDs* which initially contains the identifiers of all *GS*'s partners (provided by *CL*). When a partner *p* receives the Sum_Update message, it first merges *NewGS* with its local summary and removes its identifier from *PeerIDs*. Then, it sends the message to another partner chosen from *PeerIDs*. If *p* is the last visited peer (i.e., *PeerIDs* is empty), it updates the summary data: *GS* is replaced by the new version *NewGS*, and all the freshness values in *CL* are reset to zero. This strategy avoids conflicts and guarantees a high availability of the summary data, since only one update operation is performed by the last visited partner.

3.3.3 Peer Dynamicity

In P2P systems, another crucial issue is to maintain the data indexes against network changes. Besides the freshness of summary descriptions, the availability of the original data sources should be also taken into account, given the dynamic behavior of peers.

In unstructured P2P systems, when a new peer *p* joins the system, it contacts some existing peers to determine the set of its neighbors. If one of these neighbors is a partner peer, *p* sends its local summary *LS* to the corresponding summary peer *SP*, and thus becomes a new partner in the *SP*'s domain. When a partner peer *p* decides to leave the system, it first sets its freshness value *v* to two in the cooperation list, through a push message. This value reminds the participation of the disconnected peer *p* to the corresponding global summary, but also indicates the unavailability of the original data. Here, there are two alternatives to deal with such a freshness value. First, one can keep the data descriptions and use it, when a query is approximately answered using the global summary. A second alternative consists in considering the data descriptions as expired, since the original data are not accessible. Thus, a partner departure will accelerate the summary update initiating. In [21], the authors have adopted the second alternative and consider only a 1-bit freshness value *v*: a value 0 to indicate the freshness of data descriptions, and a value 1 to indicate either their expiration or their unavailability.

However, if peer *p* failed, it could not notify its summary peer by its departure. In that case, its data descriptions will remain in the global summary until a new summary update is executed. The update algorithm does not require the participation of a disconnected peer. The global summary *GS* is reconstructed, and descriptions of unavailable data will be then omitted.

Now, when a summary peer *SP* decides to leave the system, it sends a release message to all its partners using the cooperation list. Upon receiving such a message, a partner *p* makes a selective walk to find a new summary peer. However, if *SP* failed, it could not notify its partners. A partner *p* who has tried to send push or query messages to *SP* will detect its departure and thus search for a new one.

4 Query Processing

We describe now how a query *Q*, posed at a peer *p*, is processed. The adopted approach consists in querying at first the global summary *GS* that is available to peer *p*. Thus, peer *p* first sends *Q* to the summary peer *SP* of its domain, which then proceeds to query the corresponding global summary *GS*. The type of the query, precise or flexible (when using linguistic terms), and the nature of the returned results allow to distinguish four cases of summary querying. These cases are illustrated when presenting the two phases of the summary querying mechanism: 1) query reformulation and 2) query evaluation.

4.1 Query Reformulation

First, a precise query Q must be rewritten into a flexible query $Q^{'}$ in order to be handled by the summary querying process. For instance, consider the following query Q on the Patient relation in Table 1:

```
select age from Patient where sex = ''female''
and BMI < 19 and disease = ''anorexia''
```

This phase replaces the original value of each selection predicate by the corresponding descriptors defined in the Background Knowledge (*BK*). Therefore, the above query is transformed to $Q^{'}$:

```
select age from Patient where sex = ''female'' and
BMI in {underweight,normal} and disease = ''anorexia''
```

The mechanism that assists this query rewriting phase is already defined and used in the mapping service of the summarization process.

Let QS (resp.$QS^{'}$) be the *Query Scope* of query Q (resp.$Q^{'}$) in the domain, that is, the set of peers that should be visited to answer the query. Obviously, the query reformulation phase may induce false positives in query results. To illustrate, a patient having a BMI value of 20 could be returned as an answer to the query $Q^{'}$, while the selection predicate on the attribute `BMI` of the original query Q is not satisfied. However, false negatives cannot occur, which is expressed by the following inclusion: $QS \subseteq QS^{'}$.

In [21], a user query is supposed to be directly formulated using descriptors defined in the BK (i.e., $Q = Q^{'}$). As we discussed in the introduction of this work, a doctor that participates to a given medical collaboration, may ask queries like "the *age* of *female* patients diagnosed with *anorexia* and having an *underweight* or *normal BMI*". This assumption eliminates potential false positives that may result from query rewriting. Note that an interface has been defined to allow users to formulate their queries, using linguistic terms, without having knowledge of the pseudo-code language used in the querying mechanism. To illustrate the formulation of queries, recall the following notations:

- R: the schema of the summarized relation
- A: the set of attributes of relation R ($|A| = n$)
- A_i: The ith attribute of relation R

- t: a tuple of the relation R
- z: a summary node of the summary hierarchy S built over the relation R

In the query Q, we distinguish the set X of attributes whose values x are specified by the query predicates, from the set Y of attributes on which the query is projected. In other terms, the general form of query Q is given by: *Select Y from R where* $X = x$.

For each attribute $A_i \in X$, a "*required characteristic*" is defined as being a linguistic term that appears in the query for attribute A_i. The set C_i of these required characteristics is given by: $C_i = \left\{d_i^1, d_i^2, \ldots, d_i^m\right\}$. By extension, the characterization of query Q is the set of k characterizations defined on the attributes of set X: $C = \{C_1, \ldots, C_k\}$, where $k = |X|$. In our example: $X = \{sex, BMI, disease\}$, $Y = \{age\}$, $C_{sex} = \{female\}$, $C_{BMI} = \{underweight, normal\}$, $C_{disease} = \{anorexia\}$, and $C = \{C_{sex}, C_{BMI}, C_{disease}\}$.

Finally, a logical proposition P is associated to a query characterization such that:

$$P = \bigwedge_{i=1}^{k} (\bigvee_{j=1}^{m_i} d_i^j)$$

Intuitively, the presence of multiple required characteristics on a given attribute denotes an alternative, and thus the intra-attribute logical connector is disjunctive (i.e., $C_i = \bigvee_{j=1}^{m_i} d_i^j$). However, the presence of simultaneous characterizations of different attributes implies that the inter-attribute connector is conjunctive (i.e., $P = \bigwedge_{i=1}^{k} (C_i)$). In our example, the query Q is transformed to the proposition *P= (female) AND (underweight OR normal) AND (anorexia)*. In fact, the proposition P represents the canonical form of the query, which will be evaluated by the summary querying process.

4.2 Query Evaluation

First of all, to eliminate any ambiguity, it is worthy to recall that all along our work, we designate by (local/global) summary a hierarchy of summary nodes whose structure has been described in Section 2.4.

This phase deals with matching the set of summary nodes organized in the hierarchy S, against the query Q. A valuation function has been defined to valuate the corresponding proposition P in the context of a summary node z, and thus to determine if z is considered as a result. Then, a selection algorithm performs a fast exploration of the hierarchy and returns the set Z_Q of most abstract summary nodes that satisfy the query. For more details see [26].

Once Z_Q determined, the evaluation process is able to achieve two distinct tasks: (1) Summary answering, and (2) Peer localization.

4.2.1 Approximate Answering

A distinctive feature of the approach presented in this chapter is that a query can be processed entirely in the summary domain. An approximate answer can be provided from summary descriptions, without having to access original database records.

In the result set Z_Q, distinct summary nodes may answer the query in different manners. For instance, the set Z_Q contains summary nodes in which the attribute *BMI* is described either by *underweight*, or *normal*, or even by the both descriptors. A classification task has been defined in order to regroup the summary nodes in Z_Q according to their characteristics vis-a-vis the query: summary nodes that have the same required characteristics on all predicates (e.g., *sex*, *BMI* and *disease*) form a class. The aggregation in a given class is a union of descriptors: for each attribute in the selection set Y (i.e., *age*), the querying process supplies a set of descriptors which characterize the summary nodes that answer the query through the same interpretation [26]. For example, according to Table 2, the output set obtained for the different classes found in z_Q is:

- $\{female, underweight, anorexia\} \Rightarrow age = \{adolescent, young\ adult\}$
- $\{female, normal, anorexia\} \Rightarrow age = \{adolescent\}$

In other words, *all* female patients diagnosed with *anorexia* and having an *underweight* or *normal BMI* are *adolescent* and *young* girls. Moreover, the information provided by the tuple count columns may further indicate that almost all of these girls are *adolescent* (i.e., a value of 2.5 for *adolescent*, and a value of 0.5 for *young adult*).

4.2.2 Peer Localization

In centralized environments, returning exact answers to the flexible query Q is simply an extension of the mechanism of returning approximate answers. The extent R_z of a summary node z (see Definition 1) provides the original records that are described by that summary intention. Thus, such a functionality only requires to connect to the underlying database in order to retrieve data.

However, in P2P systems, a summary node may describe tuples that are highly distributed in the network. Therefore, the query Q should be first forwarded to the relevant peers whose databases contain the result tuples. In Definition 3, the *Peer-extent* dimension has been added to a summary node structure, in order to provide the set of peers P_z having data described by its intent.

Here, one can assume that in a given local summary, the extents of its summary nodes are maintained to be able to retrieve raw data from the underlying database. While in a global summary this information is omitted and only the *Peer-extents* of its summary nodes are maintained to be able to reach the relevant peers in the distributed network.

Based on the *Peer-extents* of summary nodes in the set Z_Q, the set P_Q of relevant peers is defined as follows: $P_Q = \{\cup_{z \in Z_Q} P_z\}$. The query Q is directly propagated to these relevant peers. However, the efficiency of this query routing depends on the completeness and the freshness of summaries, since stale answers may occur in query results. We define a *False Positive* as the case in which a peer p belongs to P_Q and there is actually no data in the p source that satisfies Q (i.e., $p \notin QS$). A *False Negative* is the reverse case in which a p does not belong to P_Q, whereas there exists at least one tuple in the p data source that satisfies Q (i.e., $p \in QS$).
In the case where exact answers are required, suppose now that processing a query Q in a given domain d_i returns C_i results, while the user requires C_t results. We note that, if C_t is less than the total number of results available in the network, Q is said to be a *partial-lookup* query. Otherwise, it is a *total-lookup* query. Obviously, when C_i is less than C_t, the query should be propagated to other domains. To this end, the following variation of the flooding mechanism is adopted.

Let P_i the subset of peers that have answered the query Q in the domain d_i: $|P_i| = (1 - FP) \cdot |P_Q|$, where FP is the fraction of false positives in query results. The query hit in the domain is given by: $(|P_i| \,/\, |d_i|)$. As shown by many studies, the existing P2P networks have small-world features [4]. In such a context, users tend to work in groups. A group of users, although not always located in geographical proximity, tends to use the same set of resources (i.e., *group locality* property). Thus, the probability of finding answers to query Q in the neighborhood of a relevant peer in P_i is supposed to be high (i.e., query answers are supposed to be *nearby*). This probability is also high in the neighborhood of the originator peer p since some of its neighbors may be interested in the same data, and thus have cached answers to similar queries. Such assumptions are even more relevant in the context of interest-based clustered networks. Therefore, the summary peer SP_i of domain d_i sends a flooding request to each peer in P_i as well as to peer p. Upon receiving this request, each of those peers sends the query to its neighbors that do not belong to its domain, with a limited value of TTL. Once a new domain is reached or TTL becomes zero, the query is stopped. Besides, the summary peer SP sends the request to the set of summary peers it knows in the system. This will accelerate covering a large number of domains. In each visited domain, the query is processed as described above. When the number of query results becomes sufficient (i.e., larger than C_t), or the network is entirely covered, the query routing is terminated.

5 Performance Evaluation

In this section, we devise a simple model for the summary management cost. Then, we evaluate that model by simulation using the BRITE topology generator and SimJava.

5.1 Cost Model

A critical issue in summary management is to trade off the summary updating cost against the benefits obtained for queries.

5.1.1 Summary Update Cost

Here, the first undertaking is to optimize the update cost while taking into account *query accuracy*. In the next section, we discuss query accuracy which is measured in terms of the percentage of false positives and false negatives in query results. The cost of updating summaries is divided into: usage of peer resources, i.e., time cost and storage cost, and the traffic overhead generated in the network.

Time Cost

A unique feature of SAINTETIQ is that the changes in the database are reflected through an incremental maintenance of the summary hierarchy. The time complexity of the summarization process is in $O(K)$ where K is the number of cells to be incorporated in that hierarchy [22]. For a global summary update, we are concerned with the complexity of merging summaries. The MERGING method that has been proposed is based on the SAINTETIQ engine. This method consists in incorporating the leaves L_z of a given summary hierarchy S_1 into an another S_2, using the same algorithm described by the SAINTETIQ summarization service (referenced in Section 2.3.2). It has been proved that the complexity C_{M12} of the MERGING(S_1, S_2) process is constant w.r.t the number of tuples [18]. More precisely, C_{M12} depends on the maximum number of leaves of S_1 to incorporate into S_2. However, the number of leaves in a summary hierarchy is not an issue because it can be adjusted by the user according to the desired precision. A detailed Background Knowledge (BK) will lead to a greater precision in summary description, with the natural consequence of a larger summary. Moreover, the hierarchy is constructed in a top-down approach and it is possible to set the summarization process so that the leaves have any desired precision.

Storage Cost

We denote by k the average size of a summary node z. In the average-case assumption, there are $\sum_{i=0}^{d} B^i = (B^{d+1} - 1)/(B - 1)$ nodes in a B-arity tree with d, the average depth of the hierarchy. Thus the average space requirement is given by: $C_m = k.(B^{d+1} - 1)/(B - 1)$. Based on real tests, $k = 512$ bytes gives a rough estimation of the space required for each summary. An important issue is that the size of the hierarchy is quite related to its stabilization (i.e., B and d). As more cells are processed, the need to adapt the hierarchy decreases and incorporating a new cell may consist only in sorting a tree. Hence, the structure of the hierarchy remains stable and no additional space is required. On the other hand, when we merge two hierarchies S_1 and S_2 having sizes of C_{m1} and C_{m2} respectively, the size of the resultant hierarchy is always in the order of the $max\,(C_{m1}, C_{m2})$. However, the size of a summary hierarchy is limited to a maximum value which corresponds to a maximum number of leaves that cover all the possible combinations of the BK descriptors. Thus, storing global summaries does not impose high space storage constraints on the summary peers.

According to the above discussion, the usage of peer resources is optimized by the summarization process itself, and the distribution of summary merging while updating a global summary. Thus, the focus in [21] is on the traffic overhead generated in the P2P network.

Network Traffic

Recall that there are two types of exchanged messages: *push* and *update* messages. Let local summaries have an average lifetime of L seconds in a given global summary. Once L expired, the node sends a (push) message to update its freshness value v in the cooperation list CL. The summary update algorithm is then initiated whenever the following condition is satisfied:

$$\sum_{v \in CL} v/|CL| \geq \alpha$$

where α is a threshold that represents the ratio of old descriptions tolerated in the global summary. While updating, only one message is propagated among all partner peers until the new global summary version is stored at the summary peer SP. Let F_{rec} be the reconciliation frequency. The update cost is:

$$C_{up} = 1/L + F_{rec} \;\; messages\ per\ node\ per\ second \tag{2}$$

In this expression, $1/L$ represents the number of push messages which depends either on the modification rate issued on local summaries or the connection/disconnection rate of peers in the system. Higher is the rate, lower is the lifetime L, and thus a large number of push messages are entailed in the system. F_{rec} represents the number of update messages which depends on the value of α. This threshold is the system parameter that provides a trade-off between the cost of summary updating and query accuracy. If α is large, the update cost is low since a low frequency of update is required, but query results may be less accurate due both to false positives stemming from the descriptions of non existent data, and to false negatives due to the loss of relevant data descriptions whereas they are available in the system. If α is small, the update cost is high but there are few query results that refer to data no longer in the system, and nearly all available results are returned by the query.

5.1.2 Query Cost

When a query Q is posed at a peer p, it is first matched against the global summary available at the summary peer SP of its domain, to determine the set of relevant peers P_Q. Then, Q is directly propagated to those peers. The query cost in a domain d is given by:

$$C_d = (1 + |P_Q| + (1 - FP) \cdot |P_Q|) \ \ messages,$$

where $(1 - FP) \cdot |P_Q|$ represents the query responses messages (i.e., query hit in the domain).

Here we note that, the cooperation list CL associated with a global summary provides information about the relevance of each database description. Thus, it gives more flexibility in tuning the *recall/precision* trade-off of the query answers in domain d. The set of all partner peers P_H in CL can be divided into two subsets: $P_{old} = \{p \in P_H \mid p.v = 1\}$, the set of peers whose descriptions are considered old, and $P_{fresh} = \{p \in P_H \mid p.v = 0\}$ the set of peers whose descriptions are considered fresh according to their current data instances. Thus, if a query Q is propagated only to the set $V = P_Q \cap P_{fresh}$, then precision is maximum since all visited peers are certainly matching peers (no false positives), but recall depends on the fraction of false negatives in query results that could be returned by the set of excluded peers $P_Q \backslash P_{fresh}$. On the contrary, if the query Q is propagated to the extended set $V = P_Q \cup P_{old}$, the recall value is maximum since all matching peers are visited (no false negatives), but precision depends on the fraction of false positives in query results that are returned by the set of peers P_{old}.

Now we consider that the selectivity of query Q is very high, such that each relevant peer has only one result tuple. Thus, when a user requires C_t tuples, we have to visit C_t relevant peers. The cost of inter-domain query flooding is given by:

$$C_f = ((1 - FP) \cdot |P_Q| + 2) \cdot \sum_{i=1}^{TTL} k^i \ \ messages,$$

where k is the average degree value (e.g., average degree of 3.5, similar to Gnutella-type graphs). Remember that, the set of relevant peers who have answered the query (i.e., $(1 - FP) \cdot |P_Q|$), the originator and the summary peers participate to query flooding. In this expression, we consider that a summary peer has on average k long-range links to k summary peers. As a consequence, the total cost of a query is:

$$C_Q = C_d \cdot \frac{C_t}{(1-FP) \cdot |P_Q|} + C_f \cdot (1 - \frac{C_t}{(1-FP) \cdot |P_Q|}) \qquad (3)$$

In this expression, the term $C_t/((1 - FP) \cdot |P_Q|)$ represents the number of domains that should be visited. For example, when $C_t = ((1 - FP) \cdot |P_Q|)$, one domain is sufficient and no query flooding is required.

5.2 Simulation

The performance of the proposed solutions is evaluated through simulation, based on the above cost model. First, we describe the simulation setup. Then we present simulation results to evaluate various performance dimensions and parameters: scale up, query accuracy, effect of the freshness threshold α.

5.2.1 Simulation Setup

In [21], the SimJava package [10] and the BRITE universal topology generator [1] are used to simulate a power law P2P network, with an average degree of 4. The simulation parameters are shown in Table 3.

Table 3 Simulation parameters

Parameter	value
Network configuration	
Local summary lifetime L	Skewed distribution, Mean=3h, Median=1h
Number of peers n	16–5000
Workload configuration	
Number of queries q	200
Matching nodes/query $hits$	10%
System parameter	
Freshness threshold α	0.1–0.8

In large-scale P2P file-sharing systems, a user connects mainly to download some data and may then leave the system without any constraint. As reported in [25], these systems are highly dynamic and node lifetimes are measured in hours. As such, data indexes should be mainly maintained against network changes. In collaborative database applications, however, the P2P system is supposed to be more stable. Here, the data indexes should be mainly maintained against data changes, since the shared data may be submitted to a significant modification rate.

As stated before, the work presented in this chapter targets collaborative applications sharing semantically rich data. In simulation tests, synthetic data have been used since it was difficult to obtain/use real, highly distributed P2P databases. Future works aim to employ the summary proposal in the context of astronomical applications, which seem to be attractive because of the huge amount of information stored into the databases. Thus, to provide meaningful results, the performance of the proposed solutions has been evaluated in worst contexts where the data are highly updated. Local summary lifetimes are supposed to follow a skewed distribution with a mean lifetime of 3 hours, and a median lifetime of 60 minutes. Note that summary lifetimes of hours means that the underlying data is submitted to a very high modification rate, since the summaries are supposed to be more stable than original data (as discussed in Section 3.3.2). Besides, such lifetime values allow to predict the performance in large-scale data sharing P2P systems where the rate of node departure/arrival dominates the global summary update initiating.

The tests have been made on moderated-size P2P networks, i.e., the number of peers varies between 16 and 5000. In the used query workload, the query rate is 0.00083 queries per node per second (one query per node per 20 minutes) as suggested in [7]. Each query is matched by 10% of the total number of peers. Finally, our system parameter α varies between 0.1 and 0.8.

5.2.2 Update Cost

This first set of experiments quantifies the trade-off between query accuracy and the cost of updating a global summary in a given domain. Figure 5 depicts the fraction of stale answers in query results for different values of the threshold α. Here, the worst case is illustrated. For each partner peer p having a freshness value equal to 1, if it is selected in the set P_Q then it is considered as false positive. Otherwise, it is considered as false negative. However, this is not the real case. Though it has a freshness value equal to 1, the peer p does not incur stale answers unless its database is changed relative to the posed query Q. Thus, Fig. 5 shows the worst, but very reasonable values. For instance, the fraction of stale answers is limited to 11% for a network of 500 peers when the threshold α is set to 0.3 (30% of the peers are tolerated to have old/non existent descriptions).

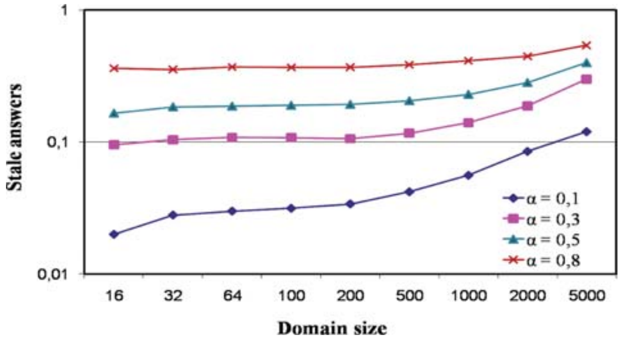


Fig. 5 Stale answers vs. domain size

As mentioned in Section 5.1.2, if we choose to propagate the query only to the set $V = P_Q \cap P_{fresh}$ we eliminate the possible false positives in query results. However, this may lead to additional false negatives. Figure 6 shows the fraction of false

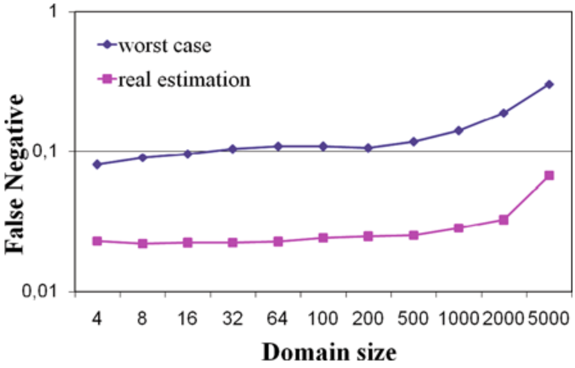


Fig. 6 False negative vs. domain size

negatives in function of the domain size. Here, for a peer having a freshness value equal to 1, the probability of the database modification relative to the issued query is taken into account. We see that the fraction of false negatives is limited to 3% for a domain size less than 2000 (i.e., network size less than 8000). The real estimation of stale answers shows a reduction by a factor of 4.5 with respect to the preceded values.

Figure 7 depicts the update cost in function of the domain size, and this for two threshold values. The total number of messages increases with the domain size, but not surprisingly, the number of messages per node remains almost the same. In the update cost Equation(2), the number of push messages for a given peer is independent of domain size. More interestingly, when the threshold value decreases (from 0.8 to 0.3) we notice a little cost increasing of 1.2 on average. For a domain of 1000 peers, the update cost increases from 0.01056 to 0.01296 messages per node per minute (not shown in figure). However, a small value of the threshold α allows to reduce significantly the fraction of stale answers in query results, as seen in Fig. 5.

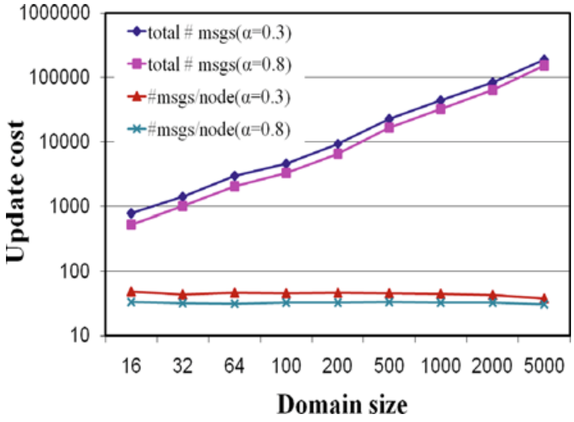


Fig. 7 Number of messages vs. domain size

We conclude therefore that tuning the system parameter, i.e., the threshold α, do not incur additional traffic overhead, while improving query accuracy.

5.2.3 Query Cost

This set of experiments compares the algorithm for query processing against centralized-index and pure non-index/flooding algorithms. A centralized-index approach is very efficient since a single message allows locating relevant data. However, a central index is vulnerable to attack and it is difficult to keep it up-to-date. Flooding algorithms are very used in real life, due to their simplicity and the lack of complex state information at each peer. A pure flooding algorithm consists in broadcasting the query in the network till a stop condition is satisfied, which may lead to a very high query execution cost. Here, the flooding is limited by a value 3 of TTL.

According to Table 3, the query hit is 10% of the total number of peers. For the query processing approach, which is mainly based on summary querying (SQ), it is considered that each visited domain provides 10% of the number of relevant peers (i.e., 1% of the network size). In other words, we should visit 10 domains for each query Q. From Equation(3), we obtain: $C_Q = (10 \cdot C_d + 9 \cdot C_f)$ messages. Figure 8 depicts the number of exchanged messages to process a query Q, in function of the total number of peers. The centralized-index algorithm shows the best results that can be expected from any query processing algorithm, when the index is complete and consistent, i.e., the index covers the totality of data available in the system, and there are no stale answers in query results. In that case, the query cost is: $C_Q = 1 + 2 \cdot ((0.1) \cdot n)$ messages, which includes the query message sent to the index, the query messages sent to the relevant peers and the query response messages returned to the originator peer p.

In Fig. 8, we observe that the algorithm SQ shows good results by significantly reducing the number of exchanged messages, in comparison with a pure query flooding algorithm. For instance, the query cost is reduced by a factor of 3.5 for a network of 2000 peers, and this reduction becomes more important with a larger-sized net-

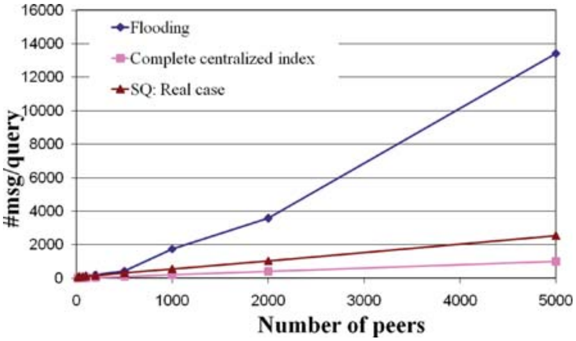


Fig. 8 Query cost vs. number of peers

work. We note that in these tests, the worst case of the *SQ* algorithm is considered, in which the fraction of stale answers of Fig. 5 occurs in query results (for $\alpha = 0.3$).

6 Related Work

Current works on P2P systems aim to employ *content-based* routing strategies, since the content of data can be exploited to more precisely guide query propagation. These strategies require gathering information about the content of peer's data. However, the limits on network bandwidth and peer storage, as well as the increasing amount of shared data, call for effective summarization techniques. These techniques allow exchanging compact information on peer's content, rather than exchanging original data in the network.

Existing P2P systems have used keyword-based approaches to summarize text documents. For instance, in [3] documents are summarized by keyword vectors, and each node knows an approximate number of documents matching a given keyword that can be retrieved through each outgoing link (i.e., Routing Indices *RI*s). Although the search is very bandwidth-efficient, *RI*s require flooding in order to be created and updated, so the method is not suitable for highly dynamic networks. Other works (e.g., [9]) investigate Vector Space Model (VSM) and build *Inverted Indexes* for every keyword to cluster content. In this model, documents and queries are both represented by a vector space corresponding to a list of orthogonal term vectors called *Term Vector*. The drawback of VSM is its high cost of vector representations in case of P2P churns. In [13], a *semantic-based* content search consists in combining VSM to Latent Semantic Index (LSI) model to find semantically relevant documents in a P2P network. This work is based on hierarchical summary structure over hybrid P2P architecture, which is closely related to what we are presenting in this paper. However, instead of representing documents by vector models, we describe structured data (i.e., relational database) by synthetic summaries that respect the original data schema.

To the best of our knowledge, none of the P2P summarization techniques allows for an *approximate query answering*. All works have focused on facilitating content-based query routing, in order to improve search efficiency. We believe that the novelty of the approach proposed in this chapter relies on the double exploitation of the employed data summaries. These summaries allow for a semantic-based query routing, but also to approximately answer the query using their intentional descriptions.

7 Conclusion

This chapter proposed a model for summary management in hierarchical P2P systems. The innovation of this proposal consists in combining the P2P and database summarization paradigms, in order to support data sharing on a world wide scale.

The database summarization approach that we proposed provides efficient techniques for data localization as well as for data description in P2P systems. In fact, the produced summaries are semantic indexes that support locating relevant data based on their content. Besides, an important feature is that these summaries are compact data descriptions that can approximately answer a query without retrieving original records from huge, highly distributed databases.

This work made the following contributions. First, an appropriate summary model for hybrid P2P systems is proposed. Then, efficient algorithms for summary management are described, and a query processing mechanism that relies on summary querying is presented. Performance evaluation showed that the cost of query routing in the context of summaries is significantly reduced in comparison with flooding algorithms, without incurring high costs of summary maintenance.

In the summary model for hierarchical P2P networks [21], it has been assumed that peers are grouped around high-connectivity nodes. However, to better exploit the *data locality* property of most of current P2P systems, future works intend to study the organization of nodes based on similarity between their summaries. Ongoing works on the SAINTETIQ model aim to define a similarity distance between summaries. However, such a proposal requires a complete study of the obtained clustering scheme, i.e., the number of clusters, the cluster sizes, the stability against node dynamics.

Besides, one can notice that the proposed summaries can serve data anonymization purposes. To illustrate, a given hospital which is participating to a medical collaborative application may first perform an obfuscation of its database through the generation of a local summary. Then, the different participants exchange and share such intelligible summaries, represented in a higher abstraction level. This allow to learn the "essential" from a given database without revealing personal information about patients. A future work plans to study how these summaries can be also used to make a data source anonymization. In fact, due to the peer-extent information, a peer may reveal the source providing a given summary while executing the summary update algorithm. In some cases, participants may also prefer hiding the characterization of their data. Hence, a given participant can exploit data summaries of other participants without being able to precise which source is providing a specified summary.

References

1. http://www.cs.bu.edu/brite/
2. http://www.snomed.org/snomedct
3. A.Crespo, H.G.Molina: Routing indices for peer-to-peer systems. In: Proc. of the 28 tn Conference on Distributed Computing Systems (2002)
4. A.Iamnitchi, M.Ripeanu, I.Foster: Locating data in (small-world?) peer-to-peer scientific collaborations. In: IPTPS, pp. 232–241 (2002)
5. A.Shoshani: OLAP and statistical databases: Similarities and differences. In: Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 185–196. ACM Press (1997)

6. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975). DOI http://doi.acm.org/10.1145/361002.361007
7. B.Yang, H.G.Molina: Comparing hybrid peer-to-peer systems. In: Proc VLDB (2001)
8. Comer, D.: Ubiquitous b-tree. ACM Comput. Surv. **11**(2), 121–137 (1979). DOI http://doi.acm.org/10.1145/356770.356776
9. F.Cuenca-Acuna, C.Peery, R.Martin, T.Nguyen: Planetp: Using gossiping to build content addressable peer-to-peer information sharing communities. In: HPDC-12 (2003)
10. F.Howell, R.McNab: Simjava: a discrete event simulation package for java with the applications in computer systems modeling. In: Int. Conf on Web-based Modelling and Simulation, San Diego CA, Society for Computer Simulation (1998)
11. Han, J., Fu, Y., Huang, Y., Cai, Y., Cercone, N.: Dblearn: A system prototype for knowledge discovery in relational databases. In: Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 1994, p. 516. ACM Press (1994)
12. H.Jagadish, R.Ng, B.Ooi, A.Tung: Itcompress: An iterative semantic compression algorithm. In: 20th International Conference on Data Engineering, p. 646 (2004)
13. H.Shen, Y.Shu, B.Yu: Efficient semantic-based content search in p2p network. IEEE Trans. Knowl. Data Eng. **16**(7) (2004)
14. Ioannidis, Y.: The history of histograms (abridged). In: VLDB '2003: Proceedings of the 29th international conference on Very large data bases, pp. 19–30. VLDB Endowment (2003)
15. I.Tartinov, *et al*: The Piazza peer data management project. In: SIGMOD (2003)
16. K.Thompson, P.Langley: Concept formation in structured domains. In: Concept formation: Knowledge and experience in unsupervised learning, pp. 127–161. Morgan Kaufmann
17. L.A.Zadeh: Concept of a linguistic variable and its application to approximate reasoning-I. Inf. Syst. **8**, 199–249 (1975)
18. M.Bechchi, G.Raschia, N.Mouaddib: Merging distributed database summaries. In: ACM Sixteenth Conference on Information and Knowledge Management (CIKM) (2007)
19. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes in C (2nd ed.): the art of scientific computing. Cambridge University Press, New York, NY, USA (1992)
20. R.Akbarinia, V.Martins, E.Pacitti, P.Valduriez: Design and implementation of appa. In: Global Data Management (Eds. R. Baldoni, G. Cortese and F. Davide). IOS press (2006)
21. R.Hayek, G.Raschia, P.Valduriez, N.Mouaddib: Summary management in p2p systems. In: EDBT, pp. 16–25 (2008)
22. R.Saint-Paul, G.Raschia, N.Mouaddib: General purpose database summarization. In: Proc VLDB, pp. 733–744 (2005)
23. Ruspini, E.H.: A new approach to clustering. Inf. Control **15**, 22–32 (1969)
24. S.Babu, G.Minos, R.Rajeev: Spartan: A model-based semantic compression system for massive data tables. In: Proc. of the 2001 ACM Intl. Conf. on Management of Data (SIGMOD), pp. 283–295 (2001)
25. S.Saroiu, P.Gummadi, S.Gribble: A measurement study of peer-to-peer file sharing systems. In: Proc of Multimedia Computing and Networking (MMCN) (2002)
26. W.A.Voglozin, G.Raschia, L.Ughetto, N.Mouaddib: Querying the SAINTETIQ summaries–a first attempt. In: Int.Conf.On Flexible Query Answering Systems (FQAS) (2004)
27. W.Nejdl, W.Siberski: Design issues and challenges for rdf- and schema-based peer-to-peer systems. SIGMOD Record **32**, 2003 (2003)

Case Study: Scoop for Partial Read from P2P Database

Farnoush Banaei-Kashani and Cyrus Shahabi

Abstract In this paper we propose *Scoop*, a mechanism to implement the "partial read operation" for peer-to-peer databases. A peer-to-peer database is a database that its relations are horizontally fragmented and distributed among the nodes of a peer-to-peer network. The partial read operation is a data retrieval operation required for approximate query processing in peer-to-peer databases. A partial read operation answers to β-queries: given $\beta \in [0,1]$ and a relation R, a fraction β of the tuples in R must be retrieved from the database to answer a β-query. Despite the simplicity of the β-query, due to the distributed, evolving and autonomous nature of the peer-to-peer databases correct and efficient implementation of the partial read operation is challenging. Scoop is designed based on an epidemic dissemination algorithm. We model the epidemic dissemination as a percolation problem and by rigorous percolation analysis tune Scoop per-query and on-the-fly to answer β-queries correctly and efficiently. We prove the correctness of Scoop by theoretical analysis, and verify the efficiency of Scoop in terms of query cost and query time via extensive simulation.

Farnoush Banaei-Kashani
University of Southern California, Department of Computer Science, Los Angeles, CA 90089-0781, USA e-mail: banaeika@usc.edu

Cyrus Shahabi
University of Southern California, Department of Computer Science, Los Angeles, CA 90089-0781, USA, e-mail: shahabi@usc.edu

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_21, © Springer Science+Business Media, LLC 2010

1 Introduction

1.1 Motivation and Problem Definition

A peer-to-peer database is a database that its relations are horizontally fragmented and distributed among the nodes of a peer-to-peer network. Each node is potentially both a client and a server of the database. Peer-to-peer databases are distinguished from the more generic distributed databases by the sheer size of the database in terms of the participating nodes, as well as higher dynamism of the data, nodes, and the interconnection network. Correspondingly, these distinctive features necessitate provision of more scalable and dynamic data management mechanisms for peer-to-peer databases (as compared to those for the generic distributed databases). Peer-to-peer databases lie at the intersection of the database and networking research fields, and currently numerous researchers and focus groups from both communities are actively working to provide the required data management capabilities for this new generation of the distributed databases.[1]

Here, we focus on unstructured peer-to-peer databases. With these databases, the process of query answering is comparable to that in regular unindexed databases. Due to the considerable amount of churn in unstructured peer-to-peer databases, construction and maintenance of distributed index structures (e.g., a DHT [24, 30]) are not cost-efficient. Without index, to answer a user query, first the query is disseminated throughout the network to scan the database and retrieve the data relevant to the query (analogous to the sequential scan in regular unindexed databases). Subsequently, the retrieved data are processed locally at the querying node to answer the user query.[2] In this paper, we introduce a basic data retrieval operation, termed the partial read operation, to execute the first step of the query answering, specifically as required for *approximate* query answering discussed below.

Approximate query answering is a desirable generalization of exact query answering, arguably essential for efficient query answering in unstructured peer-to-peer databases. With these unindexable databases, returning exact answers (i.e., the complete result-set) requires a complete scan of the entire database for each query. Considering the large size of the peer-to-peer databases and the abundance of the database users with numerous queries, answering all queries with exact result is obviously too costly to scale. Besides, with most of the peer-to-peer database applications the workload mainly consists of exploratory queries for which exact results are unnecessary and redundant [11]. With approximate query answering, queries are assumed to be satisfied by an approximate/partial result, which is a subset of the complete result-set of the query. Along with the query, users specify the

[1] For instance, see the publications at the recurrent DBISP2P workshop (held in conjunction with VLDB) and the NetDB workshop (held in conjunction with ICDE).

[2] Here, by "query" we refer to user queries such as SQL queries. In this paper, we also use "query" to refer to β-*queries*, i.e., requests for retrieving the tuples of the base relations stored in the database, as we later define formally. The intended meaning is apparent from the context.

required “completeness” of the approximate result that satisfies the query. Correspondingly, the database system can reduce the overhead of the data retrieval and query processing to the least sufficient to satisfy the query, effectively enhancing the efficiency of the query answering by eliminating the redundancy of the result. If required, still a query can be answered exactly. An exact query is simply a specific case of approximate query requiring the most complete result-set, the exact result.

Formally, an approximate query is defined as an arbitrary query with a user-defined *completeness ratio* $\varepsilon \in [0,1]$; any fraction ε of the complete result-set X, noted as $X^{(\varepsilon)}$, is sufficient to satisfy the approximate query. An exact query is an approximate query with $\varepsilon = 1$. Approximate queries are also called *partial* query. To answer a partial query, first the peer-to-peer database should be sampled to retrieve a certain fraction β_i (also called the completeness ratio, in lack of other conveying terms) from each input relation R_i of the query. Subsequently, using partial query processing techniques such as [23, 31] the retrieved data are processed locally at the querying node to generate the final partial result-set $X^{(\varepsilon)}$. The main challenge with answering partial queries is with devising a basic sampling operation that given any $\beta \in [0,1]$, retrieves a fraction β of an input relation R (i.e., retrieving a partial relation noted as $R^{(\beta)}$), where R is horizontally fragmented and distributed among the nodes of the peer-to-peer database. We term such a sampling operation the *partial read operation*. Partial read is a nonprobability sampling operation, because it is not expected to guarantee randomness of the sample retrieved from the database. It is informative to note that with *structured* peer-to-peer databases, since the placement of the data objects is a priori known, partial read is trivial to implement. In this paper, we focus on implementing the partial read operation in *unstructured* peer-to-peer databases.

To be applicable, the partial read operation must be both correct and efficient. A partial read is correct if $|R'|/|R| \geq \beta$, where R' is the retrieved portion of the input relation R. This is a sufficient condition for correctness of the partial read. However, ideally the redundancy of the data retrieval is minimal when $|R'|/|R| = |R^{(\beta)}|/|R| = \beta$. A peer-to-peer database is a fragmented database which is randomly distributed among a variable set of nodes with unknown population. Under such circumstances, to be correct the partial read operation must implement a distributed query dissemination scheme to visit sufficient number of nodes and retrieve sufficient number of tuples, such that in expectation the collective set of the retrieved tuples R' satisfies the correctness condition with high probability. On the other hand, the efficiency of the partial read operation is defined in terms of the query time (or sampling time) and query cost (or sampling cost), which respectively map to the total communication time and communication cost of disseminating the query for distributed data retrieval. There is a trade-off between the query time and the query cost, with higher cost to achieve shorter time and vice versa. An efficient partial read operation satisfies the ideal case of the correctness condition with a balanced query time versus query cost. Scoop is a mechanism to implement the partial read operation correctly and efficiently.

1.2 Related Work

Data retrieval from networks has been previously studied as a search problem, but none of the proposed search mechanisms can satisfy the correctness and efficiency requirements of the partial read operation. There are two main proposals for search in unstructured peer-to-peer networks: search by flooding [18, 33] and search by random walk [1, 6, 10, 19, 20]. With both of these search mechanisms, query is disseminated throughout the network by recursive forwarding from node to node. With flooding each node that receives the query forwards it to all of its neighbors, whereas with random walk query is forwarded to only one (uniformly or non-uniformly) selected random neighbor. None of these approaches can strike a balance between the two metrics of efficiency for search, i.e., the communication time and the communication cost. Flooding is most efficient in communication time but incurs too high of redundant communication to be practical, whereas random walk is potentially more efficient in communication cost but is intolerably slow in scanning the network. In [33], a two-tier hierarchy is proposed where flooding is restricted to the supernodes at the top tier. This solution only alleviates the communication cost of flooding and the problem resurfaces as the top tier scales. In [19], using k random walkers in parallel is proposed as a way to balance the communication cost and the communication time of the query. However, this proposal does not provide any rigorous solution for selecting the value of k for optimal performance.

Previous search mechanisms are not only inefficient, but also inappropriate for executing partial read. With Top-K queries, the relevant data is ordered in terms of a preference measure, whereas partial read allows unbiased data retrieval. Also, as mentioned above the main benefit of the approximate query answering is that it allows trading off completeness of the result for better efficiency by limiting the scan of the database to a just sufficiently large fraction of the database that satisfies the β-query. To enable such a trade-off, a search mechanism that is used to implement the partial read should allow adjusting the coverage of the database (i.e., the fraction of the network nodes, and hence data objects, visited during dissemination) according to the user specified parameter β of each query. With both flooding and random walk, TTL (Time-To-Live) is the control parameter that can be used to limit the coverage of the network. However, it is not clear how one can adjust TTL according to β for sufficient coverage (where the size of the network is also unknown). TTL is often set to a fixed value, a value that is selected in an ad hoc fashion based on the average performance of the typical search queries and must be re-adjusted as the peer-to-peer database evolves. Alternatively, TTL is gradually increased to expand the coverage, each time repeating the query dissemination from the beginning, until sufficient fraction of the database is covered to answer the query. Although this scheme may result in *correct* partial read, due to the redundancy of repeating the query dissemination, query cost can even exceed that of the regular flooding. Finally, another problem specific to the flooding is that the granularity of the coverage is too coarse (the number of covered nodes grow exponentially with TTL), rendering fine adjustment of the coverage impossible.

1.3 Scoop

1.3.1 Overview

We propose *Scoop* as a sampling mechanism to implement partial read. To sample the database, beginning from the originator of the query Scoop spreads the query throughout the network. While spreading, the query inspects the nodes of the network to locate and retrieve a fraction β of the tuples of the input relation R. With Scoop, the spread of the query is modelled on epidemic dissemination of diseases in social networks.[3] By epidemic dissemination, query spreading is probabilistic, i.e., when a node receives a query, it replicates and forwards the query to each of its neighbors independently with a forwarding probability $0 \leq p \leq 1$. Therefore, a node forwards the replicas of the query to zero or more neighbors at each time. Such a query forwarding algorithm is obviously more flexible as compared to both flooding and random walk and subsumes these dissemination mechanisms. The *communication graph* of the epidemic dissemination (i.e., the subgraph of the peer-to-peer network covered by the dissemination) is sparse with small values of p. The communication graph grows larger and denser with larger values of p such that with $p = 1$ the epidemic dissemination is equivalent to regular flooding which covers the entire network.

Scoop specifically implements SIR (Susceptible-Infected-Removed), which is a classic epidemic dissemination model [12]. Our main contribution with Scoop is derivation of a closed-form formula that given a partial read request, maps the value of the completeness ratio β to an appropriate value for the forwarding probability p such that the request is correctly satisfied. Leveraging on this derivation, Scoop *on-the-fly* and *per read-request* tunes p based on β such that the communication graph grows just sufficiently large to cover a fraction of the database that satisfies the partial read request. For partial read requests with small β the communication graph is sparse and as β increases the graph becomes denser. Since both the communication cost and communication time of the sampling increase proportional to the size of the communication graph, the partial read requests with higher completeness ratio are more expensive, as expected.

1.3.2 Correctness and Efficiency

In Section 3.3, we rigorously prove that Scoop satisfies the ideal case of the correctness condition for the partial read operation. Scoop achieves correctness without a priori knowledge about the size of the network. With each value of p Scoop covers a certain fixed fraction of the network nodes *independent* of the size of the network. In other words, for the same value of p, size of the communication graph is always pro-

[3] In the literature, sometimes *gossip-based* or *rumor-based* spreading techniques are also termed epidemic techniques [5, 7, 15, 16]. Here, we are not referring to such many-to-many communication techniques, but specifically to the techniques that are modelled after disease spreading in social networks.

portional to the size of the entire network, such that its relative size (i.e., the covered fraction of the network) is fixed. This property allows answering β-queries (which expect retrieving a fixed fraction of a relation) without acquiring and maintaining global knowledge about the current size of the network. Intuitively, this is feasible because unlike flooding and random walk with which a query never dies unless it is explicitly terminated (e.g., when TTL expires), with epidemic dissemination query forwarding is probabilistic and with some non-zero probability each replica of the query may naturally die at each step. The dissemination terminates whenever all replicas of the query die. The larger the network, the more it takes for the dissemination to die and, therefore proportionally, the larger is the communication graph of the dissemination.

Scoop is also efficient, in that it strikes a balance between the communication cost and communication time of the sampling. Since epidemic dissemination is essentially a flood-based technique, as we show in Section 6, its communication time is low and comparable with that of the regular flooding. On the other hand, due to the phase transition phenomenon associated with the SIR epidemic model, for the common case of the partial read requests the communication cost of the Scoop is up to two orders of magnitude less than that of the regular flooding and comparable with the low communication cost of the random walk. Intuitively, with epidemic dissemination the dense communication graph of the regular flooding, which with numerous loops represents a large amount of redundant and duplicate query forwarding, is reduced to a sparse communication graph. With fewer loops, the sparse graph contains less redundant paths and therefore, causes less duplicate queries, while covering almost the same set of nodes. Hence, epidemic dissemination can be tuned such that the overhead of the flooding is effectively eliminated while its reachability and communication time is preserved.

It is also important to mention that Scoop is simple to implement, and since it is a randomized mechanism, it is inherently reliable to use with dynamic peer-to-peer databases. Moreover, with Scoop the larger the degree of a node, the larger its load. However, as Gribble et al. have shown in [26], with peer-to-peer databases the degree of connectivity of the nodes is positively correlated with the amount of the shared resources of the nodes. Therefore, with Scoop the load is distributed among the nodes roughly proportional to the resources of the nodes, which is a desirable load balancing property.

1.3.3 Originality

The process of epidemic disease dissemination has been previously used as a model to design other information dissemination techniques [13]. Particularly, in the networking community, epidemic dissemination is termed probabilistic flooding and is applied for search and routing in various types of networks [8, 17, 27]. We distinguish Scoop from this body of work in two ways. First, although epidemic algorithms are simple to implement, due to their randomized and distributed nature they are often difficult to analyze theoretically. For the same reason, most of the previous work restrict themselves to empirical study of the performance with results that are subject to inaccuracy and lack of generality. We employ the percolation theory to rigorously tune Scoop to its best operating point. Second, to the best of our knowledge those of the few percolation-based theoretical studies of epidemic algorithms often adopt simplistic mathematical models [12] that assume a homogenous topology (a fully connected topology) for the underlying network to simplify the analysis. However, recently it is shown that considering the actual topology of the network in the analysis extensively affects the results of the analysis [9]. We perform our analysis of Scoop assuming an arbitrary random graph as the underlying topology of the peer-to-peer network and specifically derive final results for a power-law random graph, which is the observed topology for some peer-to-peer databases [26].

We originally described our preliminary ideas with Scoop in a poster paper [2]. This paper is merely a descriptive introduction to Scoop (equivalent to the introduction section of the current paper). The current paper extends the poster paper by including the Scoop algorithm, analytical details, and experimental results.

1.3.4 Experimental Results

We performed an empirical study via simulation to evaluate the efficiency of Scoop. In lack of other existing solutions, we compared Scoop with hypothetical partial read operations that use scope-limited flooding (i.e., flooding with limited TTL) and k-random-walkers (with various k) for query dissemination. As we explained in Section 1.2, these dissemination mechanisms are not originally appropriate for execution of the partial read requests and we had to artificially inform them about the coverage required to satisfy each request. Our results show that even under such artificial conditions, Scoop still outperforms scope-limited flooding in communication cost while maintaining a reasonable communication time. Also, to our surprise, Scoop not only has a much better communication time as compared to that of the random-walk but also outperforms a 32-random-walker (the optimal case as suggested in [19]) even in communication cost.

1.3.5 Summary of Contributions

To summarize, we enlist our contributions with Scoop as follows:

- Identifying and formulating a basic query type, i.e., β-query, as an essential type of sub-query for answering generic approximate queries in peer-to-peer databases.
- Developing an epidemic-based query answering mechanism to evaluate β-queries in unstructured peer-to-peer databases with arbitrary random graph topology. The novelty with this query answering mechanism is in introducing a generic and rigorous analytic approach based on percolation analysis to control the epidemic dissemination for both correct and efficient query answering.

- Evaluation of the proposed query answering mechanism by extensive empirical analysis via simulation.

It is important to note that although Scoop is applicable to the more generic distributed databases, it is particularly designed to accommodate the scalability and dynamism requirements of the peer-to-peer databases (as discussed above).

1.4 Roadmap

The remainder of this paper is organized as follows. In Section 2, we define the partial read problem with further details. Section 3 introduces the generic case of Scoop for partial read from peer-to-peer databases with arbitrary random graph topology. In Section 4, we specialize our results from Section 3 to develop a specific case of Scoop for the real-world peer-to-peer databases with power-law topology. In Section 5, we briefly discuss several variants of Scoop. Section 6 presents the results of our empirical study on Scoop. Finally, Section 7 concludes the paper and discusses the future directions of this research.

2 Partial Read Operation

Given $\beta \in [0,1]$, a partial read operation must retrieve a fraction β of the relation R, which is horizontally fragmented and distributed among the nodes of the network N. We term such a partial read request a β-query. It is important to note that the semantics of the β-query is less restrictive as compared to that of the stronger *top-k* query. While both these types of queries are useful and popular in peer-to-peer databases, they require distinct query answering mechanisms. To answer a β-query, beginning from the originator of the β-query (which is the same node where the user query is issued), query is disseminated through the network to visit a fraction

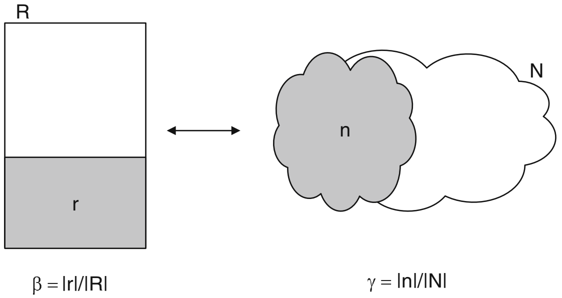


Fig. 1 Partial read

γ of the nodes of the network which collectively store a sub-relation r such that $|r|/|R| = \beta$ (see Fig. 1).

Regardless of the choice of the dissemination mechanism to implement the partial read operation, the main challenge is tuning the dissemination mechanism per β-query such that the query is correctly satisfied. Figure 2 depicts the parameter mapping process for tuning a generic dissemination mechanism. At Step I, considering the distribution of the tuples among the nodes, γ (i.e., the fractional size of the subset n of the nodes that should be visited to retrieve r) is calculated based on β. Subsequently, at Step II some parameter(s) of the dissemination mechanism (such as parameter p for the epidemic dissemination) is tuned based on γ such that the dissemination actually covers n nodes of the network. Specifics of Step II of the mapping process depend on the particular dissemination mechanism applied to implement the partial read.

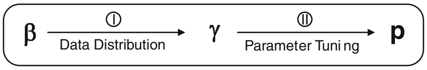


Fig. 2 Parameter mapping process for tuning a generic dissemination mechanism

With Scoop, we employ an epidemic dissemination mechanism to implement the partial read operation. We assume the tuples of the relation R are uniformly distributed among the nodes of the network (hence, the mapping at Step I will be trivial $\gamma = \beta$), and focus on the specifics of Step II for epidemic dissemination to derive p as a function of γ (or equivalently, β). Before describing Scoop and its tuning process, here we provide some definitions.

2.1 Definitions

2.1.1 Communication Graph

Communication graphs are used to represent and visualize query dissemination over networks. A network with the node-set N and link/edge-set E can be modelled as an undirected graph $G(N,E)$. For a query dissemination initiated at time $t = t_0$ on G, the *communication graph* at any time $t \geq t_0$ is a subgraph $G_t(N_t,E_t)$ of G, where $E_t \subseteq E$ is the set of links traversed by at least one query replica during the time interval $[t_0,t]$, and $N_t \subseteq N$ is the set of nodes visited by at least one query replica during the same time interval. Associated with any link e of G_t is a weight w_e that is the number of times the link e is traversed during the time interval $[t_0,t]$. We assume a discrete time model; thus, the dynamic process of disseminating a query is completely represented by the set of communication graphs $\{G_{t_0}, G_{t_0+1}, G_{t_0+2}, ..., G_{t_0+T}\}$, where at time $t = t_0 + T$ the query dissemination is terminated (hence, for all $t \geq (t_0 + T)$, $G_{t_0+t} =$

G_{t_0+T}). For example, Fig. 3 depicts the 6 first communication graphs of a query that is initiated at node A and disseminated based on the random walk dissemination mechanism.

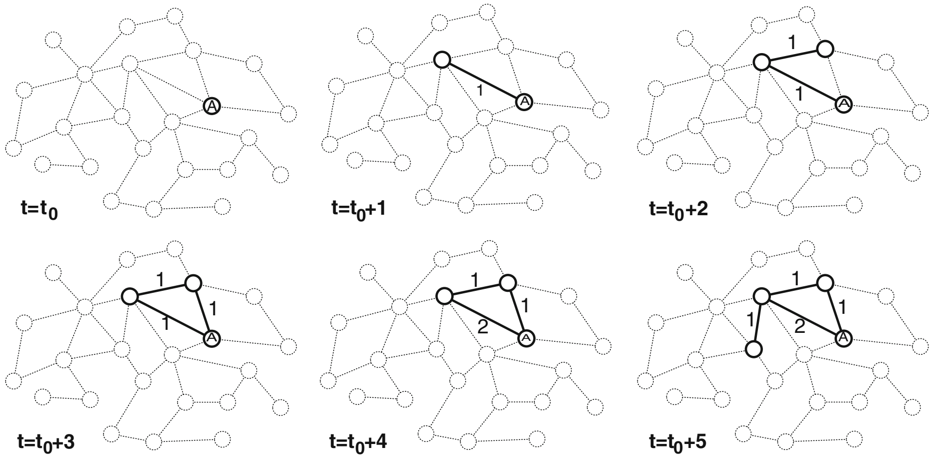


Fig. 3 Communication graph

2.1.2 Efficiency Measures for Sampling

We define two metrics to measure the efficiency of query dissemination mechanisms used for sampling from networks:

1. *Query cost (or sampling cost, or communication cost)* **C**: Assuming uniform cost for traversing the links of the network and uniform query size, we model the communication cost of disseminating a query based on a particular dissemination mechanism as the total number of query replicas communicated between the nodes of the network during the entire process of query dissemination. In communication-graph terminology:

$$\mathbf{C} = \sum_{e \in E_{t_0+T}} w_e$$

2. *Query time (or sampling time, or communication time)* **T**: Assuming uniform latency for the network links, the sampling time is the total time it takes to disseminated the query. In communication-graph terminology:

$$\mathbf{T} = T$$

3 Scoop: Partial Read by Epidemic Dissemination

Epidemic dissemination is inspired by epidemic spreading of contagious diseases in social networks (i.e., networks that model a society, with nodes as people and links as social links between people who are in contact). A contagious disease first appears at one node (the originator of the disease), and then through the social links disseminates throughout the network in a flood-like fashion, from the infected person to its direct neighbors in the social network, from the infected neighbors to their neighbors, and so on. The success in transmission of the disease through a social link is probabilistic, i.e., the disease is transmitted from an infected node to its susceptible neighbor with probability p ($0 \leq p \leq 1$) and is ceased with probability $1 - p$. The value of p depends on the infectiousness of the particular disease as well as some other environmental parameters, and with simple disease spreading models the value is generic to all links of the network. When the spreading terminates, the disease has covered/reached a sample h of the total node population H ($h \subseteq H$), where the relative size of h increases with increasing value of p (although not necessarily linearly).

With epidemic dissemination by Scoop, we model the query dissemination mechanism on the disease spreading process. By analogy, we take the dissemination of a query in a peer-to-peer database as the spreading of a disease in a social network. With this analogy, the infection probability p translates to the query forwarding probability, and the infected sample h maps to the sampled node-set n. Among various disease spreading models, we model epidemic dissemination on the *SIR (Susceptible-Infected-Removed)* disease spreading model. Below, first we describe the SIR disease spreading model, which readily defines our epidemic query dissemination mechanism with Scoop. Thereafter, for our SIR-based epidemic query dissemination mechanism we develop a percolation model to derive γ as a function of the query forwarding probability p for peer-to-peer databases with arbitrary random graph topology. The function $\gamma(p)$ is a one-to-one function. Therefore, in turn, it defines p as a function of γ, which enables Step II of the tuning process for Scoop.

3.1 SIR Epidemic Dissemination

With the SIR disease spreading model, at any particular time during dissemination of the disease (or equivalently, the query), each node of the network is in one of the three states susceptible (S), infected (I), or removed (R). A "susceptible" node is liable to infection but is not infected yet; an "infected" node is currently infected and is able to infect its neighbors; and a "removed" node has recovered from the infection and is both immune to further infection and impotent to infect other nodes. The discrete-time and dynamic process of SIR epidemic dissemination can be explained as follows. Initially, at time t_0 all nodes of the network are susceptible except the originator of the disease, which is infected. As the disease propagates throughout the network, if at time $t \geq t_0$ a susceptible node A has an infected neighbor B, at time $t+1$ with probability p node A conceives the disease from B and becomes infected (see Fig. 4 for the state transition diagram of a node). An infected node remains in the infectious state for a period of time τ (during which it is able to infect its neighbors), and then finally becomes removed. We assume $\tau = 1$ without loss of generality. The disease dissemination terminates (dies out) at time $t_0 + T$ (where $T \geq 1$) when all nodes of the network are either in the removed state (affected by the disease) or the susceptible state (survived the disease), and none of the nodes are in the infected state. By analogy, with the SIR-based epidemic query dissemination of Scoop, when the query dissemination terminates the set of the removed nodes is the set n of nodes visited by the query, and the set of susceptible nodes is the set $N \backslash n$ of the nodes not covered by the query dissemination (see Fig. 1).

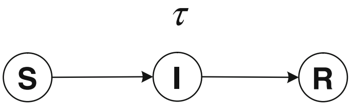


Fig. 4 State diagram for SIR epidemic dissemination

3.2 Percolation Model

To tune the epidemic dissemination with Scoop, we need to answer two questions:

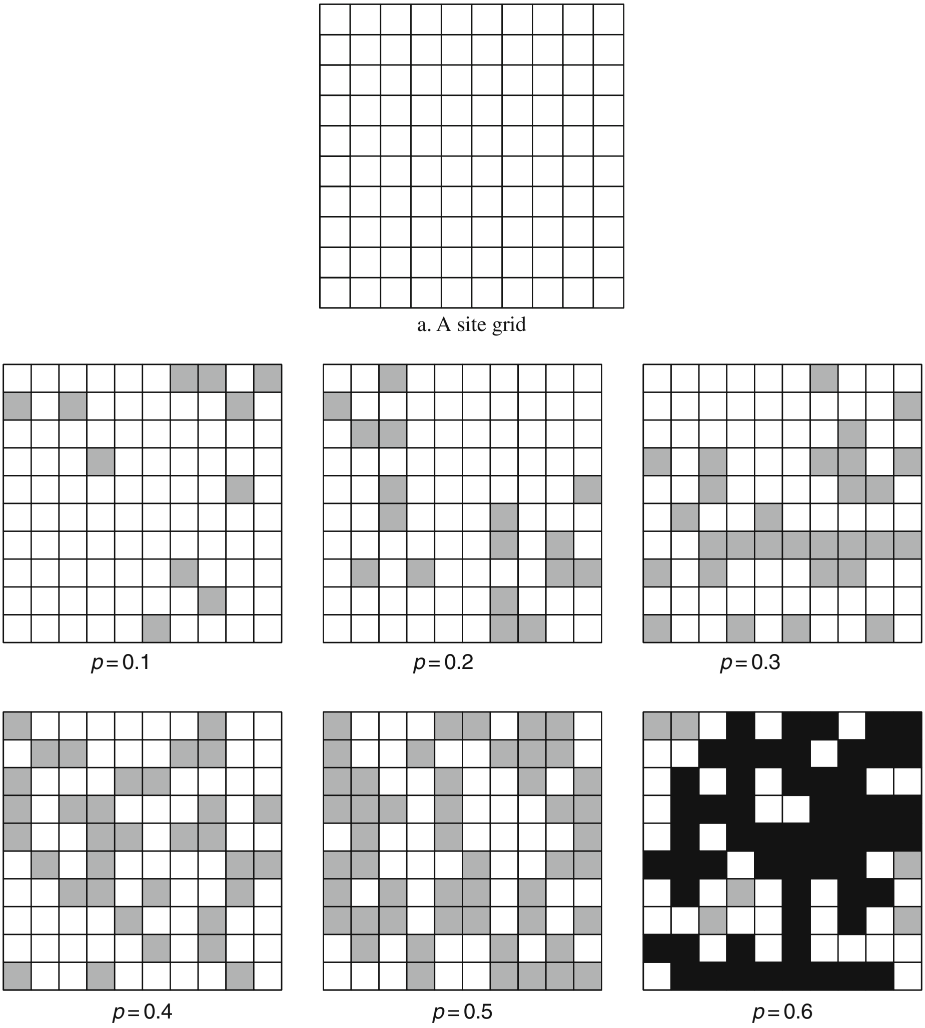
1. *How large p should be for the query dissemination to prevail a large network?* For a query to prevail a network N, we should have:

$$\lim_{|N| \to \infty} \frac{|n|}{|N|} = \gamma \tag{1}$$

with $\gamma > 0$. In other words, the size of the covered node-set n must be comparable to the size of the entire network N, otherwise the partial read operation cannot satisfy any β-queries other than the trivial β-query with $\beta = 0$. With too small values of p, the dissemination dies out quickly, and the query fails to prevail and covers an infinitesimally small number of nodes as compared to the total number of nodes $|N|$ in large networks; i.e., we have:

$$\lim_{|N| \to \infty} \frac{|n|}{|N|} = 0$$

2. *How can we derive γ as a function of p?* γ is an increasing function of p. Having $\gamma(p)$, we can fulfill Step II of the tuning process (see Fig. 2) by deriving $p(\gamma)$ as $\gamma^{-1}(p)$. In other words, given a β-query and assuming $\gamma = \beta$, we can tune the forwarding probability p of Scoop on-the-fly to satisfy the β-query.



a. A site grid

$p=0.1$ $p=0.2$ $p=0.3$

$p=0.4$ $p=0.5$ $p=0.6$

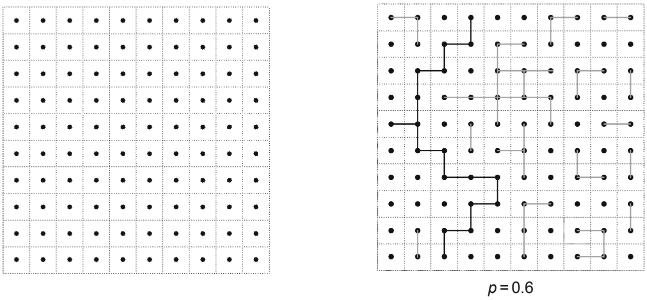
b. Site precolation instances

Fig. 5 Site percolation problem

To answer these questions, we model the epidemic dissemination process as a percolation problem. First, we illustrate this percolation problem by describing two toy percolation problems (Figs. 5 and 6). Consider the grid in Fig. 5a. Each cell of the grid is called a site. Suppose we color each site of the grid independently with probability p. Figure 5b depicts several instances of the colored grid with various probabilities. As p increases, larger clusters of colored sites appear. Particularly, at $p = 0.6$ there is a giant cluster (marked in black) that spans from the top side of the grid to the bottom side. When such a giant cluster appears in a colored grid, we say the grid percolates. It is proved [21, 22, 29] that there exists a critical probability p_c

(in this case, $p_c \approx 0.59$) such that whp[4] the grid percolates only if $p \geq p_c$. The size of the giant cluster depends on the value of p and as p increases the giant cluster grows such that at $p = 1$ the giant cluster covers the entire grid. The toy problem described above is called a site percolation problem on two dimensional grid.

Equivalently, one can define the dual bond percolation problem where a set of nodes are arranged into a grid (see Fig. 6a) with a bond (not shown) between every pair of adjacent nodes. Suppose each bond is colored independently with probability p. In this case a giant cluster is a cluster of nodes connected with colored bonds that percolates from one side to the other side of the grid (in Fig. 6b the giant cluster is marked in black).



a. A bond grid

b. Bond percolation instance

Fig. 6 Bond percolation problem

We model the epidemic dissemination process as a bond percolation problem on an arbitrary random graph. Our problem differs from the toy bond percolation problem described above in two ways. First, instead of a grid we assume an arbitrary random graph as the underlying bonding structure among the nodes. With an arbitrary random graph, each node is not restricted to a fixed number of neighbors (e.g., four neighbors as in two dimensional grid) but can have a random number of neighbors according to some arbitrary degree distribution. On the other hand, unlike a grid, a random graph is not delimited by sides. For a bond percolation problem on a random graph, a giant cluster is defined as a cluster of nodes (i.e., a set of nodes connected with colored bonds) where the size of the cluster is comparable to the size of the entire graph.

This bond percolation problem models the epidemic dissemination as follows: the underlying random graph represents the physical topology of the peer-to-peer network (i.e., the logical overlay); and a cluster generated with the coloring probability p, is an instance of the communication graph of a query if the query was initiated by a node within the cluster and disseminated with the forwarding probability p. With this model, we can answer the two questions raised before as follows:

[4] "whp" stands for "With High Probability".

1. *How large p should be for the query dissemination to prevail a large network?* The query prevails the network if and only if a giant cluster exists. Thus, whp query prevails the network if and only if $p \geq p_c$, where p_c is the critical percolation probability of the bond percolation problem.
2. *How can we derive γ as a function of p?* To derive $\gamma(p)$ we should derive the relative size of the giant cluster as a function of the coloring probability p, for all $p \geq p_c$.

Next, we solve the bond percolation problem for the critical probability p_c, and the size of the giant cluster as a function of p.

3.3 Tuning Scoop

3.3.1 Definitions

We use the generating-function formalism [32] to represent probability distribution functions. Particularly, the generating function $G_0(x)$ for the distribution of the node-degree k in an arbitrary random graph is defined as:

$$G_0(x) = \sum_{k=0}^{\infty} p_k x^k \tag{2}$$

where p_k is the probability that a randomly chosen node of the graph has degree k. From Equation (2), one can derive the n-th moment of the node degree distribution as follows:

$$\langle k^n \rangle = \left[(x\frac{d}{dx})^n G_0(x) \right]_{x=1} \tag{3}$$

Also, for a random graph represented by the generating function $G_0(x)$, we define $G_1(x)$, which is the generating function for the distribution of the degree of the nodes we arrive at by following a randomly chosen link from the graph. $G_1(x)$ depends on $G_0(x)$ and is derived as follows:

$$G_1(x) = \frac{1}{\langle k \rangle} G_0'(x) \tag{4}$$

3.3.2 Analysis

First, consider the bond percolation model described in Section 3.2. Suppose the coloring probability is p. A cluster of nodes connected by the colored bonds is itself a random graph embedded within the underlying random graph $G_0(x)$. It is easy to derive the generating function $G_0(x;p)$ for the degree distribution of the graphs representing the clusters based on $G_0(x)$:

$$G_0(x;p) = \sum_{m=0}^{\infty}\sum_{k=m}^{\infty} p_k \binom{k}{m} p^m (1-p)^{k-m} x^m \tag{5}$$

$$= \sum_{k=0}^{\infty} p_k \sum_{m=0}^{k} \binom{k}{m} (xp)^m (1-p)^{k-m} \tag{6}$$

$$= \sum_{k=0}^{\infty} p_k (1-p+xp)^k \tag{7}$$

$$= G_0(1+(x-1)p) \tag{8}$$

Similarly one can derive $G_1(x;p)$ as follows:

$$G_1(x;p) = G_1(1+(x-1)p) \tag{9}$$

Next, we derive the distribution of the size s (i.e., the number of nodes) of the clusters. Assume $H_0(x;p)$ is the generating function for the distribution of the size of the clusters. Observing that each cluster consists of a node connected to k other sub-clusters (where k is distributed according to $G_0(x;p)$), we derive the distribution of the cluster size by a recursive argument as follows:

$$H_0(x;p) = xG_0(H_1(x;p);p) \tag{10}$$

and similarly:

$$H_1(x;p) = xG_1(H_1(x;p);p) \tag{11}$$

From $H_0(x;p)$, we can also compute the average size $\langle s\rangle$ of the clusters using Equation (3):

$$\langle s\rangle = H_0'(1;p) = 1 + G_0'(1;p)H_1'(1;p) \tag{12}$$

However, according to Equation (11) we have:

$$H_1'(1;p) = 1 + G_1'(1;p)H_1'(1;p) = \frac{1}{1-G_1'(1;p)} \tag{13}$$

Thus:

$$\langle s\rangle = 1 + \frac{G_0'(1;p))}{1-G_1'(1;p)} = 1 + \frac{pG_0'(1)}{1-pG_1'(1)} \tag{14}$$

Now, since at the critical probability p_c the giant cluster appears, the average size of the clusters $\langle s\rangle$ goes to infinity at $p = p_c$; i.e.:

$$\langle s\rangle = 1 + \frac{p_cG_0'(1)}{1-p_cG_1'(1)} \to \infty \tag{15}$$

Therefore, the critical probability p_c can be computed as:

$$p_c = \frac{1}{G_1'(1)} = \frac{1}{\frac{\langle k^2\rangle}{\langle k\rangle} - 1} \tag{16}$$

This concludes the solution for the first question raised in Section 3.2. To answer the second question, i.e., calculating the relative size of the giant cluster as a function of p, we observe that for $p \geq p_c$, $H_0(x;p)$ remains the distribution of the *finite* size clusters, i.e., all clusters except the giant cluster. Thus, we have:

$$H_0(1;p) = 1 - \gamma(p) \tag{17}$$

Using Equation (10) we can derive $\gamma(p)$ as follows:

$$\gamma(p) = 1 - G_0(y;p) \tag{18}$$

where according to Equation (11), $y = H_1(1;p)$ is the solution of:

$$y = G_1(y;p) \tag{19}$$

We solve these equations for $\gamma(p)$ numerically by iteration.

4 A Real-World Example of Scoop

In Section 3, we described the generic case of Scoop for the peer-to-peer databases with arbitrary random graph topology. As captured by empirical studies, some of the real-world peer-to-peer databases such as Gnutella [18] and Kazaa [28] have power-law random graph topologies [11, 14, 25, 26]. Here we specialize the generic case of Scoop for the peer-to-peer databases with power-law topology.

4.1 Network Topology

In this section, we assume the topology of the peer-to-peer database is a power-law (or scale-free) random graph, i.e., a random graph [4] with power-law probability distribution for node degrees. Intuitively, in a power-law random graph most of the nodes are of low degree while there are still a few nodes with very high connectivity. We define the power-law probability distribution function for the node degree k as follows:

$$p_k = Ck^{-\eta}e^{-k/\nu} \tag{20}$$

where η, ν, and C are constants. η is the skew factor of the power-law distribution, often in the range $2 < \eta < 3.75$ for real networks. For example, a case study reports $\eta = 2.3$ for Gnutella [26]. The less the skew factor, the heavier the tail of the power-law distribution, which translates to larger number of highly connected nodes. A pure power-law distribution does not include the exponential cut-off factor ($e^{-k/\nu}$), allowing for nodes with infinite degree, which is unrealistic for real peer-to-peer databases. The cut-off factor with index ν shortens the heavy tail of the power-law distribution such that the maximum node degree for the nodes of the graph is in the same order of magnitude as ν. Finally, C is the normalization factor that is computed as $C = [\mathbf{Li}_{\eta}(e^{-1/\nu})]^{-1}$, where $\mathbf{Li}_{\eta}(x) = \sum_{k=1}^{\infty} \frac{x^k}{k^{\eta}}$ is the η-th polylogarithm function of x.

4.2 Analysis

The generating function of the power-law degree distribution (see Equation 20) can be represented based on the polylogarithm function as follows:

$$G_0(x) = \frac{\mathbf{Li}_{\eta}(xe^{-1/\nu})}{\mathbf{Li}_{\eta}(e^{-1/\nu})} \tag{21}$$

From Equation (3), we can compute the first and the second moments of the power-law degree distribution[5]:

$$\begin{aligned} \langle k \rangle &= (x\frac{d}{dx})G_0(x)\Big|_{x=1} = \frac{\mathbf{Li}_{\eta-1}(e^{-1/\nu})}{\mathbf{Li}_{\eta}(e^{-1/\nu})} \\ \langle k^2 \rangle &= (x\frac{d}{dx})^2 G_0(x)\Big|_{x=1} = \frac{\mathbf{Li}_{\eta-2}(e^{-1/\nu})}{\mathbf{Li}_{\eta}(e^{-1/\nu})} \end{aligned}$$

Consequently, from Equation (16) we can derive the critical probability p_c for a power-law graph as follows:

$$p_c = \frac{\mathbf{Li}_{\eta-1}(e^{-1/\nu})}{\mathbf{Li}_{\eta-2}(e^{-1/\nu}) - \mathbf{Li}_{\eta-1}(e^{-1/\nu})} \tag{22}$$

In Fig. 7, we illustrate p_c as a function of ν for various η values in a real-world power-law peer-to-peer database, i.e., Gnutella. For Gnutella, the skew factor η is estimated as low as $\eta = 1.4$ and as high as $\eta = 2.3$, in different scenarios. Also, ν is in the range of 100 to 1000. As illustrated by this example, the critical probability p_c in power-law networks can be as low as 0.01.

We also solved Equation (18) to derive $\gamma(p)$ for power-law networks by numerical iteration. In Fig. 8a, we show our result for a power-law random graph with the skew factor $\eta = 2.3$.

4.3 Algorithm

1. Seeding the Query (Phase I)
   To make sure the query dissemination is initiated by a node belonging to the giant cluster, the actual query originator initiates a selective walker to locate a

[5] Note: $\frac{d}{dx}\mathbf{Li}_{\eta}(x) = \frac{1}{x}\mathbf{Li}_{\eta-1}(x)$

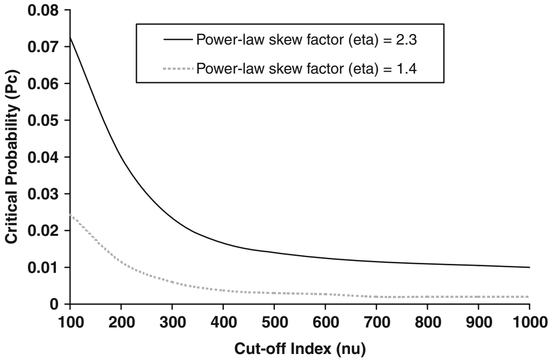


Fig. 7 Critical probability in power-law peer-to-peer databases

highly connected node in the network. With the selective walk, at each hop the query is forwarded to a neighbor with the maximum connectivity degree until the walker reaches a node with a connectivity degree higher than the degree of all its neighbors. In [1], it is shown that in a power-law network whp such a selective walker finds a highly connected node belonging to the giant cluster in $O(\log N)$ hops, where N is the size of the network. Our experiments also verify this result.

2. Disseminating the Query (Phase II)
   Next, the SIR-based epidemic query dissemination is initiated at the highly connected node located by the selective walker. The query is disseminated with a forwarding probability $p > p_c$ selected according to Equation (18), such that $\gamma(p)$ satisfies the given β-query.

5 Variants of Scoop

Since in peer-to-peer databases some nodes may refrain from participating in query dissemination, here we introduce a family of variants for the basic case of Scoop to model this behavior. With this family, unlike the original SIR model where initially all nodes are in the "susceptible" state, some nodes begin in the "removed" state. In the context of the disease epidemic, these nodes represent the people who are vaccinated. We term such nodes the *blocked* nodes.

We consider three different variants with blocking for the basic Scoop, each representing a particular real-world scenario:

1. *Scoop with uniform blocking*: In this case, nodes are blocked with uniform probability. This case models the networks where nodes autonomously decide whether or not to participate in the query dissemination.
2. *Scoop with negative degree-correlated blocking*: In this case, the nodes with lower connectivity degrees are blocked with higher probability. For example, this case models the peer-to-peer file-sharing networks where low degree nodes

are usually also low-bandwidth and therefore, to avoid congestion and possible isolation, may refrain from participating in query dissemination with higher probability.

3. *Scoop with positive degree-correlated blocking*: This case is opposite to the previous case, where nodes with higher connectivity degrees are more probably blocked. This case models the scenario where, for example, high degree nodes of a power-law network are attacked and disabled.

6 Experiments

We conducted two sets of experiments via simulation to (1) study the behavior of Scoop empirically, and (2) evaluate the efficiency of Scoop. For this purpose, we implemented a discrete-time event-driven simulator in C++. We used an Enterprise E220 SUN server to perform the experiments.

6.1 Methodology

With the first set of experiments, we studied the relation between the forwarding probability p and the size of the sample node-set covered by the Scoop query dissemination. Therefore, with these experiments data content of the nodes is irrelevant. With the second set of experiments, we evaluated the efficiency of various partial read operations in resolving β-queries. Our Monte Carlo simulation was organized as a set of "runs". For the first set of experiments each run consists of (1) selecting a network topology, (2) selecting a query originator, and finally (3) initiating 50 query disseminations per forwarding probability p (for each one of the partial read operations) while varying p from 0 to 1, and recording the average size of the covered node-set as well as **C** and **T**. For the second set of experiments a run comprises (1) selecting a network topology, (2) selecting an object-set (a multiset of tuples), (3) distributing the object-set among the network nodes, (4) selecting a query originator and finally (5) initiating the query for 50 times per β (for each of the partial read operations) while varying β from 0 to 1, and recording the average value of their efficiency numbers **C** and **T**. With this set of experiments, all issued queries are answered correctly by Scoop and we focus on reporting the efficiency of the query answering with Scoop. Each result data-point reported in Section 6.2 is the average result of 50 runs. The coefficient of variance across the runs was below 2.5% and hence show the stability of the result. The high stability of the performance is an expected benefit of the randomization inherent in Scoop.

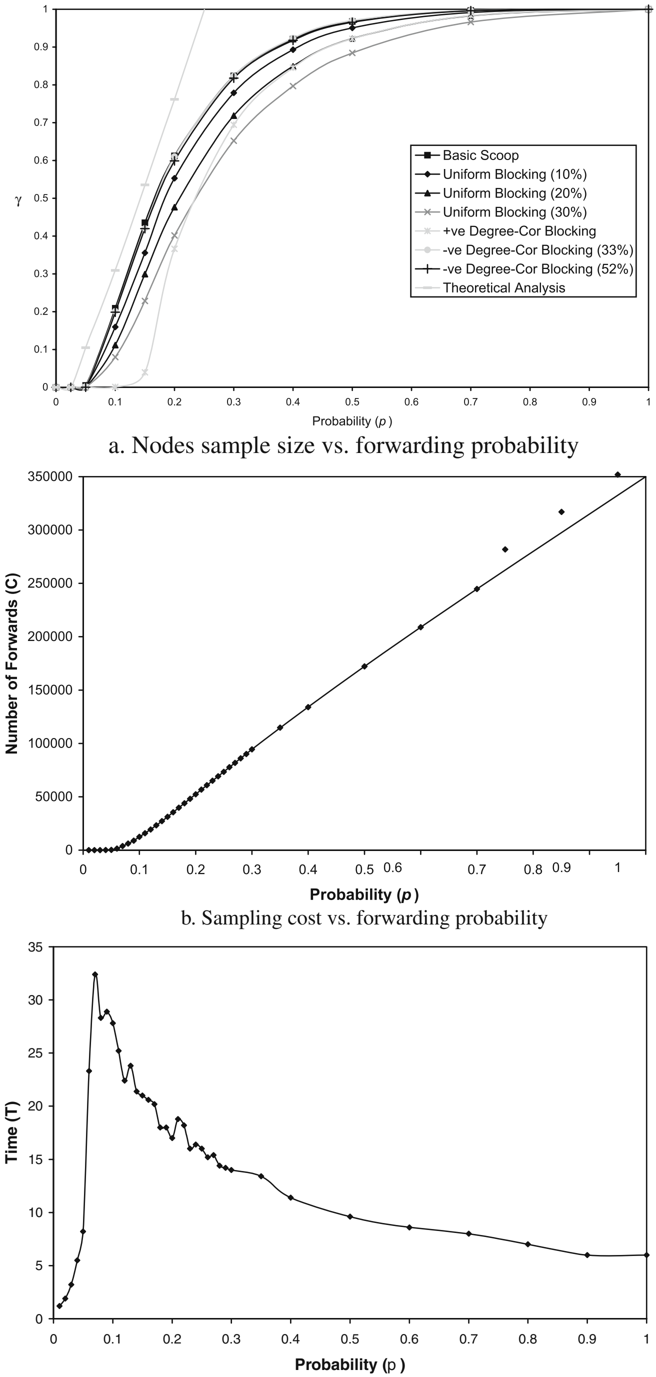
We generated a set of 100 undirected power-law graphs $G(N,E)$ each with $|N| = 50000$ and $|E| \approx 200000$. The skew factors of the graphs are all about $\eta = 2.3$ as measured in [26]. The minimum number of edges per node is 4, and the cut-off index of the graphs is $v = 100$. The graphs are generated using the preferential attachment model proposed by Barabasi et al. [3].

We considered a 5-dimensional content space and generated 100 object-sets. For each object-set, we first generated $|U| = 100000$ objects $u = \langle a_1, a_2, ..., a_5 \rangle$, where a_i is an integer value uniformly selected from the interval $[1, 10]$. Thereafter, we replicated the objects according to the object replication scheme defined in Section 4.1 with the total number of objects $|R| = 500000$. R is uniformly distributed among the set of network nodes N.

6.2 Results

Figure 8 illustrates the results of our first set of experiments. Figure 8a depicts the relation between the query forwarding probability p and the relative size γ of the covered node-set n. First, we notice how close the results of our theoretical analysis conforms with the performance of the basic case of Scoop in practice, specially for our p values of interest close to the critical forwarding probability p_c. Also, we observe that while performance of the Scoop with negative degree-correlated blocking is almost identical to that of the basic Scoop (they overlap in the figure), with positive degree-correlated blocking, the coverage for the same forwarding probability decreases significantly. This shows (1) the importance of the highly connected nodes in the performance of Scoop, and (2) the independence of its performance from the nodes with lower connectivity degrees, which are often low-bandwidth and volatile. Figure 8b confirms our previous conjecture that the sampling cost of Scoop is linearly proportional to the query forwarding probability. Also, (from Fig. 8a, b) notice that with $p = 0.3$ almost 80% of the network nodes can be covered with only about 25% of the sampling cost of the regular flooding (with $p = 1$). Besides, the size of the covered node-set becomes sublinearly proportional to the network size starting at $p_c \approx 0.05$, where the sampling cost is almost two orders of magnitude less than that of the flooding. Finally, Fig. 8c illustrates how the sampling time reduces as the forwarding probability goes from p_c towards 1, because the giant cluster becomes more strongly connected. Also, we observe that in the worst case, the sampling time with Scoop only increases by a factor of 4 over the minimum possible sampling time with flooding.

Figures 9 and 10 illustrate the results of our second set of experiments. With these experiments, we compared the efficiency of Scoop in answering β-queries with that of partial read operations based on random walk and scope-limited-flooding dissemination mechanisms. As we mentioned in Section 1.2, unlike Scoop, these two partial read operations are unable to determine whether they have covered a sufficiently large fraction of the network to satisfy a particular β-query. Nevertheless, to be able to compare Scoop with these partial read operations, for each particular β-query with given β we calculated the absolute (not relative) number of nodes that must be covered to satisfy the query, and terminated the random walk and flooding

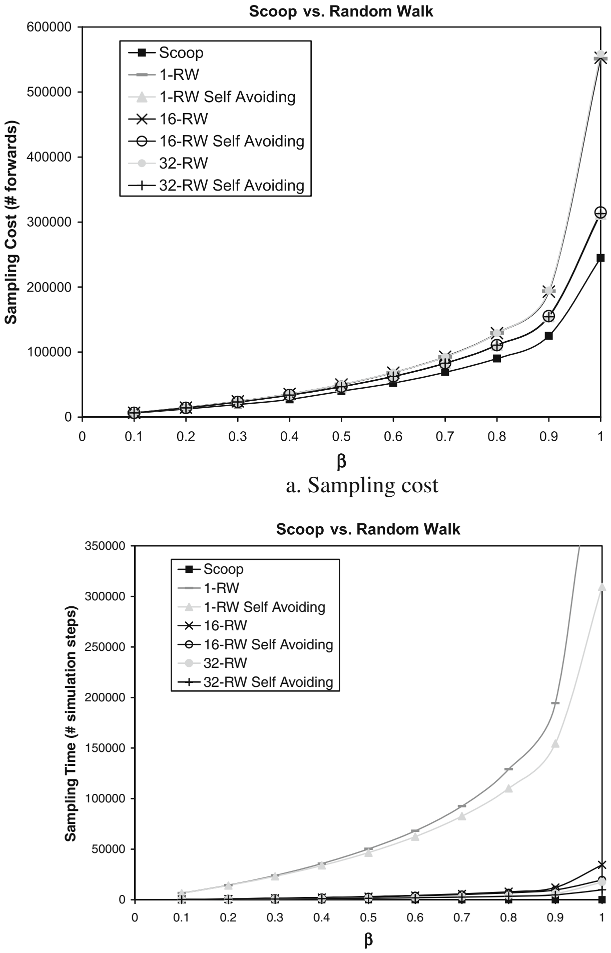


a. Nodes sample size vs. forwarding probability

b. Sampling cost vs. forwarding probability

c. Sampling time vs. forwarding probability

Fig. 8 Verification of the analytical results



a. Sampling cost

b. Sampling time

Fig. 9 Scoop vs. random walk

as soon as their coverage exceed the required number of nodes. Scoop can decide on the required coverage on its own.

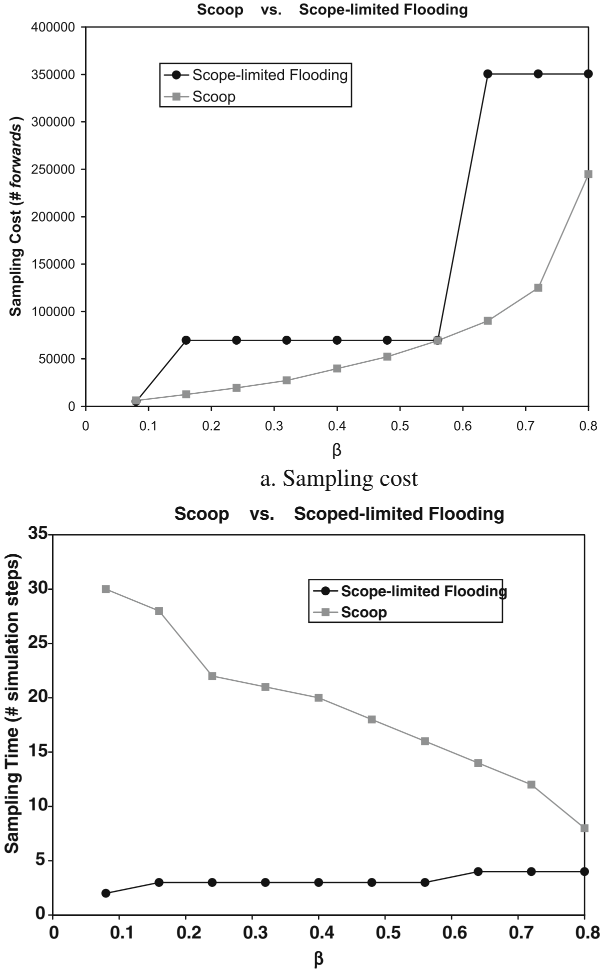
Figure 9b shows that, as one can expect, the sampling time of Scoop is always incomparably shorter than that of the random walk, even with 32 parallel walkers (in the figure, the Scoop plot lies on the *x* axis). However, to our surprise, Scoop also outperforms random walk in sampling cost (see Fig. 9a). In other words, to cover the same number of nodes the SIR-based dissemination uses a "lighter" communication graph with less query forwarding (see Section 2.1.1 for the definition of the communication graph) as compared to that of the random walk. This can be justified by considering the fact that multiple random walkers are traversing the topology with no (even implicit) inter-communication to avoid redundant traversal. However, with the SIR-based dissemination adopted by Scoop the more a part of the topology is explored the less the chance of it to be re-explored; hence, redundant traversal is reduced. Also note that as illustrated in Fig. 9a, the random walk algorithms with different number of walkers incur the same sampling cost. This should not be surprising; more walkers enhance the sampling time of the random walk by scanning the network in parallel, but a single random walker walks as much as 32 random walkers walk in aggregate to cover the same number of nodes. Finally, notice that among random walk algorithms, self-avoiding random walk (with which each random walker avoids its repeated paths) always outperforms regular random walk in sampling time.

Figure 10a shows that Scoop always outperforms scope-limited flooding in sampling cost. Notice the step-like diagram for the scope-limited flooding. Since at each hop flooding scans an exponentially larger number of new nodes, unlike Scoop it cannot be tuned properly to cover a certain fraction of the network nodes with fine granularity. Finally, Fig. 10b shows that although the sampling time of Scoop always exceeds that of the flooding (which is optimal), even in the worst case it remains tolerable.

7 Conclusion and Future Work

In this paper, we proposed Scoop to answer β-queries. Scoop is easy to implement because each node of the peer-to-peer database can simply toss a biased coin with probability p to decide whether the query should be disseminated through a link or not. We showed through percolation analysis that the value of p can be rigorously computed for each β-query such that the query is correctly satisfied. Also, with a comparative empirical study we showed that Scoop outperforms flooding by up to two orders of magnitude in communication cost while maintaining a tolerable response-time. Also, as compared to a 32-random-walker, Scoop has not only faster response time but also less communication cost.

We intend to extend this study in three directions. First, since Scoop is essentially a flood-based dissemination technique with relatively short communication time (see Fig. 8c), considering the typical rate of churn in peer-to-peer databases we do not expect significant changes happening during a Scoop dissemination. Hence, we do not anticipate the performance of Scoop is noticeably affected by the typical rate of churn. However, for the peer-to-peer databases that may intrinsically have higher churn such that the time-scale of the dynamic changes in the peer-to-peer database is comparable to that of the query dissemination, we are planning to use *dynamic* percolation analysis to factor in the effect of the changes during the query dissemination. Second, we intend to explore the more specific characteristics of the topologies of the peer-to-peer databases such as degree correlations, sparsity, and



a. Sampling cost

b. Sampling time

Fig. 10 Scoop vs. scope-limited flooding

expansion/conductance, and consider these characteristics (in addition to the degree distribution) in our percolation-based analysis of the query dissemination. Third, we plan to extend our proposed family of data retrieval operations to include operations for answering *continuous* data retrieval requests. For this purpose, we will adopt the SIS (Susceptible-Infected-Susceptible) disease dissemination model. Unlike SIR, which models epidemic diseases that disseminate once throughout the social network and quickly disappear, SIS models *endemic* diseases that become resident in the social network and continuously disseminate.

Acknowledgments

This research has been funded in part by NSF grants IIS-0238560 (PECASE), IIS-0534761, IIS-0742811 and CNS-0831505 (CyberTrust), and in part from the METRANS Transportation Center, under grants from USDOT and Caltrans. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Adamic, L., Lukose, R., Puniyani, A., Huberman, B.: Search in power-law networks. Physics Review Letters **64**(46135) (2001)
2. Banaei-Kashani, F., Shahabi, C.: Partial selection query in peer-to-peer databases (poster paper). In: Proceedings of the 22nd International Conference on Data Engineering (ICDE) (2006)
3. Barabasi, A., Albert, R.: Emergence of scaling in random networks. Science **286**, 509–512 (1999)
4. Bollobas, B.: Random Graphs. Academic Press, New York (1985)
5. Boyd, S., Ghosh, A., Prabhakar, B., Shah, D.: Gossip algorithms: Design, analysis and applications. In: Proceedings of the Conference on Computer Communications (INFOCOM) (2005)
6. Crespo, A., Garcia-Molina, H.: Routing indices for peer-to-peer systems. In: Proceedings of the 22nd International Conference on Distributed Computing Systems(ICDCS) (2002)
7. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J.: Epidemic algorithms for replicated database maintenance. In: Proceedings of the sixth annual ACM Symposium on Principles of Distributed Computing (PODC) (1987)
8. Ganesan, D., Krishnamachari, B., Woo, A., Culler, D., Estrin, D., Wicker, S.B.: An empirical study of epidemic algorithms in large scale multihop wireless networks. Tech. Rep. CSD-TR 02-0013, UCLA (2002)
9. Ganesh, A., Massoulié, L., Towsley, D.: The effect of network topology on the spread of epidemics. In: Proceedings of the Conference on Computer Communications (INFOCOM) (2005)
10. Gkantsidis, C., Mihail, M., Saberi, A.: Hybrid search schemes for unstructured peer-to-peer networks. In: Proceedings of the Conference on Computer Communications (INFOCOM) (2005)
11. Gummadi, K., Dunn, R., Saroiu, S., Gribble, S., Levy, H., Zahorjan, J.: Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In: Proceedings of the Symposium on Operating Systems Principles (SOSP) (2003)
12. Hethcote, H.: The mathematics of infectious diseases. SIAM Review **42**(4), 599–653 (2000)
13. Hromkovic, J., Klasing, R., Monien, B., Peine, R.: Dissemination of information in interconnection networks (broadcasting and gossiping). Combinatorial Network Theory pp. 125–212 (1996)
14. Jovanovic, M.: Modeling large-scale peer-to-peer networks and a case study of gnutella. Master's thesis, University of Cincinnati (2001)
15. Karp, R., Schindelhauer, C., Shenker, S., Vocking, B.: Randomized rumor spreading. In: Proceedings of the Symposium on Foundations of Computer Science (FOCS) (2000)
16. Kempe, D., Dobra, A., Gehrke, J.: Gossip-based computation of aggregate information. In: Proceedings of the Symposium on Foundations of Computer Science (FOCS) (2003)

17. Li, L., Halpern, J., Haas, Z.: Gossip-based ad hoc routing. In: Proceedings of the Conference on Computer Communications (INFOCOM) (2002)
18. Limewire.com: Gnutella (2006). Http://www.limewire.com/
19. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: Proceedings of the 16th International Conference on supercomputing (ICS) (2002)
20. Lv, Q., Ratnasamy, S., Shenker, S.: Can heterogeneity make gnutella scalable? In: Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS) (2002)
21. Molloy, M., Reed, B.: A critical point for random grraphs with a given degree sequence. Random Structures and Algorithms **6**, 161–180 (1995)
22. Newman, M., Strogatz, S., Watts, D.: Random graphs with arbitrary degree distribution and their applications. Physical Review E **64**(026118) (2001)
23. Ozsoyoglu, G., Du, K., Guruswamy, S., Hou, W.: Processing real-time, non-aggregate queries with time-constraintsin case-db. In: Proceedings of the 8th International Conference on Data Engineering (ICDE) (1992)
24. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM) (2001)
25. Ripeanu, M.: Peer-to-peer architecture case study: Gnutella network. In: Proceedings of the First International Conference on Peer-to-Peer Computing (P2P) (2001)
26. Saroiu, S., Gummadi, P., Gribble, S.: A measurement study of peer-to-peer file sharing systems. In: Proceedings of Multimedia Computing and Networking (MMCN) (2002)
27. Sarshar, N., Boykin, P.O., Roychowdhury, V.: Percolation search in power law networks: Making unstructured peer-to-peer networks scalable. In: Fourth International Conference on Peer-to-Peer Computing (P2P) (2004)
28. SharmanNetworks: Kazaa (2006). Http://www.kazaa.com/
29. Stauffer, D., Aharony, A.: Introduction to Percolation Theory, second edn. Taylor and Francis (1992)
30. Stoica, I., Morris, R., Karger, D., Kaashoek, M., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM) (2001)
31. Vrbsky, S., Liu, J.: Approximate - a query processor that produces monotonically improving approximate answers. IEEE Transactions on Knowledge and Data Engineering (TKDE) **5**(6), 1056–1068 (1993)
32. Wilf, H.: GeneratingFunctionology, second edn. Academic Press (1994)
33. Yang, B., Garcia-Molina, H.: Designing a super-peer network. In: Proceedings of the 19th International Conference on Data Engineering (ICDE) (2003)

Part V
Incentive Mechanisms

Incentive Mechanisms for Cooperation in Peer-to-Peer Networks

Daniel A. G. Manzato and Nelson L. S. da Fonseca

Abstract Peer-to-peer networks have recently emerged as a solution for the well-known problems of scalability imposed by client-server architecture. In these networks, the roles of serving and being served are not always asymmetrical, that is to say, one selfsame participant may at times cooperate and at others enjoy the cooperation of others. Moreover, quite often the figure of central authorities or servers is lacking, which implies an autonomy for the entities to decide on the patterns of behavior they will display, with the power to act selfishly, lavishly, selflessly or quite simply fairly, that is to say, cooperating only when possible. The feasibility of peer-to-peer systems is strongly based on the premise of cooperative behaviors. Moreover, the concept of synergy is fundamental to such systems, that is, it is expected to exploit the local resources of the participants so that their joint utilization results in a better result than the sum of the respective individual utilization. Thus, leaving the decision to cooperate or not under the sole responsibility of the participants, without any outside influence or incentive, entails putting at risk not only the system's efficiency, but also its very feasibility. This chapter sets out a study of incentive mechanisms for cooperation. The chapter place greater emphasis on the different incentive patterns used in common by the various incentive schemes in peer-to-peer networks.

Daniel A. G. Manzato
University of Campinas, Institute of Computing,
Av. Albert Einstein, 1251, P.O. Box 6176
13084-971, Campinas, SP, Brazil, e-mail: dmanzato@ic.unicamp.br

Nelson L. S. da Fonseca
University of Campinas, Institute of Computing,
Av. Albert Einstein, 1251, P.O. Box 6176
13084-971, Campinas, SP, Brazil, e-mail: nfonseca@ic.unicamp.br

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_22, © Springer Science+Business Media, LLC 2010

1 Introduction

With the predominance of client-server architecture in the majority of computing systems until just a short time ago, hitherto, there was no concern with ensuring certain patterns of behavior by the constituent entities, both clients and servers. In this architecture, while the servers are already intrinsically made available to attend to requests, it is not expected that clients will have any type of helpfulness towards one another or towards the very servers that attend them. This asymmetry, peculiar to this type of architecture, assures that each role is properly performed by the respective constituent elements, either serving or requesting services.

Nevertheless, decentralized architectures have recently emerged as a solution for the well-known problems of scalability imposed by client-server architecture. In these architectures, the roles of serving and being served are not always asymmetrical, that is to say, one selfsame participant may at times cooperate and at others enjoy the cooperation of others. Moreover, quite often the figure of central authorities or servers is lacking, which implies an autonomy for the entities to decide on the patterns of behavior they will display, with the power to act selfishly, lavishly, selflessly or quite simply fairly, that is to say, cooperating only when possible.

Peer-to-peer and ad hoc networks, plus multi-agent systems, are examples where the agents display this kind of autonomy to opt for patterns of behavior. However, the feasibility of such systems is strongly based on the premise of cooperative behaviors. Moreover, the concept of synergy is fundamental to such systems, that is, it is expected to exploit the local resources of the participants so that their joint utilization results in a better result than the sum of the respective individual utilization [22]. Thus, leaving the decision to cooperate or not under the sole responsibility of the participants, without any outside influence or incentive, entails putting at risk not only the system's efficiency, but also its very feasibility.

To aggravate the problem mentioned, the option of assuming cooperative behavior incurs costs, as resources are consumed, such as bandwidth, storage, processing, memory or even energy. Particularly on mobile devices, common in wireless networks, these resources can be precious, increasing yet further the entities' lack of interest in cooperating selflessly. Studies carried out in file-sharing peer-to-peer networks show that just 30% of users share their resources and, further, that the majority of requests are met by just 1% of the principal clients [1]. Such levels of cooperation would destroy the feasibility of the majority of decentralized systems mentioned, while imposing a particular loss on applications for content distribution that are time-sensitive [12–14], as is the case of media-streaming peer-to-peer networks [6, 11, 25–27].

Thus is it that incentive mechanisms for cooperation have recently been receiving special attention in the media, at the same time that decentralized systems are gaining ground over the traditional client-server architectures. In general, these mechanisms must be designed so as to discourage uncooperative behaviors, taking into consideration the naturally heterogeneous nature of clients, and also certain circumstances where cooperation is not possible, either for a static reason, for instance, limited hardware, or for a dynamic reason, such as network congestion. In such circumstances, the incentive mechanism must exempt the client from penalties, which is not always at all simple to implement, as will be addressed in Section 3.

This chapter sets out a study of incentive mechanisms for cooperation. As the number of approaches proposed in the literature is huge, we have opted to place greater emphasis on the different *incentive patterns* used in common by the various *incentive schemes*. An incentive pattern consists of a set of abstract mechanisms that the incentive schemes can apply, with a view to stimulating cooperation [22, 23]. An incentive scheme may use more than one incentive pattern, implementing their abstract concepts so as to be effective for the peculiarities of the application that will use it. It is hoped, with studying incentive patterns, to furnish information so that a broader scope of existing incentive schemes can be understood, and also for analysis of future proposals still to come.

The rest of this chapter is organized in the following manner: Section 2 discusses the different types of uncooperative behavior, including a proposed taxonomy [20, 21]. Section 3 seeks to study the incentive patterns mentioned, considering properties such as type of remuneration, effectiveness against uncooperative behaviors, anonymity and scalability. Section 4 sets out a selection of works related to incentive schemes, comparing them with the patterns seen in the previous section. Lastly, some closing remarks are made in Section 5.

2 Characteristics and a Classification of Uncooperative Behaviors

An understanding of the different types of uncooperative behavior is necessary for a correct analysis of the existing incentive patterns, and also for the conception of new incentive schemes. This section is organized according to the taxonomy of uncooperative behaviors proposed in[20, 21], which can be seen in Fig. 1.

According to [20, 21], a cooperative *transaction* consists of two phases: *negotiation* and *processing*. In the first phase, negotiation, the entities reach an agreement on the services rendered and respective remunerations. The importance of this first phase lies in the equalization of differences of resources, standards of access and asymmetry of topology between the entities. For example, an entity overburdened with network traffic or with a limited link might present a routing cost higher than the others. In the second phase of the cooperation transaction, processing, we have the cooperation and respective remuneration. In this context, it is customary to denominate the entity that cooperates as *provider* and the entity that enjoys the services of the former as the *consumer*. These terms, along with the operations of the phase of processing cooperation transactions are illustrated in Fig. 2 [22].

Cooperation is beneficial to the consumer entity, which repays the provider entity for the favor received by means of a remuneration, so as to encourage it to continue with this cooperative behavior. In the absence of this remuneration, that is to say, without an incentive, the provider entity might refuse to cooperate, which would characterize *selfish behavior*. This type of *uncooperative behavior* may be

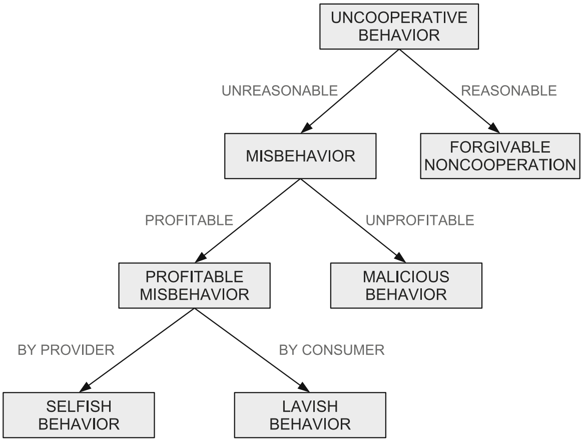


Fig. 1 Taxonomy of uncooperative behaviors [20, 21]

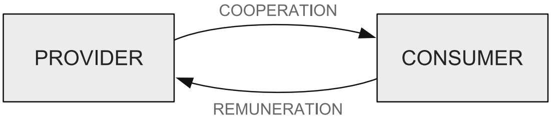


Fig. 2 Terminology of a cooperation transaction [22]

classified as a *profitable behavior*, as the provider entity refuses to cooperate with the consumer not seeking to harm it, but rather to save its own resources *with a profit*, as it would obtain nothing in return as remuneration. Going further, this selfish behavior may be classified as *misbehavior*, that is, it may be characterized as *unreasonable*, given that the provider entity was in a position, with abundant resources, for instance, to cooperate, unlike an entity in a contention situation where saving its resources would be vital – such as the power battery of a mobile device at the end of its charge, for instance. As examples of selfish behaviors, one might mention a user who refuses to make his files available to the others, in a peer-to-peer network for file sharing; a TV viewer who refuses to pass on video streaming in a media-streaming peer-to-peer network; and an agent refusing to pass on packets in an ad hoc network.

When the consumer entity remunerates the provider insufficiently, or also makes use of the same remuneration to consume more resources that agreed on in the negotiation phase of the cooperative transaction, we are faced with *lavish behavior*. Just like selfish behavior, this can be justified as profitable, as the consumer entity is not aiming at simply harming the provider, but rather at obtaining an advantage over it. What differentiates profitable behavior into being selfish or lavish is the fact of its being practiced *by the provider* or *by the consumer*, respectively. Lavish behavior can also be classified as misbehavior, characterizing it as unreasonable, for reasons analogous to those presented for selfish behavior. As examples of lavish behaviors, one might mention users who always download the same files instead of storing them locally, in a peer-to-peer network for file sharing; viewers who request a video quality better than its peripherals can reproduce in a video-streaming peer-to-peer network; and an agent who sends unnecessary packets in an ad hoc network, such as excessive advertising for the services he offers.

Misbehavior may be *unprofitable*, that is to say, the agent misbehaving might aim at quite simply harming the others, without obtaining profit for himself. In this case, we have *malicious behavior*. In [20, 21], it is provided a categorization of different types of uncooperative behaviors into separate layers or domains – link, network, transport, discovery and application. Malicious behavior only takes place when, even though unprofitable in the layer in question, it becomes profitable in higher layers. For example, in an incentive scheme based on reputation, spreading defamatory information would consume resources of the network layer, and is, therefore, not a profitable behavior in this domain of cooperation. Nevertheless, in the domain of the application layer, this behavior would be considered profitable if another entity providing services were excluded from the network due to its bad reputation.

Considering all the domains of cooperation together, malicious behavior would only be displayed by dishonest entities, that is, those whose aim is not to gain an advantage over the other, based on logical rules for the maximization of their benefits, but rather to degrade the performance of the system as a whole or of certain specific entities. In this case, it is not appropriate to address this malicious behavior through incentive schemes, but rather through additional mechanisms that detect and effectively punish the entities responsible. As an example of malicious behavior one might mention attacks of denial of service – a type of attack where an entity makes another unavailable by bombarding it with multiple requests for service, in the different types of existing networks.

Lastly, an uncooperative behavior may be classified as *reasonable* when the entity thus behaving is in fact unable to cooperate due to the scarcity of its own resources. In this case, we have *forgivable noncooperation*. Scarcity of resources may be permanent, when the entity shows limitations on resources such as computing power, memory, bandwidth or energy; or transient, when, for instance, the entities face situations of overload or connectivity problems. In both cases, an ideal incentive mechanism should spare the entity from the obligation of cooperation, that is to say, allow it to enjoy the cooperation of others while demanding nothing in return. Naturally, it is important to make a distinction between forgivable noncooperations and other misbehaviors, which poses a problem for incentive mechanisms [20, 21]. One may mention, as examples of forgivable noncooperations, an entity that refuses to cooperate because it has become a routing bottleneck in an ad hoc network, or a user that needs to limit the number of connections attended to, as he made available a popular file in a peer-to-peer network, quickly becoming the provider of many transfers, which exhausts his bandwidth.

It should also be mentioned that use of a given incentive mechanism may lead to new uncooperative behaviors, over and above those it seeks to combat. For example, in a scenario of remuneration by reputation, an entity may be defamatory about other entities in the network so as to boost its relative reputation, and even manage to exclude certain entities from the group of providers that compete with it. Still within the same scenario of reputation, a group of entities may put out false mutual praises so as to enhance their reputations within the group, acting dishonestly towards the others. Such uncooperative behaviors would not exist if the reputation mechanism in question were not being used.

In a scenario of remuneration by payment, certain uncooperative behaviors may likewise be induced, when remuneration is linked to the scarcity of resources of the provider entity. This type of remuneration is denominated flexible [20, 21] and is addressed in the negotiation phase of cooperation transactions, as mentioned before. In this scenario, a consumer entity might maliciously prevent a given provider entity from cooperating with the others, which would not consume its resources, consequently, reducing their value, as they would now abound. On the other hand, a provider entity might manage to raise its remuneration by making its competitors unavailable, for example, by attacks of denial of service [20, 21]. In this case, such attacks are considered profitable rather than malicious, since they bring benefits to those entities that practice them. Once again, these induced uncooperative behaviors would not exist if the incentive mechanism were not being applied.

3 Comprehensive Study of Incentive Patterns

Study of the various incentive patterns is important to enable an understanding of a broader scope of incentive schemes, as well as for an analysis of future proposals. This section is organized according to the taxonomy of incentive patterns proposed in [22, 23], which may be seen in Fig. 3. Firstly, trust-based patterns are discussed, and later on those based on trade.

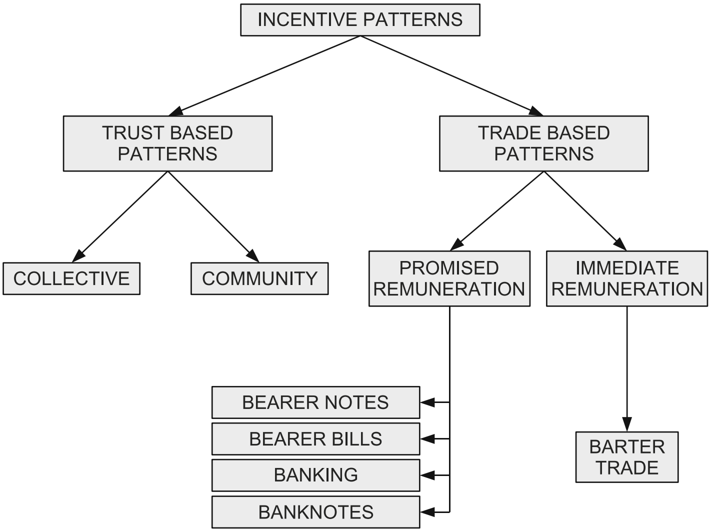


Fig. 3 Taxonomy of incentive patterns [22, 23]

3.1 Trust Based Incentive Patterns

In trust-based patterns, provider entities cooperate with consumers because they trust them. This trust may be static – it does not vary over time and as a result of their behaviors; or dynamic – the opposite case. Thus, there is no explicit remuneration from the consumer to the provider. On the contrary, the repayment for the cooperation takes place naturally, because: (1) the entities make up a group with the same ideals, or (2) because the reputation of the provider has been raised, and as a consequence other entities will later on cooperate with it. These two cases are presented below.

3.1.1 The Collective Pattern

In the collective pattern, entities mutually help one another because they share the same ideals, which are common to the collective group, without individual interests. Thus, when the group wins, everybody wins, when the group loses, everybody loses. The upshot of this model is that no entity is tempted to not cooperate with a view to gaining an advantage over the rest, since the group as a whole would be affected, and on top of this, no gain in terms of resources savings would be obtained individually.

Making an analogy, one might compare this model to a corporate organization. Taking into account that in such a scenario all profits and losses are shared equally among its partners, none of them are motivated not to collaborate with the productivity of the other, given that their interests are in common. Another comparison, from the computing field, is that of an ad hoc sensor network. In these networks, it makes no sense, for example, for one sensor to cooperate less than another, since they all pursue the same goal. The abstraction of a collectivity seen as one sole entity allows the collective pattern to be combined with the others, set out below. For example, one may consider each entity belonging to the other models as being a set of machines, instead of just one.

This incentive pattern belongs to the scope of patterns based upon trust, as the entities cooperate because they have mutual and unconditional trust in each other – as they are part of the same collective. What is involved is static trust, that is, the reputation of the members does not vary over time and as a result of their behaviors. For this very reason it is important to guarantee that each entity belonging to the collective is in actual fact a genuine member. If not, one might have, for instance, an entity from another collective enjoying the resources of the group without offering anything in return, as its interests and responsibilities would be focused on another group.

While other incentive patterns always display some type of remuneration for the entity that cooperated, such as reputation, currency or service in return, the collective incentive pattern is the only one that dispenses with this procedure. This feature derives from the collective nature of the group and the lack of individual interests. As we shall see later on, in the other patterns, this remuneration, when not consumed immediately, may be stored in different locations – at the provider, consumer or even at third-party entities.

The three main types of uncooperative behaviors – selfish, lavish and forgivable – are naturally addressed in this incentive pattern, given their intrinsic characteristics. The selfish and lavish behaviors do not occur due to the lack of individual interests, while forgivable noncooperation is tolerated due to the unconditional trust existing among the members of the group. This is the incentive pattern that best addresses forgivable noncooperations, exempting them from punishments.

One drawback of this pattern, and of the others based on trust, is the impossibility of maintaining the anonymity of the entities, since they need to identify themselves to prove that they are trustworthy. Given that the entities have static situations of trust in this model, it is the consumers who must relinquish anonymity, identifying themselves as a member of the group (the provider entities need not identify themselves as they have their reputations unchanged after their cooperation). Depending on the nature of the application in question, the characteristic of anonymity may be particularly desirable. For example, one may mention an on-demand video service whose programming is characterized as "adult", where, in consequence, viewers would prefer not to be identified.

Another disadvantage of this pattern is restricted scalability. In this context, scalability is understood as being the capacity of the incentive pattern to remain efficient and effective as the number of entities participating increases. In general, trust-based patterns are not highly scalable [22, 23]. Specifically for this case, a relatively large collective may present challenges to ensure that its entities are authentic (that they are not exclusively from other groups), since it is impracticable for all of them to know each other mutually in a numerous group.

Application of the collective incentive pattern in peer-to-peer networks, while somewhat specific, may be highly useful. In file-sharing networks, one may have select and closed groups of users who cooperate with one another, in a manner similar to what happens to the private trackers of the BitTorrent system [3]. Normally these groups allow the inclusion of new users only after invitations made by some member, which restricts the entry of uncooperative users. In video-streaming networks one may have the same concept, especially considering the growing quantity of cooperative IPTV (Internet Protocol Television) systems emerging. However, due to the restricted scalability of the incentive pattern, along with issues of authenticity of the members mentioned above, it is important that the number of members of the groups should not grow too much. In any event, the idea of combining this incentive pattern with the others existing, through the abstraction of a collective as being one sole entity, is applicable to any context of peer-to-peer networks.

3.1.2 The Community Pattern

In the community pattern, unlike the previous pattern, the entities cooperate with one another based on individual interests, without the concept of the collective. In cooperating, an entity boosts its reputation in that community, and this reputation will be necessary for it to be able to enjoy cooperation from others later on. In this context, a given entity might refuse to cooperate with another whose reputation is poor, or even more radically, if the latter's reputation is lower than its own. In other words, the motivation for the entities to pursue the enhancement of their reputations, and, consequently, cooperate more, is for them to be qualified to enjoy cooperation from the largest possible number of members of the group.

One may compare this model of incentive with a group of students in the same class. The more helpful ones will always manage to get help from their classmates when they need it, as they have already done this on previous occasions, thus achieving a good reputation in the group. Meanwhile those more self-centered, who have refused to help in the past, will quite surely face difficulties to achieve this in the future. Various computing systems also use this model; and some peer-to-peer networks for file sharing may be cited, as is the case of Kazaa [15]. In these networks, generally, a user may make various transfers of files from others with reputations below his own. On top of this, on receiving multiple requests, a given user may prioritize attention to those with better reputations.

Just like the collective, this pattern also belongs to the scope of trust-based patterns, given that the entities cooperate with those who display greater trustworthiness, that is to say, good reputations. Unlike the collective, this trustworthiness is dynamic, varying over time and as a result of the behaviors displayed. Consequently, it is necessary to ensure that the entities do not cheat on reputations in a manner divorced from reality, with the defamation of good cooperators or false praise of those who do not cooperate. This issue is particularly delicate when this pattern is combined with the collective. Given that each entity of the community is an abstraction of a collective where collectivism is practiced, members of this latter may cooperate only among themselves, which is already foreseen in the collective pattern, boosting their reputations in the community, and, consequently, enjoying resources from outside the group, without providing anything in exchange. One way of getting around this problem would be to attribute reputations to the groups instead of to each individual member of the collective groups [22, 23].

As the remuneration in this incentive pattern is reputation, it is interesting to note where this is stored. Going back to the analogy of the students in the same class, it would make no sense for one of them to assert to the others that his reputation is good, or rather, that he himself is a helpful person. Thus, in this model it is the consumers that need to remember from which providers they have enjoyed cooperation, and, further, how much each one of them cooperated, so as to make it feasible to quantify the individual reputation of each one. Normally, these reputations are not absolute, but rather relative to the rest of the group. For this to come about, overall knowledge of all the reputations is necessary, which can be obtained by spreading such information in the network or by its being stored at central entities. Naturally this latter approach would raise disadvantages, as being one sole critical point in the system.

Selfish and lavish uncooperative behaviors are combated by this pattern, provided the information on reputations is available to all entities in the group alike. Otherwise a certain entity might refuse to cooperate with a given sub-set of the community, operating only with those in which it had future interests. Lavish behavior, in particular, could be fought effectively with the provider diminishing the reputation of a consumer if the latter displayed such behavior. In this case, the reputations would be stored or changed by the providers too, as well as the consumers. However, forgivable noncooperation is not as well handled as in the collective, where trust was unconditional and static. In any event, given the difficulty involved in differentiating lack of cooperation into selfish or forgivable noncooperation, one may take into account the track record of the entity, that is to say, those who have built up a certain trustworthiness could refuse to cooperate in a situation of overload, without this having a negative impact on their reputation.

As mentioned above, the impossibility of maintaining the anonymity of the entities is a drawback for trust-based patterns. In the case of this model, as the trustworthiness of the entities is dynamic, both consumers and providers must identify themselves – the consumers to prove that they deserve the cooperation they seek, through their reputations, and the providers so as to have their reputations boosted, after completion of their efforts of cooperation.

Given that this is a trust-based pattern, it also displays limited scalability. In particular, the existence of many entities in the group may interfere with the spread of information on reputations which, as we have seen, helps in combating misbehavior. Even if this information on reputation were not propagated, scalability could still be compromised, as it is unlikely that the same pair of entities would cooperate with one another multiple times in a numerous group, which makes the local information useless. Use of central entities for the storage of these reputations, already suggested, may also impose bottlenecks on the system's performance, affecting even further the scalability in question.

Application of the community incentive pattern in peer-to-peer networks is widespread and well-known, representing one of the leading paradigms in the area of incentives to cooperation: reputation systems. In file-sharing networks, as already mentioned, this pattern is typically used to restrict availability of the contents in the network, favoring members with better track records of behavior. In media-streaming networks, there are various proposals for incentive schemes that use reputation for various ends: a stimulus to longer sojourns, seeking to reduce breakdowns in distribution trees; the selection of better stream providers, to obtain a differentiated service; a model to threaten selfish clients, so as to prevent uncooperative behavior. All these scenarios are illustrated in Section 4. Given that the reputations are dynamic and reflect the historical behavior of the members, they can be applied in heterogeneous groups, which may or may not contain uncooperative entities, which makes this incentive pattern less specific than the collective. Nevertheless, due to the large number of participants that any peer-to-peer network has the potential to achieve, it is important to consider the issues of limited scalability of this pattern, the storage and availability of information on reputation and occurrences of defamation of good cooperators or false praise for those who do not cooperate.

3.2 Trade Based Incentive Patterns

In trade-based patterns, the provider entities obtain an explicit remuneration from consumers for their cooperation. This remuneration consists of a return of cooperation, which may occur immediately, while the cooperation transaction occurs, or later on, through a promise, which will be honored by the consumer entity itself or by a third-party entity in its favor. These possibilities are discussed below.

3.2.1 The Barter Trade Pattern

In the barter trade incentive pattern, the entities cooperate so as to complete a direct exchange of favors, which exchange occurs simultaneously, that is to say, while one entity is cooperating, it is also enjoying, at the same time, cooperation from the entity benefiting from its cooperation. The upshot is that neither of the entities leaves obligations pending after termination of the cooperation transaction, and also, those entities that refuse to cooperate obtain nothing from the others. This is the only incentive pattern in which the members assume symmetrical roles in the interactions, with consumer acting as provider and vice versa, at the same time.

An example to illustrate this model is fairs for exchange, popularly know as "swap marts". At these fairs, people exchange objects for keeps, without the use of currency, and thus they must have something that belongs to them to obtain a new object that they seek. Moreover, after a certain trade, neither party owes anything to the other, as he has already paid for what he received, at that very same moment. Typically common, these days, are fairs for swapping books, where people barter books already read for those they would like to read. In the computing field, some systems may be cited, as is the case of BitTorrent [2, 9], whose incentive mechanism will be better described later on. In this network, the greater the sending bandwidth made available by the client, the greater the receiving rate obtained, or rather, the more you cooperate, the more you gain.

This incentive pattern belongs to the scope of patterns based on trade, as the entities cooperate to receive an explicit remuneration in return, unlike those addressed previously, in which the cooperation is on trust. This is the only incentive pattern that naturally displays the characteristic of persistence, that is, remuneration occurs even when the entities participating in a cooperation disconnect from one another straight afterwards [22, 23]. For the other patterns to display this characteristic, it is necessary to introduce a third entity, for example, a bank, which is trustworthy and is always available to honor the promises made by the entities that are related to it. On the other hand, the introduction of this third entity may breach another characteristic that is intrinsic to the barter trade incentive pattern: the characteristic of localization, which is the lack of need for interaction with third-party entities in operations of cooperation and remuneration [22, 23].

Given that remuneration of cooperation in this incentive model takes place immediately and consists of another cooperation in exchange, it need not be stored, as happens in the other patterns, except in the collective. Consequently, difficulties with spreading information on reputations in the network, repetitive reputation calculations, the introduction of central entities or distributed storage of reputations are not necessary. This fact makes the barter incentive pattern relatively simple to implement, although too powerful.

If on the one hand uncooperative selfish and lavish behaviors are very well handled in this incentive pattern, considering the requirement of immediate remuneration, this does not happen with forgivable noncooperation. Due to the symmetrical nature of the cooperation protocol, those entities that do not cooperate, even if justifiably, cannot enjoy any benefits. However, one can get around this limitation by combining this pattern with the collective, when appropriate. In this case, for each collective where the collective mechanism exists, the cooperation of some members with the outside medium (through barter) might offset the lack of justified cooperation from other members of the same collective.

In this incentive model, both entities participating in a cooperation can remain anonymous. As explained in [22, 23], trust, and the consequent breakdown of anonymity are necessary when remuneration does not take place simultaneously. The term trust, in this context, is not restricted to trust-based incentive patterns. For example, if a consumer entity remunerated a provider with a promissory note, check or banknotes, incentive patterns to be seen further on, the creditor entity must trust that the debtor will honor the promise in the future, will have funds in its bank account, or also that the notes used are genuine, respectively. The barter pattern avoids such risks with the use of simultaneous remuneration, and also with the non-use of promises, that is to say, guaranteeing that payment will also be a cooperation in exchange.

Another great advantage of this incentive pattern is its scalability, a direct consequence of the linkage in time between initial cooperation and the respective remuneration. The lack of need to store remuneration, the characteristic of localization and freedom from the obligation of mutual trust between the entities make this one of the most scalable incentive patterns among those existing. Furthermore, considering that an entity will only act as a provider if it is also interested in the cooperation of the consumer entity, the larger the group, the greater the chances of finding pairs whose interests match.

Among the drawbacks of this pattern, one may mention the difficulty of the equivalence of the transactions. For this to occur it is necessary for the parties to remain accessible throughout the whole transaction. Moreover, additional mechanisms may be necessary so as to ensure remuneration in exchange, even if this is synchronous, preventing lavish behavior [22, 23]. Lastly, quite often, use of this incentive pattern is not feasible, that is to say, the use of deferred remunerations is more appropriate.

Application of the barter incentive pattern in peer-to-peer networks became popular with the BitTorrent system, in file-sharing networks. As it was a great success, various other similar proposals arose, principally in media-streaming networks. In particular, consideration was given to the very logistics of content distribution with scaling the BitTorrent, which inspired media-streaming peer-to-peer networks in mesh format, such as CoolStreaming [28]. Given that in both networks with a mesh and those with trees the synchronized distribution of content is common, in which members use their incoming bandwidth to receive the streaming desired at the same time that they employ their outgoing bandwidth to pass it on to the other participants, the barter incentive pattern proves to be an excellent option, especially considering its advantages in relation to the other existing patterns (anonymity, persistence, scalability and localization).

3.2.2 The Bearer Notes Pattern and the Bearer Bills Pattern

In the bearer notes and bearer bills patterns of incentive, entities also cooperate to obtain an explicit remuneration in exchange. The difference between them and the barter pattern lies in the temporal de-linkage between cooperation and remuneration, which allows the provider entity to enjoy its payment at a point in time after its cooperation. In this context, a promissory note may be understood as a promise of cooperation in favor of its bearer, to be honored in demand in the future [22, 23]. In the bearer notes pattern, it is the consumer entity that issued the note that must honor it later on, that is to say, the roles of issuer and debtor of that note are assumed by the same entity. In the case of the bearer bill, it is another entity, on an order from the issuing entity, that must honor the bill in the future, that is, the roles of issuer and debtor are assumed by different entities. In both cases, the motivation for the entities to cooperate acting as providers is to obtain promissory instruments, which make them creditors in the group, and therefore capable of enjoying cooperation in the future, when they wish.

These incentive models can be compared to a hypothetical society in which money is not used as remuneration. In lieu of this, every time a person rendered his services to another, this latter would give him a kind of “service voucher” in return, in his specialty. For instance, on repairing the electrical installation of the residence of a doctor, an electrician might gain a medical consultation to treat his health in future. This would be nothing else but a bearer note, issued and honored by the same entity, the doctor, who enjoyed a certain cooperation, the service of the electrician. In a similar way, the doctor might also be married to a dentist, now remunerating the electrician with dental treatment, to be performed by his spouse. In this case we would have a bearer bill, issued by an entity, the doctor, and honored by another, the dentist, to remunerate the cooperation received.

In the computing field, the bearer note and bearer bill incentive models may also be recognized in other systems, such as the old BBS systems and the present-day IRC forums. In these systems, the server, on receiving an upload from a user, remunerates him by releasing a credit for downloading, which may be done at any time in the future. While this procedure illustrates a bearer note (the server issues and honors the promise), a bearer bill may be exemplified in the same systems with a transfer of credits from the quota of one user to another, a typically common operation. Thus, the user requests an upload from the server, which cooperates and, consequently, obtains remuneration through a deduction from the quota of that user. Instead of this latter making an upload later on to honor his promise of repayment to the server, he asks a friend, another user of the system, to make this contribution, and then transfers the credit obtained to his account, taking it out of the red. We have, therefore, one entity as issuer of a promissory instrument (the user whose download quota was reduced), another entity as debtor, the user who effectively honors the promissory bill, making the upload) and, lastly, a third party entity as creditor (the server, which made the initial cooperation).

Just like the barter incentive pattern, the bearer notes and bearer bills patterns also belong to the scope of trade-based patterns, since the entities receive an explicit remuneration for their cooperation. We may subdivide this category into two: on the one side, immediate remuneration, where we find the barter pattern, on the other, deferred remuneration, where we have the bearer note and bearer bills patterns, plus other to be presented. In this second subcategory, given that repayment does not occur immediately, and that what guarantees it in the future is a promise which, in this context, is in the form of promissory instruments, it is important to ensure the authenticity of these promises, so that dishonest entities do not forge them to gain an advantage over the others.

Promises in the form of promissory instruments are the types of remuneration in the bearer note and bearer bill incentive patterns. Unlike the community pattern, in which the remuneration, in the form of reputation, is normally stored at the consumers, in these patterns, it is stored at the providers, who may be regarded as the bearers of the instruments. It is interesting to note a new concept introduced by these incentive models: that of transferring remuneration among the entities, that is to say, after receipt of its remuneration, a provider entity may, possibly, pass it on to others that come to cooperate with it, in this way, avoiding issuance of a new promissory note.

Selfish uncooperative behavior is fought effectively in both the bearer bill and bearer notes patterns, as entities do not cease to cooperate, seeking to obtain the promissory instruments. However, as to lavish behavior, while it is unconditionally avoided in the bearer notes pattern, since the entity that becomes the debtor of the promissory is the same that issues it, in the bearer bills pattern it is necessary for the entity that becomes the debtor to be trustworthy to contain abuse by the issuer. If this premise is not true, an entity may feel itself at liberty to issue third party promissory notes at the cost of any other entities, passing on to them all its repayment liabilities, which would probably never be honored. With regard for forgivable noncooperation, the bearer notes pattern does not handle this very well, given that the entity that issued the promissory note will, itself, have to become the debtor of that promise, whether or not it is in a position to do so. This incentive pattern, with barter, and for analogous reasons, are the ones least able to deal with forgivable noncooperation. The bearer bills pattern, on the other hand, can handle this issue better. If an entity needs a certain cooperation but is unable to become debtor of a promise, in a situation of overload, for example, it may convince another entity to become a debtor on its behalf, in this way issuing a bearer bill.

With regard for the anonymity of entities in the bearer note and bearer bills patterns, in general both issuers and debtors of these need to identify themselves so as to prove their trustworthiness [22, 23]. As both roles are assumed by the same entity in the bearer notes pattern, the drawback of breach of anonymity is the same as happens in the trust-based patterns, that is, there is no way for the consumer to remain anonymous. In the case of the bearer bill, however, if the debtor of the promissory bill is sufficiently trustworthy, it would be possible for the issuer not to identify itself, provided the chances of issue of false bearer bills is eliminated, or rather, those for which the debtor does not authorize their participation in the issuer's transaction. In this context of trust based on identity, it is interesting to note how the bearer note and community patterns lead the consumer to want to change or renew his identity, to get rid of the obligations assumed, while on the other hand, they lead the providers to want to maintain it, to receive the repayments for the cooperation they have already rendered.

As a result of the need for trust of the entities in these patterns, scalability is restricted. However, supposing that these patterns were being applied in an ambient free from dishonesty, the problems linked to proof of trustworthiness would not exist, making the scalability of these patterns less restricted than the trust-based patterns. In these latter, as we have seen, such information is indispensable in any event, as it determines the levels of cooperation among the entities.

3.2.3 The Banking Pattern

In the banking pattern of incentives, just as in the bearer notes and bearer bills patterns, entities cooperate to obtain a specific remuneration in exchange, which remuneration will be enjoyed at a point in time after the cooperation. The difference lies in the type of remuneration which, instead of being a promissory note, is a check. However, in this scenario, one may understand a check as being a bearer bill whose debtor is a bank [22, 23], that is to say, a check is a special type of bearer bill, to be issued by the consumer entities as remuneration for the provider entities after the respective cooperation. The upshot of this is that each entity, in this incentive pattern, has a bank account at an associate bank. In this way, the bearer of a check presents it to his bank to have his account credited. If the bank in question is different from the bank issuing the check, a transaction between banks becomes necessary. In brief, the motivation for entities to cooperate is to obtain checks as payments that will be credited to their accounts, enabling them to pay for future cooperation events they may come to require.

One may compare this incentive model to the use of checks in the banking system of present-day society. Each person has one or more bank accounts, associated to specific banks, and uses checks to remunerate the services he has enjoyed. The bearer of a check presents it to his bank to have his account credited, and it is then up to the different banking institutions to make the due monetary clearing between them. People, then, render services to others to increase their accounts and to be able to pay for what they wish, later on.

In the computing field, one may mention ad hoc networks that make use of incentive mechanisms based on this pattern, as is the case of TermiNodes and Sprite [20, 21]. While in TermiNodes [4, 5] each entity has a security module that manages its own bank account, and thus this incentive model implemented is totally distributed, in Sprite [29], there are dedicated nodes that act as banks, centralizing the management of the accounts that belong to the same groups, and employing inter-bank transactions when necessary.

This incentive pattern, just like the bearer notes and bearer bills patterns, belongs to the scope of trade-based patterns, and also to the category of deferred remuneration, where there is a temporal de-linkage between cooperation and remuneration. In this way, given that what guarantees repayment of future cooperation is the checks issued, it is important to guarantee their authenticity, so as to avoid forgeries. Nevertheless, as the banks are well-known entities and these, in their turn, know the identity of their clients, we have it that the risk of forged checks is less than that of forged promissory instruments.

The type of remuneration in this pattern, as mentioned above, consists of a check. Like the other patterns with deferred remuneration, the checks are also stored at the providers, until they are presented to the banks. It is important to note, in the case of this pattern, the distinction between storage of the checks (promises) and the storage of the bank accounts that each entity has. These latter, usually, are held in banks. Just as in the case of promissory instruments, checks may also be transferable.

Selfish uncooperative behavior is suitably fought in the banking system pattern, as entities do not cease to cooperate with the aim of obtaining checks that boost their bank accounts. Lavish behavior is also handled well: as the banks are trustworthy, always honor their promises and know the identity of their account holders, which means that abuses by issuing entities fall back on them, ensuring that this kind of misbehavior does not occur. With regard for forgivable noncooperation, this can be dealt with in a way similar to the bearer bills pattern: if an entity needs cooperation but is prevented from cooperating in return to "cover" its bank account, it may convince another entity to do so, subsequently depositing the resulting check in its account.

Analogously to what happens in the bearer notes and bearer bills patterns, both the issuing entities and the debtors of the checks need to identify themselves to prove their trustworthiness [22, 23]. However, as the debtors of the checks are the banks, who do not mind revealing their identities, we have it that the question of their anonymity is no longer a problem in this incentive pattern. Considering, once again, that the banks are trustworthy and know the identity of their clients, there is also a possibility that the issuing entities need not identify themselves, provided the chances of false checks being issued is eliminated, that is, those without the consent of the respective bank.

Given that the set of debtor entities that needs to prove their trustworthiness to the providers is limited to the banks, and therefore considerably reduced, we have it that this incentive pattern scales better than the bearer notes and bearer bills patterns. Nevertheless, this scalability is limited not only by the accessibility of the banks, but also by the transactions among them that prove necessary. In this context, the existence of one sole bank might, on the one hand, resolve the questions of inter-bank interaction, on the other, it might represent one sole point of access and easily become a bottleneck for the system. The existence of multiple banks, however, while it does eliminate the issue of one sole point of access, reveals its difficulties in interbank transactions.

3.2.4 The Banknotes Pattern

In the banknotes pattern of incentives, the entities also cooperate seeking to obtain an explicit remuneration in return, which will be enjoyed at a time after the cooperation. The difference between this and the other patterns with deferred remuneration once again lies in the type of remuneration which, in this case, consists of the banknotes. One may understand a banknote, in this context, as a bearer note pre-issued by a central authority [22, 23], that is to say, a banknote is a particular type of bearer note which is used by the consumer entities to remunerate the providers. Thus, the motivation for the entities to cooperate is to obtain these notes, which will enable them to pay for any future cooperation they may come to require.

This incentive model may be compared to the use of banknotes in present-day society. People may remunerate services they have enjoyed by means of notes they have in their possession. In this way, people render their services to others to accumulate money in cash, which will be used later on for what they wish. In computing, one may mention systems of a payment mechanism based on exchanges of tokens, without the existence of banks or bank accounts. In these systems, the entities receive a token, which takes on the role of a banknote, when cooperating. Later on, this token can be exchanged for services or cooperation that the entity desires, and is therefore the motivation for everyone to cooperate.

Just like the banking, bearer note and bearer bill patterns, this one also belongs to the scope of trade-based patterns and with deferred remuneration. As the promise that ensures future repayment of the cooperation given is in the form of banknotes, a guarantee of their genuineness is fundamental. Bearing in mind that banks are not present in the transactions of cooperation and remuneration, the fact is that the risk of forgery is greater in this pattern than in the banking pattern, which may require the use of additional security mechanisms, such as an encrypted infrastructure. However, the computing power necessary for installing such an infrastructure may perhaps not be available, as in some of the ad hoc networks set up by entities with limited resources.

The type of remuneration of this pattern, as already mentioned, consists of banknotes. Just as in the other patterns with deferred remuneration, these are stored at the providers, which may be regarded as bearers of the notes. More than in any other incentive pattern, the concept of transfer of remuneration applies to this model. It is also worth reiterating that in the patterns already studied, the remuneration was either kept at the consumer (in the community pattern) or was non-existent (in the collective pattern), or did away with storage as it was immediate (in the barter pattern).

Due to entities' interest in obtaining the banknotes, to enable them to pay for cooperation they want in the future, selfish uncooperative behavior is effectively combated. Lavish behavior is also avoided, recalling that the entities are billed directly for cooperation they enjoy. As to forgivable noncooperation, this can be handled in a manner analogous to treatment in the banking and bearer bill patterns: if an entity needs cooperation but is prevented from doing the same thing to obtain banknotes and make that payment, it may convince another entity to do this for it, that is, to assume responsibility for its debt.

Taking into account, once again, that in this incentive pattern banknotes are pre-issued by a central authority, which assumes the roles of issuer and debtor of that type of promissory instrument, anonymity of the consumer is guaranteed, regardless of the trustworthiness of ant entity involved in the cooperation transaction, just as happens in the barter pattern, only. Nevertheless, considering the possibility of forging the banknotes, there may be a demand for the trustworthiness of the consumer, which would entail a breach of anonymity, that is, either the banknotes are beyond forgery or consumer anonymity is compromised. In any event, the trustworthiness of the central authority that pre-issues the banknotes is fundamental, although this is not a problem.

As the central authority need to not be accessible in transactions of cooperation and remuneration, unlike the banks in the case of the banking system, we have it that this pattern is more scalable than the banking pattern, avoiding the problems related to the accessibility of the banks, already mentioned (availability and inter-bank transactions). Moreover, according to [22, 23], this is the most-scalable incentive pattern among all those presented.

Application of the incentive patterns with deferred remuneration – bearer notes, bearer bills, banking system and banknotes – in peer-to-peer networks is also widespread and well-known, representing another principal paradigm in the area of incentives to cooperation: the systems of payment. One can find these patterns in various proposals for incentive schemes, in both file-sharing networks and also media-streaming. Although the types of remuneration vary from one to another, with the forms of tokens or balances appearing chiefly, all systems of payment have one selfsame objective in common: allowing the de-linkage over time between cooperation and remuneration. Depending on the type of application in question, this factor may lead to a whole series of advantages. For example, in a media-streaming peer-to-peer network, one may attain the following benefits:

- The system may draw on cooperation from users offline, who decide to go on cooperating even when they are not using the system, seeking to accumulate a balance.
- Users using their outgoing bandwidth for other purposes (other applications) may opt not to cooperate for a certain period of time, provided they have a balance to pay for their immediate consumption.
- Less-qualified users may, making use of their balances, obtain a streaming quality better than what their outgoing bandwidth would permit on-line with the barter pattern. In this way, the asymmetry of incoming and outgoing bandwidth of different access technologies ceases to be a problem in multicast video distribution.

- Better-qualified users may continue cooperating with more network bandwidth than necessary, building up a balance for a subsequent financial reward, at the end of a stipulated period.

Given that trade-based incentive patterns, in general, are more scalable than those based on trust, for reasons already discussed, one may state that they are more easily employed in peer-to-peer networks, considering once again the large quantity of participants these may reach. On the other hand, this makes it important to ensure the question of authenticity of the remunerations, and also that service promises will be duly issued and honored, irrespective of the context of the peer-to-peer network employed.

A summary of the incentive patterns studied in this section is presented in Table 1. The main characteristics, pros and cons of each one are pointed out.

4 Selection of Incentive Schemes

Various works concerning incentive mechanisms have been presented in the literature. These works discuss economic issues of peer-to-peer and ad hoc networks and multi-agent systems, taking into consideration the autonomy of the entities to decide whether to assume a cooperative behavior or not. While some propose new incentive schemes that can be related to the incentive patterns already studied, others investigate issues arising from the use of existing schemes, such as an enhancement of quality of the service, boosting scalability, strategic mobility of the entities and new induced uncooperative behaviors. The aim of this section is to provide an overview of the research performed in the area of incentives to cooperation, through the presentation of some selected works.

4.1 Incentive Mechanism for the CoopNet Network

In [16, 17], an incentive mechanism was proposed for the CoopNet system [25–27]. CoopNet is a live video-streaming peer-to-peer network with synchronized distribution, which makes use of Layered Multiple Description Coding, or LMDC, and multiple distribution trees. These trees are sensitive to the constant connections and disconnections of participant nodes, since depending on the position of a given interior node, various descendant nodes may be affected by its disconnection. On top of this, CoopNet also faces the problem of poor cooperation from clients, which has an adverse affect on the system's scalability.

The incentive mechanism proposed in [16, 17] seeks to ease both the problem of poor cooperation and also that of constant connections and disconnections of clients. Thus, it consist of two parts, one of them a primary incentive and the other secondary. The primary incentive implements the barter pattern and seeks to increase cooperation from clients, improving the scalability of the network. The secondary

Patterns	Characteristics	Pros	Cons
Collective	Trust based; Static trust; No remuneration	No motivation to behave selfishly; Easily combined with other patterns; Selfish, lavish and forgivable behaviors well addressed	No anonymity for consumers; Restricted scalability; Authentic membership required
Community	Trust based; Dynamic trust; Remuneration stored at consumers	Applicable to heterogeneous groups; Address selfish and lavish behaviors if reputation is globally available; Address forgivable noncooperation if track record of entities is kept	No anonymity for providers and consumers; Restricted scalability; Propagation and storage of reputation required; Prevention of false defamations and false praises is necessary
Barter trade	Trade based; Immediate remuneration	No storage of remuneration; No pendencies left after cooperation (persistence); No third-party entities required for transactions (localization); Anonymity for providers and consumers; Good scalability; Selfish and lavish behaviors very well addressed	Forgivable noncooperation not addressed; Symmetrical cooperation/remuneration not always applicable
Bearer notes	Trade based; Deferred remuneration; Promises in the form of promissory notes; Remuneration stored at providers	De-linkage between cooperation and remuneration; Selfish and lavish behaviors addressed	No anonymity for consumers; Restricted scalability; Authentic promises required; Forgivable noncooperation not addressed
Bearer bills	Trade based; Deferred remuneration; Promises in the form of promissory bills; Remuneration stored at providers	De-linkage between cooperation and remuneration; Anonymity for consumers possible; Selfish and forgivable behaviors addressed	Restricted scalability; Authentic promises required; Trusted promissory debtors required; Lavish behavior not well addressed
Banking	Trade based; Deferred remuneration; Promises in the form of checks; Remuneration stored at providers and banks	De-linkage between cooperation and remuneration; Anonymity for consumers and low risk of forged promises, given that banks are trusted; Selfish, lavish and forgivable behaviors addressed	Restricted scalability
Banknotes	Trade based; Deferred remuneration; Promises in the form of banknotes; Remuneration stored at providers	De-linkage between cooperation and remuneration; Anonymity for providers and consumers; Enhanced scalability; Selfish and lavish behaviors addressed; Forgivable noncooperation partially addressed	Genuine banknotes required

Table 1 Summary of incentive patterns

incentive implements the community pattern and seeks to increase session times, reducing the breaks in the trees; to this end, it sees to it that clients have reputations that are directly proportional to their times of presence, while the cooperation demanded of them will be inversely proportional to their reputations. With the stimulus of cooperating less, clients remain connected for a longer time, seeking to enhance their reputations.

The barter pattern is implemented through the equation $B_{Oi} = CR_i.B_{Ii}$, where B_{Oi} is the outgoing bandwidth of the i-th client made available for cooperation purposes,

B_{Ii} is the incoming bandwidth used for reception of the stream at the quality desired, and CR_i is its cooperation rate, that is to say, how much the system demands of it in cooperation as a function of its consumption. For example, with $CR_i = 1$, it is required that the client cooperate exactly the same quantity of bandwidth that it consumes; with $CR_i = 2$, it is required to cooperate double its consumption; with $CR_i = 0.5$, it is required to cooperate half its consumption, and so on.

The community pattern is implemented through two equations: $R_i = t_i/(t. + 1)$, where R_i is the reputation of the i-th client, t_i its time of presence in the system and $t.$ the time of presence of the oldest client considered; and $CR_i = UB_{CR} - R_i(UB_{CR} - LB_{CR})$, where LB_{CR} and UB_{CR} are, respectively, the lower and upper bounds for the cooperation rates practiced by the system. The first equation is responsible for calculating the reputation of the i-th client as a function of its time of presence, while the second equation calculates its cooperation rate as a function of its reputation. Thus, each client connected to the system has its own cooperation rate, which varies over time. This cooperation rate is the same used in implementing the barter pattern, that is to say, we have both patterns, barter and community, acting together and implementing the two incentives that make up the mechanism proposed.

One particularity of this mechanism is that, depending on the upper and lower limits adopted in the above equation, one has a deficit or excess of resources in the system. Typically, the values of 0.5 and 1.5 are used, respectively, which allow older clients to cooperate at up to half of what they consume (rates below 1), while more recent clients cooperate up to 50% more than they consume (rates above 1). It is, therefore, a system that uses the excess cooperation from certain clients to allow other clients to cooperate less than what is necessary, thus providing these latter with a stimulus to remain connected to the network for a longer time.

The results show that the primary incentive eliminates all occurrences of blockage, besides doubling, on the average, the qualities received, given that all the clients begin to cooperate unconditionally. This provides strong evidence that the scalability of the system increases. Meanwhile, the secondary incentive presents a cost of use that corresponds to 2 to 4 describers in 16 to the lesser at the quality received. The benefits obtained with this are notable only in situations of a Flash Crowd, in which the arrival rates are high. In these cases, it is possible to reduce the breakages in the trees by around 45%, doubling the average time of presence. Thus, its use is advantageous only when the arrival rates are sufficiently high for the gain obtained to offset the cost incurred.

An alternative employment for the secondary incentive of the mechanism proposed is the economic purpose of keeping clients connected to the system for a longer time. Just as occurs with free analog television, it is the advertisers who pay the costs of distribution, and they also pursue a larger audience. Similarly, in a system of distribution of digital streaming like CoopNet one may, on the one hand, have the advertisers shouldering the costs of the central servers where the service and the distribution trees originate; on the other, the clients of the peer-to-peer network always cooperating, based on the primary incentive, and remaining connected for a longer time, based on the secondary incentive. While the first factor provides greater scalability for the system, so that more clients can be served, the second satisfies the economic interests of the advertisers, who have a larger audience and, consequently, have a great motivation to go on maintaining the service.

4.2 Altruism in Peer-to-Peer Media Streaming

In [8] a framework for the explicit modeling of altruism was proposed for the context of media streaming peer-to-peer networks with multicasting at application level, such as CoopNet [25–27] and SplitStream [6]. This framework considers a system parameter K that relates the bandwidth consumed and cooperated by clients, in a manner similar to the model proposed in [16, 17]. The K parameter may vary from 1 to ∞: when it assumes the value 1, each client cooperates with exactly the same bandwidth he consumes; assuming the value 2, each client cooperates with double the bandwidth he consumes; and so on. The limit of bandwidth cooperated is the capacity of the client's outgoing bandwidth, that is to say, the lower of the following values: (1) bandwidth consumed times K and (2) the capacity of the outgoing bandwidth. Given that parameter K is the same for all clients, unlike the model proposed in [16, 17], it cannot be less than 1, as in this case, the sum total of all the bandwidth cooperated would be less than the sum total of all the bandwidth consumed, or rather, the demand for bandwidth would be greater than supply, and in this way the system would begin to saturate and block new clients.

One notes once again the use of the barter incentive pattern, that is to say, remuneration occurs simultaneously with the cooperation effected. While the client receives the media stream from the higher nodes of the multicast tree (remuneration), he passes this on to its lower nodes, using his outgoing bandwidth (cooperation).

The principal idea of this work is to distribute the excess outgoing bandwidth accumulated in the system equally to all the clients connected, improving the qualities of streaming they receive. That is, given that each client is already receiving its rightful bandwidth, calculated by the ratio between its cooperated bandwidth and the parameter K, the excess bandwidth is distributed gradually, thus entailing the maximization of bandwidth obtained by each client, if this is not yet at its maximum value. This maximum value is either the bandwidth of the stream obtained or the capacity of the client's incoming bandwidth. It is not hard to observe that this approach is only effective in a heterogeneous ambient, where client bandwidth are separate, besides the natural asymmetry of the incoming and outgoing links of each client. If this premise is not true, either there will be a lot of excess bandwidth, when the majority of clients can contribute what the parameter K determines, or there will be no excess capacity to be distributed, when the majority of clients have limited outgoing bandwidth.

A distributed protocol was proposed for calculation of the excess bandwidth to be redistributed. In principle, this calculation would call for overall knowledge of use of all the clients' bandwidth, so as to determine the excess quantity. However, using concepts of multiple distribution trees and codification into multiple describers, presented in [6, 7, 24–27], a proposal was made to try the gradual and periodic admission of each client to the various trees, until the excess bandwidth is exhausted. In this case, through priorities attributed to each client in each separate tree, one can determine the order of the trees in which the client will cease to participate if the system's excess bandwidth is reduced. In consequence, when this is exhausted, clients are disconnected from the trees in which they have lower priorities, to offer a place to those with higher priorities. This distributed protocol was evaluated, via simulations, and compared with what would be the "ideal" protocol, with knowledge of clients' bandwidths.

The results show that the distributed protocol may attain good levels of use of excess bandwidth, if compared to the protocol with overall knowledge. On top of this, it proved the efficacy of allocating the bandwidth received by each client; that this matches the policy of modeling by altruism, as originally proposed. Nevertheless, an analysis was also made of the price of implementing this policy, in terms of breakdowns in the distribution trees. That is to say, each time a certain client is disconnected from a tree, there is an interruption in the reception of the related describer, which causes a temporary worsening of the quality received, until the tree is reestablished. Some breakdowns are inevitable, as is the case of those arising from arrivals and departures of clients in the network. Others are the outcome of the policy proposed, such as those arising from disconnections of clients with lower priorities. It was concluded that the cost of the policy proposed is double the fundamental cost of maintaining the distribution trees.

What did not emerge clearly, though, is how much this additional cost affects the improvement obtained by the excess bandwidth determined by the altruism parameter K. On the other hand, it was asserted that the rate of breakdowns varies little as a function of K (in spite of which, the greater the parameter K, the greater the degree of admission of nodes, the wider the trees and, consequently, the smaller the rate of breakdowns). Moreover, it was shown that small increases in K, from 1 to 1.5, for example, provide good results in terms of increase of excess bandwidth. However, there was no evaluation of the relative gain from the policy, that is to say, how many more describers each client receives on the average, considering the additional capacities afforded by the excess bandwidth accumulated in the system, and how many less, as a result of the temporary losses caused by the breakdowns in the distribution trees.

4.3 Multicast with Incentive in Peer-to-Peer Media Streaming

Another incentive scheme for media streaming peer-to-peer networks based on reputation was proposed in [19], that is to say, it uses the community incentive pattern. It was proposed to the SplitStream architecture [6], yet according to the authors it is sufficiently generic to be used in practically any system with multicasting at application level and with multiple distribution trees. The only requirement of the policy is that the trees should be periodically rebuilt, so as to maximize the chances of a higher node with uncooperative behavior becoming a node below those to whom it denied cooperation in the past, allowing it, thus, to be retaliated. This approach constitutes a threat model based on the experiences that the nodes have had of one another previously.

Some approaches have been proposed to identify the uncooperative nodes in multicast trees. The main difficulty lies in the false positives, that is to say, those nodes that are cooperating, but that sporadically refuse to serve certain requests as they are saturated – which configures forgivable noncooperation, as we have already been discussed. To get around this problem, use was made of a combination of the various approaches proposed, given that each one of them, used individually, was not sufficient to effectively reduce the number of false positives. With regard for the cost of periodically rebuilding the trees, it was estimated that this would be very low, not making the proposal unfeasible. The results show that, for a small fraction of selfish users (5%), the system is able to identify and punish them with a denial of service, which becomes an interesting incentive to cooperation. Studies with higher rates of selfish users were left for future works.

It is interesting to note how the proposal gets around the problem of changing identity when the entity is bad (this issue was already commented before, in Section 3). It is suggested that each one of the distribution trees should be rebuilt at a time, at predefined time intervals. Additionally, a new node joining the network can only sign up for new distribution trees when these are rebuilt. In this way, if a certain node with a bad reputation opted to disconnect and reconnect to “clean up” its bad name, it would have to wait a given period of time to once again enjoy the streaming quality it hitherto had. Naturally, this policy of punishing new arrivals also hurts cooperative users. For example, if students were connecting to the system to attend a lecture, they would need to join k time intervals in advance, where k is the number of distribution trees that the system uses and the intervals are the times between one tree rebuild and another. In this case, if the time were 15 seconds and $k = 16$, for instance, then the students would need to connect 4 minutes before the start of the lecture.

One drawback of the incentive scheme proposed is the fact that it does not consider how much each node must cooperate, unlike the previous schemes, failing to exploit the heterogeneous nature of the clients. This means that two users with different cooperation capacities (outgoing bandwidth) may obtain the same streaming quality (incoming bandwidth) provided only that they do not fail to cooperate, which might call into question issues of the policy’s fairness. As a consequence, this problem might lead to new uncooperative behaviors, such as users declaring outgoing bandwidth smaller than they effectively have, so as not to consume their resources beyond the strictly essential.

4.4 Client Selection with Differentiated Service

Yet another proposal of an incentive mechanism for media streaming peer-to-peer networks was presented in [12–14] and it is percentile-based. Unlike the three mechanisms presented so far, designed for live streaming with synchronized distribution through multicast trees, this mechanism was proposed for streaming on demand with asynchronous distribution, and consequently, remuneration. In this scheme, the entities that cooperate are remunerated with a score, which may be understood as a reputation. This score is mapped to a percentile, which indicates the position, in terms of cooperation, in which the user is classified in relation to the others. For example, a user that displays a percentile of 90% cooperates less than just 10% of the members of the network. The user's percentile is then taken into consideration at the moment when he requests streaming: he can only be served by those clients who have a percentile lower than his. One sees quite clearly the use of the community incentive pattern in this incentive scheme.

For this type of application, one sees that the selection of good candidates to serve a given request is an important issue in improving the quality of the streaming received. Good candidates are understood as being those that display good availability, good rates offered and good network route dynamics, from the supplier to the consumer of the streaming. While the arbitrary selection of candidates reveals an unforeseeable quality of service, which is worsened with the number of suppliers chosen, the selection of good candidates provides good and foreseeable qualities, regardless of the number of suppliers considered. In this way, the motivation for cooperation introduced by this scheme is the increase of the percentile, which allows the entity to make a more inclusive selection of good candidates to serve its requests, obtaining better and more foreseeable qualities of service than those afforded by arbitrary selection. Unlike the majority of incentive mechanisms based on reputation and designed for file-sharing peer-to-peer networks, the idea of this mechanism is not an increase of availability of the content desired, but rather a differentiation of service, which promotes an enhancement in the quality of the streaming received.

For each entity to be able to determine how much cooperation it is advantageous to provide, a function of use of resources was defined. This function takes into consideration the quality of the flow received and the cost of the cooperation. In this way, through simulations, it is proven that the mechanism proposed does provide almost optimal quality of service when the network is not congested. When compared to its non-utilization, one sees that the network cannot achieve the same levels of quality of service only with the cooperation of a few altruistic users. as sometimes happens in the file-sharing peer-to-peer networks. One also concludes that the mechanism may not be efficient when the network is completely congested or idle, situations in which the quality received may be very low or almost optimal, respectively, regardless of its use. Another conclusion is that the mechanism reduces the quantity of redundant data necessary to compensate the loss of packets, typically present in media-streaming sessions and normally controlled by techniques of stream coding, such as FEC (Forward Error Correction) and MDC (Multiple Description Coding).

To get around the issue of unfair change of identities, pursued by users with low percentiles due to their uncooperative behaviors, this mechanism likewise does not privilege newly-arrived users, just as the previous mechanism. Therefore, newly-arrived clients are submitted to the arbitrary selection of candidates to have their requests served, and they are subject to a policy of best endeavor, given that the quality to be obtained is unforeseeable and depends on the altruistic users in the system.

4.5 Incentives in BitTorrent

The incentive mechanism used in the BitTorrent system [2] is described in [9]. It is a scheme denominated tit-for-tat, which is quite simply the direct application of the barter incentive pattern in the context of peer-to-peer networks for file sharing. Users strive to receive a given file at a rate proportional to that they simultaneously deliver this very same file to other users. In other words, the greater the outgoing bandwidth made available by the user for purposes of cooperation, the greater the receiving rate achieved, minimizing the time required for the complete transfer of the file.

In this system, each file made available for transfer is segmented into parts of equal size, in such a way that each user can obtain a different segment and pass it on to the others in the network. In this way, the server needs to transmit the file made available just once, in theory. In practice, quite possibly some segment may become unavailable in the network due to disconnection of the client that maintained it, thus entailing another transfer of that segment by the server.

Each file made available is also linked to the address of a coordinating system (Tracker) responsible for assisting the users to discover which clients possess which parts of the file. However, it is the local application that is responsible for the logistics of the connections. It is found that random graphs display good qualities in terms of sturdiness. To discover which segments of the file must be obtained first, and, moreover, from which users, some policies are defined and used by the local applications, which take such decisions interacting with one another.

The application is also responsible for controlling the relation between the clients' sending and receiving rates, ensuring the efficacy of the tit-for-tat system. Certain strategies determine when and how much the client must cooperate more or less, based on his receiving rate, so as to maximize the utilization of his resources. Given that the consumption of the servers' resources – mainly bandwidth – may be considerably minimized, the BitTorrent system has become a highly useful tool for the publication of files on the Internet, chiefly for open code applications that make available their sources and binary codes, as is the case of some Linux distributions.

4.6 Trading in Trust, Tokens and Stamps

In [18] an interesting incentive scheme based on stamps was proposed as being a generalization of the mechanisms based on reputation (community incentive pattern) and payment (incentive patterns based on deferred remuneration). A comparative study was presented of these two models, listing the properties and types of economies and incentives induced by them. It was demonstrated that the essence of both can be captured by a more generic scheme, denominated stamp trading.

In this mechanism, the entities must obtain, firstly, personalized stamps of the entity from which they wish to enjoy cooperation, before requesting it, so as to have means to make the respective payment. To obtain these personalized stamps, they trade their own stamps, issued by them, or stamps from other entities they have in their possession, respecting a swap or exchange rate that indicates the value of each type of stamp. These values are calculated by a central entity and according to different policies, proposed and analyzed in the work. Irrespective of the policy, every time an entity refuses to cooperate upon the presentation of one of its stamps by another requesting entity, the value of that type of stamp is reduced, that is to say, the trustworthiness or reputation of that entity falls. The value of a stamp may also drop with the number of issues made by the same entity, or rather, the greater the number of stamps of the same type circulating in the network, the smaller will be their value.

One of the issues discussed in this work is whether the reputation of the entities can be considered a currency. It was argued that on the one hand it can, considering that it can be traded so as to induce an economy, but, on the other, it cannot, bearing in mind that it cannot be bought or sold, but rather lost or won. The stamps, in principle, recall the bearer bonds, seen above, since cooperation is effected through their presentation to the entity that issued them. However, unlike the bearer bonds, they are pre-issued and later on traded, subject to variable prices based on the different reputations that exist. In this case, they recall more banknotes, with the difference of having been pre-issued by the entities, rather than by one sole central authority.

4.7 Mobility in Ad hoc Wireless Networks with Incentives

The focus of the work presented in [10] is not a proposal for a new incentive scheme, but rather the study of a strategic mobility that the nodes of ad hoc wireless networks may display when existing incentive mechanisms are employed, such as those of reputation (community incentive pattern) and those of payment (incentive patterns based on deferred remuneration). This mobility is induced by the fact of autonomous entities seeking to maximize the use of their resources. Given that the incentive mechanisms oblige them to cooperate in any event, strategies of geographical movements are then assumed so as to prioritize their interests.

For example, with reputation schemes, the entities seek to minimize the quantity of traffic passed on – to economize their resources, as they do not gain an explicit reputation for passing-on – and they become indifferent to the number of hops necessary for their packets to reach their destinations – there is no charge for the number of hops. In this case, they remain closer to the edges of the network's topology, where there is less traffic to be passed on, and are not concerned that their packets use more hops to reach opposite sides or ends. On the other hand, with payment schemes, the entities seek to maximize the quantity of traffic passed on, since they are paid for each packet passed on, and also to minimize the number of hops necessary for their packets to reach their destinations, as one pays the other entities that pass on the packets. In this case, then, they try to remain close to the center of the network's topology, where there is more traffic to be passed on and smaller distances, in terms of hops, to the different destinations of their packets.

By means of models based on game theory and simulations, it is proven that the strategic mobility of users may degrade performance of the network, due to the inefficient topologies that are formed. Generally speaking, it is a problem caused by uncooperative behaviors induced by the incentive mechanism used. With regard to the inefficient topologies formed, one identifies two trends: one clustering, which leads the nodes to stay as close as possible to one another, and another of dispersion, which leads them to the formation of simple chains. In both cases, even though the final topologies are different, one concludes that strategic mobility, in general, degrades the capacity of the ad hoc wireless networks. Therefore, future incentive mechanisms proposed for this context must consider this factor, as well as fighting the uncooperative behaviors already expected, such as the non-routing of packets due to high energy and bandwidth costs.

5 Final Remarks

We have seen that, with the predominance of the client-server architecture until a short while ago, until then there was no concern with ensuring certain patterns of behavior of the constituent entities, as these had clearly defined roles in the network. However, decentralized architectures have emerged as a form of resolving the problems of scalability imposed by the previous architecture. In this new context, entities both serve and are served. Moreover, they are free to decide between cooperating or not, especially due to the lack of central authorities or servers. These facts, along with the cost of cooperation, which is quite often high, due to the consumption of local resources, lead the entities to practice uncooperative behaviors.

On the other hand, the majority of decentralized systems rely on such cooperation to make them efficient, and principally, viable, which fact is particularly important for applications that distribute content sensitive to time, as is the case of media streaming. Thus, incentive mechanisms for cooperation are necessary in this context, and have received particular attention in the literature, recently. Such mechanisms have been the subject of the study in this chapter.

Firstly, a study was made of the different types of uncooperative behavior common in decentralized systems, such as peer-to-peer and ad hoc networks and multi-agent systems. There was then a presentation of the existing incentive patterns, their characteristics and how they combat uncooperative behaviors, with a view to providing information for the understanding of a wider scope of existing incentive schemes, as well as future proposals as yet to come. Given that the number of related works is vast, including proposals for new schemes and studies of questions arising from the use of already-known schemes, a selection was made of these, then presented in the sequence.

One notes that a good incentive mechanism is one that effectively fights the common uncooperative behaviors, and also those induced by its use. On top of this, ideally, it exempts forgivable noncooperations from punishment, an issue that cannot always be regarded as trivial, as per the study made in Section 3.

However, no incentive mechanism is ideal for all types of contexts, that is to say, it is important to take into account the specific features of each type of application in designing the incentive scheme to be employed, that is, in choosing the incentive patterns to be used. Section 4 illustrated how different schemes, made up of separate patterns, may be pertinent to the most varied types of systems, such as live media streaming, media streaming on demand or file sharing. Lastly, one may mention that there is a trend towards combining different incentive patterns in the definition of new incentive schemes, so as to bring together the advantages and offset the shortcomings of each one [22, 23].

An interesting topic for future studies is the understanding of how effective incentive mechanisms are when subject to malicious behavior such as attacks. The conception of robust incentive mechanisms is still a challenging issue. Another topic to be considered for further investigation is the handling of new uncooperative behaviors induced by the employment of incentive schemes. For instance, how the impact of mobile users can change the effectiveness of incentive mechanisms. Other forms of interference include Sybil and white-washing. A discussion of these issues as well as the workarounds for them is quite useful. Furthermore, the conception of incentive mechanisms based on Game Theory can bring new perspectives to the treatment of uncooperative behavior.

References

1. Adar, E., Huberman, B.A.: Free riding on gnutella. First Monday **5**(10) (2000)
2. Bittorrent. http://www.bittorrent.com/ (2008). (Access Date)
3. Bittorrent tracker. http://en.wikipedia.org/wiki/BitTorrent_tracker (2008). (Access Date)
4. Buttyán, L., Hubaux, J.P.: Nuglets: a virtual currency to stimulate cooperation in self-organized ad hoc networks. Tech. Rep. DSC/2001, EPFL (2001). URL `citeseer.ist.psu.edu/buttyan01nuglets.html`
5. Buttyán, L., Hubaux, J.P.: Stimulating cooperation in self-organizing mobile ad hoc networks. ACM/Kluwer Mobile Networks and Applications (MONET) **8**(5), 579–592 (2003)
6. Castro, M., Druschel, P., Kermarrec, A.M., Nandi, A., Rowstron, A., Singh, A.: Splitstream: High-bandwidth multicast in a cooperative environment. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03), pp. 298–313. ACM Press (2003)
7. Chou, P.A., Wang, H.J., Padmanabhan, V.N.: Layered multiple description coding. In: Proceedings of the Packet Video Workshop (2003)
8. hua Chu, Y., Zhang, H.: Considering altruism in peer-to-peer internet streaming broadcast. In: Proceedings of the 14th international workshop on network and operating systems support for digital audio and video (NOSSDAV '04), pp. 10–15. ACM Press, New York, NY, USA (2004)

9. Cohen, B.: Incentives build robustness in bittorrent. In: Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems (2003)
10. Figueiredo, D.R., Garetto, M., Towsley, D.: Exploiting mobility in ad-hoc wireless networks with incentives. Tech. Rep. 04-66, UM-CS (2004)
11. Guo, Y., Suh, K., Kurose, J.F., Towsley, D.F.: P2cast: peer-to-peer patching scheme for vod service. In: Proceedings of the twelfth international conference on World Wide Web, pp. 301–309. ACM Press (2003)
12. Habib, A., Chuang, J.: Incentive mechanism for peer-to-peer media streaming. In: Proceedings of the 12th IEEE International Workshop on Quality of Service (IWQoS'04), pp. 171–180 (2004)
13. Habib, A., Chuang, J.: Service differentiated peer selection: An incentive mechanism for peer-to-peer media streaming. IEEE Transactions on Multimedia **8**(3), 610–621 (2006)
14. Habib, A., Chuang, J., Hefeeda, M.M.: Do we need incentive mechanisms for peer-to-peer media streaming. IEEE Communications, Submitted but not published (2003)
15. Kazaa. http://www.kazaa.com/ (2008). (Access Date)
16. Manzato, D.A.G., da Fonseca, N.L.S.: An incentive mechanism for peer-to-peer networks with media streaming. In: Proceedings of the 49th IEEE Global Telecommunication Conference (GLOBECOM 2006), pp. 1–6 (2006)
17. Manzato, D.A.G., da Fonseca, N.L.S.: Incentive mechanism for the coopnet network. Peer-to-Peer Networking and Applications (P2PNA) **1**(1), 29–44 (2008). DOI 10.1007/s12083-007-0004-0
18. Moreton, T., Twigg, A.: Trading in trust, tokens, and stamps. In: Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems (2003)
19. Ngan, T.W.J., Wallach, D.S., Druschel, P.: Incentives-compatible peer-to-peer multicast. In: Proceedings of the 2nd Workshop on Economics of Peer-to-Peer Systems (2004)
20. Obreiter, P., König-Ries, B., Klein, M.: Stimulating cooperative behavior of autonomous devices – an analysis of requirements and existing approaches. In: Proceedings of the Second International Workshop on Wireless Information Systems (WIS2003), pp. 71–82 (2003)
21. Obreiter, P., König-Ries, B., Klein, M.: Stimulating cooperative behavior of autonomous devices – an analysis of requirements and existing approaches. Tech. Rep. 2003-1, Universität Karlsruhe, Faculty of Informatics (2003)
22. Obreiter, P., Nimis, J.: A taxonomy of incentive patterns – the design space of incentives for cooperation. In: Proceedings of the Second International Workshop on Agents and Peer-to-Peer Computing (AP2PC'03), Springer LNCS 2872, pp. 89–100 (2003)
23. Obreiter, P., Nimis, J.: A taxonomy of incentive patterns – the design space of incentives for cooperation. Tech. Rep. 2003-9, Universität Karlsruhe, Faculty of Informatics (2003)
24. Padmanabhan, V.N., Sripanidkulchai, K.: The case for cooperative networking. In: Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS), pp. 178–190. Springer-Verlag (2002)
25. Padmanabhan, V.N., Wang, H.J., Chou, P.A.: Resilient peer-to-peer streaming. In: Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP), pp. 16–27. IEEE Computer Society (2003)
26. Padmanabhan, V.N., Wang, H.J., Chou, P.A.: Supporting heterogeneity and congestion control in peer-to-peer multicast streaming. In: Proceedings of the Third International Workshop on Peer-to-Peer Systems (IPTPS), pp. 54–63. Springer (2004)
27. Padmanabhan, V.N., Wang, H.J., Chou, P.A., Sripanidkulchai, K.: Distributing streaming media content using cooperative networking. In: Proceedings of the 12th international workshop on network and operating systems support for digital audio and video (NOSSDAV), pp. 177–186. ACM Press (2002)
28. Zhang, X., Liu, J., Li, B., Yum, T.S.P.: Donet/coolstreaming: A data-driven overlay network for live media streaming. In: IEEE INFOCOM, vol. 3, pp. 2102–2111 (2005)
29. Zhong, S., Chen, J., Yang, Y.R.: Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In: Proceedings of IEEE INFOCOM (2003)

Bandwidth Trading as Incentive

Kolja Eger and Ulrich Killat

Abstract In P2P networks with multi-source download the file of interest is fragmented into pieces and peers exchange pieces with each other although they did not finish the download of the complete file. Peers can adopt different strategies to trade upload for download bandwidth. These trading schemes should give peers an incentive to contribute bandwidth to the P2P network. This chapter studies different trading schemes analytically and by simulations. A mathematical framework for bandwidth trading is introduced and two distributed algorithms, which are denoted as Resource Pricing and Reciprocal Rate Control, are derived. The algorithms are compared to the tit-for-tat principle in BitTorrent. Nash Equilibria and results from simulations of static and dynamic networks are presented. Additionally, we discuss how trading schemes can be combined with a piece selection algorithm to increase the availability of a full copy of the file. The chapter closes with an extension of the mathematical model which takes also the underlying IP network into account. This results in a TCP variant optimised for P2P content distribution.

1 Introduction

P2P systems for file-sharing have to define functions for constructing and maintaining the overlay topology, to search and request for content and to transfer the content between peers. Some P2P applications like BitTorrent are optimised for the dissemination of the content. Hence, they are also denoted as P2P Content Distribution Networks (CDNs). To speed-up the dissemination of the file a multi-source

Kolja Eger
Siemens AG, Corporate Technology Information and Communications, Otto-Hahn-Ring 6, 81739 Munich, Germany, e-mail: kolja.eger@siemens.com

Ulrich Killat
Hamburg University of Technology (TUHH), Institute of Communication Networks, 21071 Hamburg, Germany, e-mail: killat@tuhh.de

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_23, © Springer Science+Business Media, LLC 2010

download (or swarming principle) is applied. Here, the file of interest is fragmented into pieces or chunks. When a peer completes the download of a single piece, it offers it to other peers which so far have not downloaded this piece. Thus, peers exchange pieces with each other although they did not finish the download of the complete file. Therefore, the resources in the P2P network are used more efficiently and the network depends not solely on the altruistic or cooperative behaviour of the peers. The swarming principle rather exploits the two-way interest of peers in different chunks, which the other one provides. Since most of the peers behave selfishly and are interested in maximising their own download rates, the mutual interest results in peers, which bargain for bandwidth with each other.

Peers can adopt different strategies to trade upload for download bandwidth. These trading schemes should give peers an incentive to contribute bandwidth to the P2P network. Hence, peers can maximise their own download rates by appropriately allocating their upload bandwidths and free-riding is decreased and/or avoided in the network. The design of a trading scheme can follow different objectives. From the user perspective, a peer might choose a strategy which outperforms all other strategies. Since the total capacity in the network is finite, the increase in download performance for this peer happens at the expense of others. On the other hand, a protocol designer might look for a strategy for which fairness is realised among peers although peers behave selfishly.

This chapter discusses different approaches to realise fairness in P2P content distribution networks. A mathematical framework for bandwidth trading is introduced. The model is based on congestion pricing (or network utility maximisation) which is used for the resource allocation in IP networks. The allocation of bandwidth is modelled as optimisation problem. Two distributed algorithms, which are denoted as Resource Pricing and Reciprocal Rate Control, are derived from this optimisation problem. Hence, their equilibrium points are optimal with respect to efficiency and fairness. In the first trading scheme, each peer receives a (virtual) payment for uploading data to others. These payments control the sending rates in the future. Peers with higher price offers will receive higher rates whereas the rate will decrease for peers with lower price offers. Furthermore, the total payment a peer can afford is limited to its own upload capacity. In the second trading scheme peers use their download rates from other peers as the payment information. Thus, no explicit pricing must be implemented in the protocol.

The two proposed trading schemes are compared to BitTorrent's tit-for-tat principle. Here, a peer uploads to others from which it receives the highest download rates. We discuss the existence of a Nash Equilibrium for each trading scheme and present results from simulations of static as well as dynamic networks.

A trading scheme is only one of the building blocks of a P2P protocol. Since every peer starts with no piece at the beginning a good implementation has to ensure that new peers have a decent chance to finish their first piece fast. Furthermore, the selection of pieces to download is important for the future exchange with other peers. Besides finding the peers which provide good download rates, a peer should download rare pieces to ensure that it has something of interest for others in the future. Hence, we discuss additionally how trading schemes can be combined with a piece selection algorithm. Thus, the availability of a full copy of the file in the network can be increased.

Bandwidth trading is affected by the performance in the underlying network, e.g. by packet loss. Based on the mathematical framework for P2P we discuss also a rate control algorithm for P2P over IP networks. We denote the approach as *TCPeer*, because a peer adopts the functionality of TCP and extends it with information from the overlay network.

2 Trading Schemes for P2P Content Distribution

Trading schemes give peers an incentive to contribute bandwidth to the P2P content distribution network. Hence, peers can maximise their own download rates by appropriately allocating their upload bandwidths and free-riding is decreased and/or avoided in the network.

An example is the game theoretical approach in BitTorrent. We compare it with two other schemes that are based on the resource pricing (or congestion pricing) approach [14, 15] for transport-layer protocols. Both proposed pricing algorithms use rate control at the application layer. In a real implementation this can be realised by the application putting different amounts of data into the socket buffers of the TCP connections. A more sophisticated approach is to schedule the traffic directly at the queue of the network interface card, e.g. with Linux Traffic Control [13]. Furthermore, a combined approach of rate control at the application and the transport layer is discussed in Section 6.

In the following section a general P2P network model is introduced, from which the two proposed pricing algorithms are derived in Sections 2.2 and 2.3. For comparison, the tit-for-tat principle used in BitTorrent is described in Section 2.4.

2.1 P2P Network Model

Similar to the model for IP networks in [14, 15] we model the resource allocation in P2P networks as optimisation problem. Consider a P2P network consisting of a set of peers $\mathcal{P}$ and a set of services, whereby each peer $p \in \mathcal{P}$ is interested in one or several services and/or offers different services. Providing a service consumes resources. In this work we concentrate on resources (e.g. access bandwidth) which are divisible and where any allocation of resources has a benefit for a requesting peer. We assume that the resource is scarce such that competition is present and denote the capacity of this resource at peer p as C_p.

To differentiate between service providing and service requesting peers in our mathematical model we introduce the set of service providers or servers $\mathcal{S}$ and the set of service customers or clients $\mathcal{C}$. The terms server and client are also used for P2P networks. In this context, a server is a peer which offers at least one service and a client is a peer which requests at least one service. A peer can be a server and a client at the same time.

Since each peer has only a partial view of the whole P2P network, a service requesting peer is not aware of all peers that provide this service, and vice versa. We define the set of peers which offer at least one service to the client c as the set of servers $\mathcal{S}(c)$ of client c. The other way round $\mathcal{C}(s)$ is the set of the known clients of the server s. Furthermore, if $c \in \mathcal{C}(s)$ then also $s \in \mathcal{S}(c)$ holds. The client-server architecture can be interpreted as a special case of a P2P application where the set $\mathcal{S}(c)$ consists of one single server.

Suppose the utility of a client c is defined by a utility function U_c, which depends on the total service rate y_c. Furthermore, y_c is the total of the rates x_{sc} of all servers s of client c. The optimisation problem for P2P networks is

$$\begin{aligned}
&\text{P2P SYSTEM:} \\
&\qquad \text{maximise} \quad \sum_{c\in\mathcal{C}} U_c(y_c) && (1)\\
&\qquad \text{subject to} \quad \sum_{s\in\mathcal{S}(c)} x_{sc} = y_c, \quad \forall c \in \mathcal{C} && (2)\\
&\qquad \qquad \sum_{c\in\mathcal{C}(s)} x_{sc} \le C_s, \quad \forall s \in \mathcal{S} && (3)\\
&\qquad \text{over} \quad x_{sc} \ge 0. && (4)
\end{aligned}$$

Maximising the aggregated utility of the service rate y_c over all clients is the objective of the whole system, where y_c is the sum of the rates x_{sc} of all servers s of c. This problem is constrained by the resource capacity at the servers.

To ensure some kind of fairness among peers, the utility function in (1) is set adequately. In general, different fairness criteria deviate from each other, because they trade-off between fairness and efficiency differently. The concept of proportional fairness [15] makes a widely accepted compromise between fairness and efficiency and is used frequently in congestion pricing algorithms. In contrast to max-min fairness, which gives absolute priority to small flows, proportional fairness depends on the proportional changes of two rate allocations. Hence, smaller flows are treated less favourably. The concept can be extended to weighted proportional fairness. Here, a client c has a willingness-to-pay W_c that weighs the value of the client. As already well studied for IP networks [14], we set the utility function of all peers to

$$U_c\left(y_c\right) = W_c \ln\left(y_c\right). \qquad (5)$$

This utility function ensures a weighted proportional fair resource allocation. Furthermore, the function in (5) is concave and strictly increasing. Hence, the optimisation problem in (1)–(4) has a unique optimum with respect to the total service rate y_c. With a logarithmic utility function we assume the law of diminishing returns applies for peers. This means that the performance enhancement decreases with increasing total service rate. Such a utility function represents users with elastic traffic demands [20] and is applicable when users have no delay requirements. It is used often for data transfers [14]. Hence, it is relevant for P2P file-sharing.

As shown in [8] for a connected P2P network, the optimum of (1)–(4) with respect to the total download rate y_c and the offered price λ_c is

$$y^{\cdot}_c = \sum_{s\in\mathcal{S}(c)} x^{\cdot}_{sc} = \frac{W_c \sum_{s\in\mathcal{S}} C_s}{\sum_{d\in\mathcal{C}} W_d} \tag{6}$$

and

$$\lambda^{\cdot}_c = \frac{\sum_{d\in\mathcal{C}} W_d}{\sum_{s\in\mathcal{S}} C_s}, \tag{7}$$

respectively.

2.2 Resource Pricing

To derive a scalable, distributed algorithm, where each entity works with limited information about the network conditions and users in the network, we divide the global optimisation problem in (1)–(4) into sub-problems. These are derived with the Lagrangian

$$L^{\mathrm{P2P}}(\mathbf{x},\mathbf{y},\lambda,\nu,\mathbf{n}) = \\ \sum_{c\in\mathcal{C}}\left(U_c(y_c) - \lambda_c\left(y_c - \sum_{s\in\mathcal{S}(c)} x_{sc}\right)\right) + \sum_{s\in\mathcal{S}} \nu_s\left(C_s - \sum_{c\in\mathcal{C}(s)} x_{sc} - n_s\right), \tag{8}$$

where $n_s \geq 0$ is a slack variable for the inequality constraint in (3) for server s and λ_c and ν_s are Lagrange multipliers. Like in [15] Lagrange multipliers are interpreted as prices for the usage of resources. Hence, we say λ_c and ν_s are prices per unit offered by the client c and charged by the server s, respectively. By looking at the Lagrangian in (8) we see that the problem is separable into the sub-problems for each client and server

CLIENT c :

$$\text{maximise } U_c(y_c) - \lambda_c y_c \tag{9}$$

$$\text{over} \quad y_c \geq 0 \tag{10}$$

SERVER s :

$$\text{maximise} \sum_{c\in\mathcal{C}(s)} \lambda_c x_{sc} + \nu_s\left(C_s - \sum_{c\in\mathcal{C}(s)} x_{sc} - n_s\right) \tag{11}$$

$$\text{over} \quad x_{sc} \geq 0. \tag{12}$$

The rate x_{sc} is set by the server s and also the capacity constraint is related to the servers. Hence, we propose a primal algorithm that adapts the rate x_{sc} at server s. A simple method to optimise (11) is by iterative descent algorithms (see Section 3.2 of [2]). The partial derivative of (11) with respect to x_{sc} depends on the Lagrange multipliers λ_c and ν_s. The offered price λ_c can be derived from the client problem in (9). However, the charged price ν_s can not be derived. Thus, we estimate ν_s by the average offered price per unit capacity. Hence, the proposed distributed algorithm is

$$x_{sc}(t+1) = \max\left(\varepsilon, x_{sc}(t) + \gamma x_{sc}(t)\left(\lambda_c(t) - \frac{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\,\lambda_d(t)}{C_s}\right)\right). \tag{13}$$

Here, γ is a positive step size and ε is a small positive constant. Hence, the sending rate x_{sc} does not fall below ε. This models a kind of probing, which is necessary to receive information about the offered price of the client. The offered price λ_c is computed by setting the derivative with respect to y_c of the CLIENT problem in (9) to zero. If we use the logarithmic utility function in (5)

$$\lambda_c(t) = \frac{W_c}{\max(\eta, y_c(t))} = \frac{W_c}{\max\left(\eta, \sum_{s\in\mathcal{S}(c)} x_{sc}(t)\right)}. \tag{14}$$

Here, a small positive constant η is introduced to ensure a bounded price offer if the total service rate is zero.

The algorithm in (13)–(14) is denoted as Resource Pricing in the following. It is interpreted from an economical perspective as follows: Since λ_c is the price per unit bandwidth, the product $x_{sc}\lambda_c$ is the total price which is paid by peer c. Hence, the sum over all clients in the numerator of (13) represents the total revenue of server s. Dividing the total revenue by C_s returns the average price server s obtains per unit capacity. If the offered price by client c is higher than the average price, then server s increases its upload rate to c and decreases it otherwise. Additionally, (13) scales the price difference by the rate x_{sc}. This model reflects a market for service capacity where prices control the rate allocations. Since the service capacity is a non-storable commodity the market is a spot market, where the current demand influences the future prices.

In [6] we show that the algorithm in (13)–(14) fulfils fundamental properties for the resource allocation in a distributed environment. We derive conditions for the efficiency of the algorithm and prove that the equilibrium point of (13)–(14) coincides with the optimum of the global optimisation problem in (1)–(4). Furthermore, global asymptotic stability of the equilibrium point is proven for a continuous model without delays as well as a discrete model with delays.

The resource pricing algorithm can be used in P2P content distribution networks, where the willingness-to-pay is set to the upload bandwidth of a peer. Hence, $W_q = C_q$ for peer q and the total download rate in steady-state in (7) becomes

$$y^{\cdot}_q = \frac{C_q \sum_{p\in\mathcal{P}} C_p}{\sum_{c\in\mathcal{C}} C_c}, \tag{15}$$

where $\mathcal{P}$ is the set of peers in the network and $\mathcal{C}$ is the set of peers which currently are downloading the file. This model gives peers an incentive to contribute to the network since a peer with a high upload capacity can offer higher prices than others and therefore receives a higher download rate.

2.3 Reciprocal Rate Control

The resource pricing algorithm assumes the correct signalling of the price offers. Thus, an implementation must ensure that malicious users cannot forge their price offers to obtain higher download rates. When this is not possible in a specific application, an upload rate control algorithm can only be based on the download rates of the connected peers (like in BitTorrent). This means a peer is only willing to provide upload bandwidth to another peer when it receives in return bandwidth from it.

In the context of pricing mechanisms the download rates can be interpreted as the price a remote peer is willing to pay for the upload bandwidth allocated to it. In the resource pricing algorithm the total price paid to peer p by a remote peer q for the allocation of the rate x_{pq} is the product $x_{pq}\lambda_q$. Hence, by replacing $x_{pq}\lambda_q$ with the rate x_{qp} in (13) we get

$$x_{pq}(t+1) = \max\left(0, x_{pq}(t) + \gamma\left(x_{qp}(t) - x_{pq}(t)\frac{\sum_{r\in\mathcal{N}(p)} x_{rp}(t)}{C_p}\right)\right). \qquad (16)$$

Here, the set $\mathcal{N}(p)$ denotes the set of neighbours of peer p. Since it uses the rate as implicit price we denote it as Reciprocal Rate Control (RRC). We show in [10] that the equilibrium point $\mathbf{x}^{\cdot}$ of the distributed algorithm in (16) finds an efficient and weighted fair resource allocation, if it is feasible. A peer uses its full upload bandwidth and downloads from a specific peer with the same rate as it is uploading, i.e.

$$x^{\cdot}_{qp} = x^{\cdot}_{pq}, \quad \forall q \in \mathcal{N}(p). \qquad (17)$$

Furthermore, the total download rate over all connections is equal to the upload capacity of the peer

$$y^{\cdot}_p = \sum_{q\in\mathcal{N}(p)} x^{\cdot}_{qp} = C_p. \qquad (18)$$

The reciprocal rate control algorithm is restricted to peers that did not complete the download already. If feasible, it ensures that a peer receives at least the rate which it provides to the network. Seeds cannot run the algorithm and have to use other rules for their upload. (E.g. these peers can upload to peers, which have nothing at all.) Therefore, an advantage of explicit prices used by resource pricing as compared to implicit prices is that fairness is preserved also over the altruistically contributed resources in the network.

2.4 BitTorrent

Also in BitTorrent each peer controls to whom it uploads data. This peer selection is called *choking/unchoking*. When a remote peer is selected for upload an *UNCHOKE* message is sent. An upload is stopped with a *CHOKE* message. Each peer uploads to a fixed number of other peers (the default value is four) and chooses to upload to others from which it gets the highest download rates. This principle is called tit-for-tat, because it is based on reciprocity. Unchoking by download rates is not applicable for seeds. Therefore, the peer selection of the seeds is based on the upload rates to the connected peers. This is driven by the idea that high upload rates are only achieved when no one else uploads to the peer. Hence, those peers should be preferred which are not served by others.

By default this tit-for-tat strategy is run every ten seconds by every peer, whereby the download rates are determined by a moving average over the last 20 seconds. To discover new peers with better performance a so-called optimistic unchoke is done additionally. Here, one of the peers is unchoked independently of its rate. The optimistic unchoke is changed every 30 seconds to provide enough time to be possibly unchoked by the remote peer in return. Another rule in BitTorrent is to choke a peer when it has sent no data message in the last minute. This is called anti-snubbing.

The unchoking algorithm of BitTorrent gives peers an incentive to upload data to the network since it increases the chance to be unchoked by others with increasing upload capacity. Analytical models of BitTorrent's incentive mechanism are presented in [19, 21]. In their simplified models global information is assumed, i.e. the peers know the upload bandwidth of all other peers. In this case an optimistic unchoke is needless and the peer selection achieves an equilibrium [21]. Peers form groups with a group size equal to the number of unchokes plus one. Furthermore, groups consist of peers with similar upload bandwidth depending on the descending order of the bandwidth (see [21] for details). Hence, the download rate of a peer depends not only on the upload bandwidth but also on its ratio to the bandwidth of other peers in the network. Thus, peers with the same upload bandwidth can be members in different groups and achieve different download rates. Further on, the peers with the lowest upload bandwidths form a group which can be smaller than the other groups.

3 Nash Equilibrium

Most peers in a P2P network behave selfishly and try to maximise their performance only. Therefore, we can interpret the trading of bandwidth as a non-cooperative game and study the Nash equilibrium. In the Nash equilibrium no peer gains by only changing its strategy while all other players keep their strategy unchanged. This means a Nash equilibrium corresponds to a set of strategies where every player has found the best response to the actions of the other players [18].

The Nash equilibrium for the BitTorrent protocol is studied in [19]. Here, each peer can choose its upload bandwidth between zero and its physical upload capacity. Since a higher upload bandwidth is associated with a higher cost, a peer is interested primarily in maximising its download rate while also minimising its cost at the same time. [19] shows that a Nash equilibrium exists for networks consisting of groups (with a group size larger than the number of unchokes plus one) of peers where peers belonging to the same group have identical upload bandwidths. In this situation the best case for a peer is to upload to other peers in its group. The threshold to be unchoked is exactly the upload bandwidth divided by the number of unchokes (which is assumed to be equal for all peers). Therefore, a peer will be choked by the peers of its group when it reduces its upload bandwidth. This results in inferior performance.

On the other hand a Nash equilibrium does not exist when peers have different upload capacities as it is demonstrated by an example in [19]. In this case some peers can decrease their upload bandwidths and are still unchoked by the same remote peers. Additionally, these peers can increase the number of peers to unchoke and are possibly unchoked by further peers in return. Hence, the strategies of peers are diverse and not only restricted by choosing an upload bandwidth. This complicates the discussion of the Nash equilibrium for BitTorrent.

In the following we study the Nash equilibria for the two proposed pricing schemes in steady-state. Assume for resource pricing that a peer cannot forge its price offer λ. It can be seen directly from (15) that any reduction of upload capacity will result in a decrease of the download rate. When maximising the download rate has priority over minimising the upload cost every peer will upload with its physical upload capacity. Thus, the steady-state of the system (13)–(14) is the Nash equilibrium.

Similarly, the existence of the Nash equilibrium for reciprocal rate control can be shown for the steady-state with (17) and (18). Since a peer receives exactly the rate which it provides to the other peer, it gains nothing by reducing its upload bandwidth. Hence, in steady-state its total download rate is its own upload capacity.

4 Performance Evaluation

The discussed trading schemes for P2P content distribution are evaluated by simulations in the following. We restrict our analysis to the application layer. It is widely assumed that the bottlenecks in P2P networks are the access links of the users [1, 19]. This seems reasonable because most peers are home users who are connected e.g. with DSL or cable modems to the Internet. Furthermore, the core of the Internet shows a low utilisation of the available bandwidths [17] and small packet loss rates due to congestion [11].

Additionally, we omit the TCP behaviour in this section and discuss briefly the interaction between P2P and the transport protocol in Section 6. Furthermore, we present in [7] a simulation analysis of the differences between packet-level and flow-level simulations for BitTorrent. Since home users often have asymmetric access links (e.g. ADSL) and/or set the upload bandwidth in the P2P application lower than their physical upload capacity, we assume the uplink is the bottleneck in the whole network.

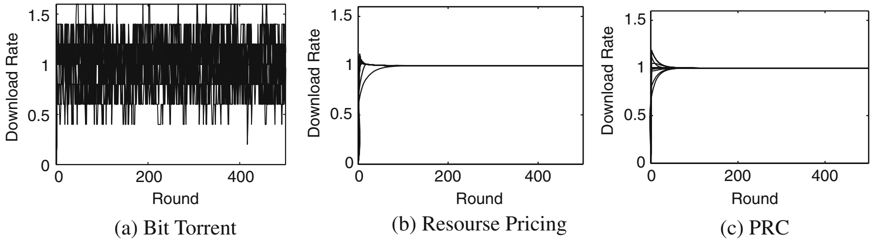
The bandwidth trading schemes depend on the number of neighbours and especially on the number of connections where both sides are interested in data of the other side. Thus, the topology generated by the protocol and the implemented piece selection algorithm influence the results of bandwidth trading. To concentrate on the trading schemes in the simulations we omit the piece selection algorithm for now and assume connected peers are always interested in pieces of each other. Furthermore, for BitTorrent anti-snubbing is neglected, because it can result in situations where a peer does not contribute its upload bandwidth although it can transfer data to other peers. This can cause inefficiency in the network.

The overlay topology is constructed in the simulations according to the original BitTorrent implementation [3, 4]. Here, a peer asks a so-called tracker, a centralised component that stores information about all peers, for a list of other peers in the network. The tracker returns a random subset of length N_R to the requesting peer. Hence, a peer opens a new connection to a remote peer based on the list returned by the tracker or when a remote peer asks for it.

We will present results for static and dynamic P2P networks. Further simulation results can be found in [10].

4.1 Static networks

One major difference of the two pricing schemes to BitTorrent's tit-for-tat strategy is their convergence to a steady-state. To demonstrate the basic functionality of all algorithms we run a simulation for a small network of 10 peers with homogeneous upload capacities of $C = 1$. The superposition of the download rates of the peers is depicted in Fig. 1. Whereas the download rates of the peers oscillate between 0.4 and 1.6 for BitTorrent, the rates converge fast to a fair allocation for the Resource Pricing (RP) and the Reciprocal Rate Control (RRC) algorithm where each peer receives exactly its upload capacity.



(a) Bit Torrent (b) Resourse Pricing (c) PRC

Fig. 1 Superposition of the download rates of 10 peers for different trading schemes

The oscillations with BitTorrent are caused by the optimistic unchoke. For the ideal case where peers have global information about the upload bandwidth of other peers, analytical studies [19, 21] show that the tit-for-tat strategy has an equilibrium point. Since the optimistic unchoke is a random process, the mean values of the download rates are closer together. In this example the mean download rates of the 10 peers lie between 0.77 and 1.23.

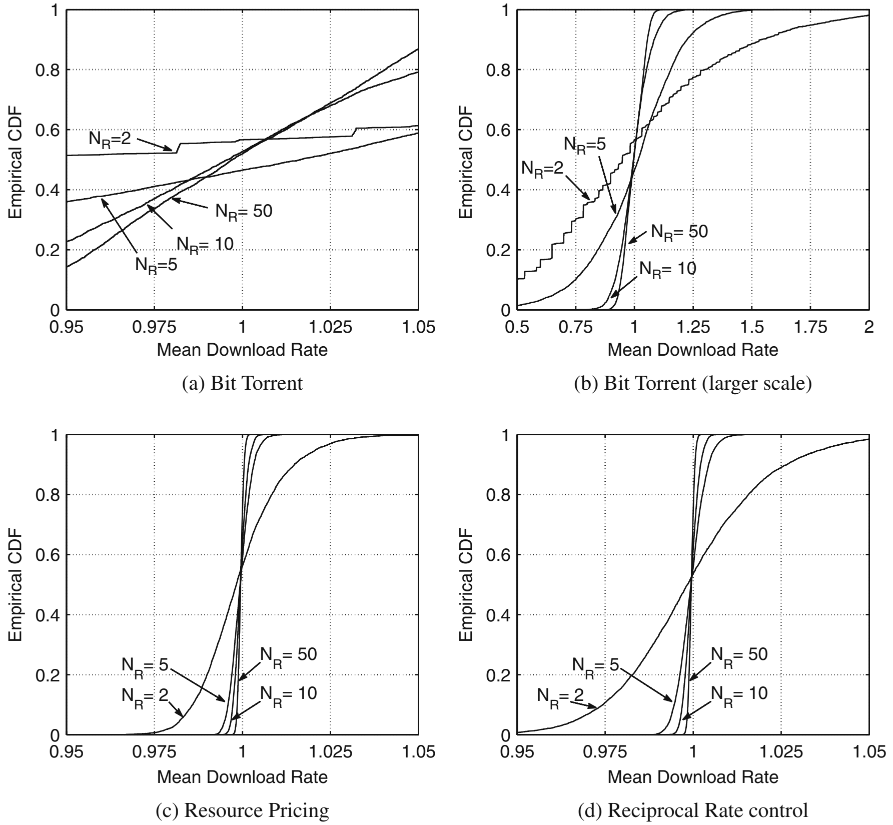
Especially, the performance of BitTorrent depends on the number of neighbours. To study the influence of the topology on the download performance simulations are run with varying parameter N_R. The network population was 1000 peers, each peer starting with a small random offset to avoid synchronisation effects. The simulation consisted of 1000 iterations of the respective algorithm and was repeated 5 times. We set $\gamma = 0.1$ for RP and RRC and $\varepsilon = 0.1C/N_C$, whereby N_C denotes the number of connections of a peer. The capacity of all peers is $C = 1$.

The empirical cumulative distribution function (CDF) of the mean download rate over the 1000 iterations is depicted over all runs in Fig. 2. To compare the three algorithms Fig. 2a, c, d, show the results for download rates between 0.95 and 1.05. With BitTorrent's tit-for-tat algorithm peers receive highly different download rates while providing the same upload capacity. To see its performance on a larger scale the CDF for download rates between 0.5 to 2 is depicted in Fig. 2b.

Tit-for-tat depends on the number of neighbours. With $N_R = 2$ a peer has on average four connections only [10] and therefore no choice in selecting peers. This results in a poor performance where 10% of the peers receive half or less of what they provide. Download rates are distributed more equitably with increasing N_R. For $N_R = 10$ all peers receive a mean download rate of $\pm 20\%$ of their upload capacity, whereas this percentage decreases to $\pm 10\%$ for $N_R = 50$.

Nonetheless, the distribution of download rates with BitTorrent is not as uniformly distributed as with the two proposed pricing algorithms. Nearly all peers receive download rates between 0.98 and 1.02 with both algorithms and values of $N_R > 2$. Furthermore, the download rates at the end of each simulation were equal to one for every peer. This means the pricing algorithms converged. Thus, the differences of the CDFs for different values of N_R in Fig. 2c, d are due to differences in the rate of convergence.

For the next simulation we consider heterogeneous peers with upload capacities of integers between 1 and 10. The capacity is determined randomly at the start of the simulation. With a total of 1000 peers there are on average 100 peers with the same capacity in the network. The download performance of each peer is measured by the ratio of mean download rate to upload capacity. Figure 3 shows the median and the 2.5 to 97.5 percentile range of the mean download rates for peers with the same upload capacity. By using BitTorrent the median for peers with an upload capacity of $C = 1$ is much larger as the fair share of one whereas the peers with $C = 10$ receive mostly a rate lower than their contribution. One reason for this is the optimistic unchoke since a low capacity peer gains more than it invests compared to a high capacity peer. Another reason is the incapability to find the right



(a) Bit Torrent

(b) Bit Torrent (larger scale)

(c) Resource Pricing

(d) Reciprocal Rate control

Fig. 2 Empirical CDF for varying parameter N_R

match, which also explains the different performances between peers with the same capacity.

For resource pricing and reciprocal rate control the median of the download performance is very close to the upload capacities of the peers. Additionally, the small percentile range indicates that peers with the same upload capacity receive nearly the same download performance.

4.2 Dynamic Networks

We conducted simulations also for dynamic networks where interarrival times between peers and session times of peers are negative exponentially distributed random variables. We set $N_R = 10$ and compare the performance for two cases with different interarrival times but for the same average peer population of 1000. We assume the respective algorithm is run per unit time. The resulting CDFs are depicted

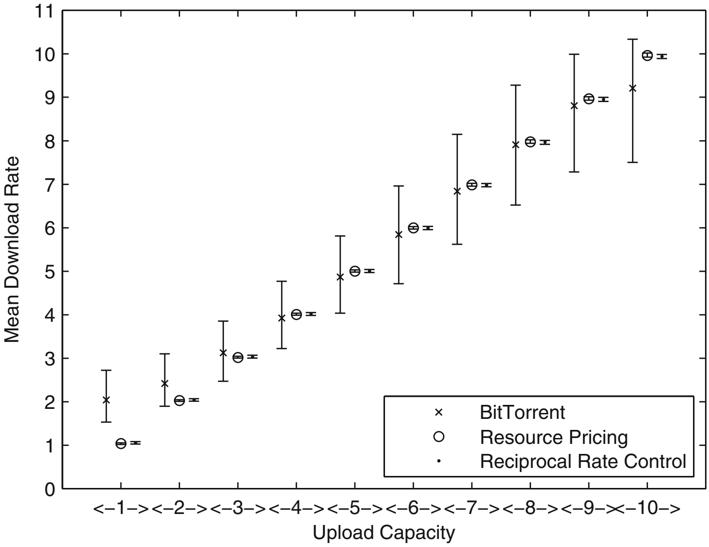
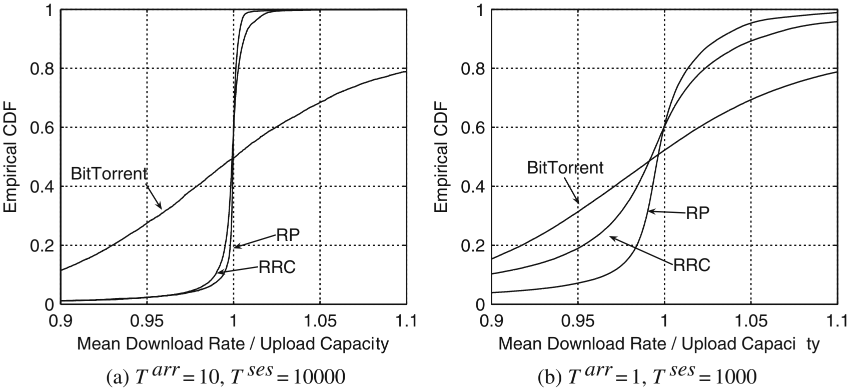


Fig. 3 Median and the 2.5–97.5 percentile range of the mean download rates for heterogeneous peers ($N_R = 10$)



(a) $T^{arr} = 10$, $T^{ses} = 10000$

(b) $T^{arr} = 1$, $T^{ses} = 1000$

Fig. 4 CDFs for dynamic networks ($N_R = 10$)

in Fig. 4a, b. With a mean interarrival time between peers of $T^{arr} = 10$, i.e. on average 10 iterations of the algorithm before a new peer enters, the two pricing schemes achieve nearly uniformly distributed download performances for heterogeneous peers.

For $T^{arr} = 1$ in Fig. 4b the CDF shows higher variability in the achieved download performance for the two proposed algorithms. Furthermore, the results for reciprocal rate control deteriorate more than for resource pricing comparing it with the results in Fig. 4a.

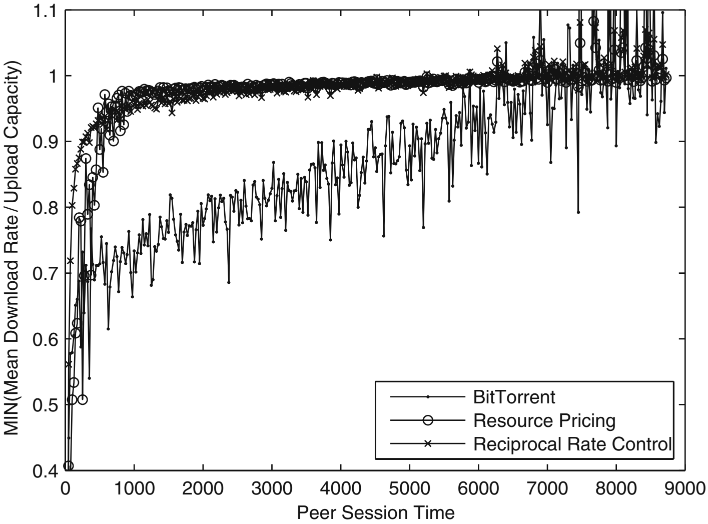


Fig. 5 Minimal download performance ($T^{arr} = 1, T^{ses} = 1000$)

Especially for $T^{arr} = 1, T^{ses} = 1000$ some peers stay only a very short time in the network and merely run the trading algorithm a few times. This results in a poor download performance as it can be seen from Fig. 5. Here, the minima of the download performances of peers with similar session times are measured for time intervals of 25 time units. For all schemes the minimal download performance increases with the session time of the peers, but the increase is more pronounced for the two trading schemes. After around 500 time units the minimal download performance in the network is above 0.9.

In conclusion, BitTorrent's tit-for-tat strategy gives peers an incentive to contribute upload bandwidth to the network. But it cannot avoid unfairness among peers with respect to the experienced download performance. The two proposed alternatives improve fairness. In static and dynamic simulations both algorithms show good convergence and better fairness as the tit-for-tat strategy used in BitTorrent.

5 Bandwidth Trading and Piece Selection

In Section 4 we assume peers have always parts of the file which are of interest for the remote peers. This simplification enables to study the incentive schemes without discussing a specific piece selection algorithm. It is also applicable to systems with network coding [12] where linear combinations of all pieces are exchanged and no piece selection is run. When network coding is not used, piece and peer selection depend on each other. Besides finding the peers which provide good download rates, a peer should download rare pieces to ensure that it has something of interest for others in the future. Furthermore, requesting rare pieces increases the availability of a full copy of the file in the network.

The optimisation problem in (1)–(4) can be extended to a multi-objective optimisation problem, where beside rates also the content availability is maximised. An optimisation problem for the content availability for BitTorrent-like networks is presented in [5]. However, the authors of [5] prove that the optimisation problem is NP-complete. Hence, no algorithm finds the optimum of the problem in polynomial time unless the well-known $P = NP$ problem in complexity theory is resolved.

On the other hand analytical and simulation-based studies [1, 19] show that BitTorrent's piece selection is efficient. Piece selection in BitTorrent follows the following rules [3]: Firstly, when some bytes are received from a specific chunk the remaining parts of that chunk are requested. This scheme is called strict priority. Since peers forward only complete chunks (where data integrity is verified) to other peers, this mechanism ensures that chunks are completed fast. When strict priority is not applicable, the rarest chunk is requested. Since a peer has only a local view of the network it can only estimate rarity based on the information of its neighbours. This information is available to the peer by the BitTorrent messages *BITFIELD* and *HAVE*. When a peer has no chunk at the beginning of the download, BitTorrent deviates from the rarest-first policy and the new peer requests a piece randomly. This is intended to ensure a faster completion of the first piece such that the upload bandwidth of that peer can be used by others.

We discuss in the following how the resource pricing algorithm in (13)–(14) can be combined with BitTorrent's piece selection. We choose resource pricing instead of reciprocal rate control, because it has two advantages. It ensures fairness also for the resources contributed by the seeds and has higher flexibility by defining the willingness-to-pay appropriately. For example, when the content distribution is realised by proprietary software of the content providers, they may set the willingness-to-pay for each peer according to the subscription status of the user. Furthermore, when the content owner contributes also resources to the P2P network, the owner has influence how these are allocated to the peers. Additionally, proprietary software (e.g. for set-top boxes) and the combination of P2P content distribution with IP multimedia subsystems as discussed in [16] have the potential to reduce the influence of malicious peers.

Assume we replace BitTorrent's peer selection by resource pricing but keep the other functionalities of BitTorrent. This means a peer controls the upload rate to all its neighbours with (13)–(14) and peers request pieces according to the rules in the BitTorrent specification [3].[1] We evaluate this new approach and compare it to the original BitTorrent implementation by simulations at the application layer. In the first simulation scenario a constant number P of peers is present in the network. Further on, S and L denote the number of seeds and leechers in the network, respectively.[2] Hence, $P = S + L$. To keep the number of seeds and leechers constant in our simulation we assume a leecher leaves the network when its download is completed

[1] For BitTorrent as well as the revised version with resource pricing the endgame-mode is neglected. We omitted the endgame mode in our implementation because it is not clearly specified when a peer switches to the endgame mode. Hence, different implementations realise it differently.

[2] Peers that have a full copy of the file and peers that are still downloading the file are denoted in the BitTorrent specification [3] as seeds and leechers, respectively.

and a new leecher with no pieces enters the network at this point in time. At the beginning of the simulation the leechers have a random set of pieces. This simulation setup studies the performance of peers with different download progress. Furthermore, it can be easily studied analytically. The total capacity in the network is $\sum_{s=1}^{S} C_s + \sum_{l=1}^{L} C_l$, where C denotes the upload capacity of a peer. A proportional relationship between upload capacity and download rate is desirable, because it gives peers an incentive to contribute bandwidth to the network. In the optimal case the download rate achieved by trading bandwidth with other leechers is equal to the peer's own upload capacity. Another portion of the download rate is contributed by the seeds in the network. If the resources of the seeds are allocated proportionally fair with respect to the leecher's upload bandwidth a leecher l has a total download rate of

$$y_l = C_l + C_l \frac{\sum_{s \in S} C_s}{\sum_{m \in L} C_m} = C_l \frac{\sum_{p \in P} C_p}{\sum_{m \in L} C_m}. \tag{19}$$

Equation (19) overestimates the download performance of a peer because it neglects that a peer needs a piece which is of interest for other peers. This is not always the case. Particularly, at the start a peer has no data and cannot contribute its resources to the network. Furthermore, (19) is identical to the steady-state download rate of resource pricing in (15) because resource pricing is designed to ensure fairness.

As in the previous section the number of neighbours has an influence on the download performance. In BitTorrent several parameters control the number of neighbours. So far we have taken only N_R, the number of peers returned by the tracker, into account. Now, we will introduce also the parameters *max initiate* and *min peers*, which are used in BitTorrent [3]. Denote *max initiate* with N_{MI} and define it as the number of connections at which the peer stops initiating new connections. Furthermore, denote *min peers* with N_{MP} and define it as the lower limit of connections at which a peer does not re-request new peer information from the tracker. By default, a BitTorrent peer requests $N_R = 50$ addresses of peers at the tracker and opens if possible up to $N_{MI} = 40$ connections to other peers. Due to churn the number of connections changes over time. It can increase above $N_{MI} = 40$ connections when remote peers ask for a connection because remote requests are not denied in the implementation. When many peers leave the network and the number of connections is N_{MP} or below the peer concerned asks the tracker for additional peer information. To avoid frequent re-requests a peer has to wait at least 300 s between two requests to the tracker.

In the following we will not present a detailed parameter study but use the default values for BitTorrent. For resource pricing we choose the parameters such that peers in the network have approximately the same number of neighbours and do not run out of neighbours. Hence, N_R is set to a large value and $N_{MI} = N_{MP}$. Additionally, we run simulations with the default parameters of BitTorrent also for resource pricing.

We simulate a network of $P = 100$ peers, which download a file of size $S_F = 100\,\text{MB}$. Pieces have a size of 256 KB. With BitTorrent each peer unchokes 4 other peers plus the optimistic unchoke. Additionally, all other parameters are set to the default values of BitTorrent [3]. For $S = 1$ and $L = 99$ and a homogeneous upload

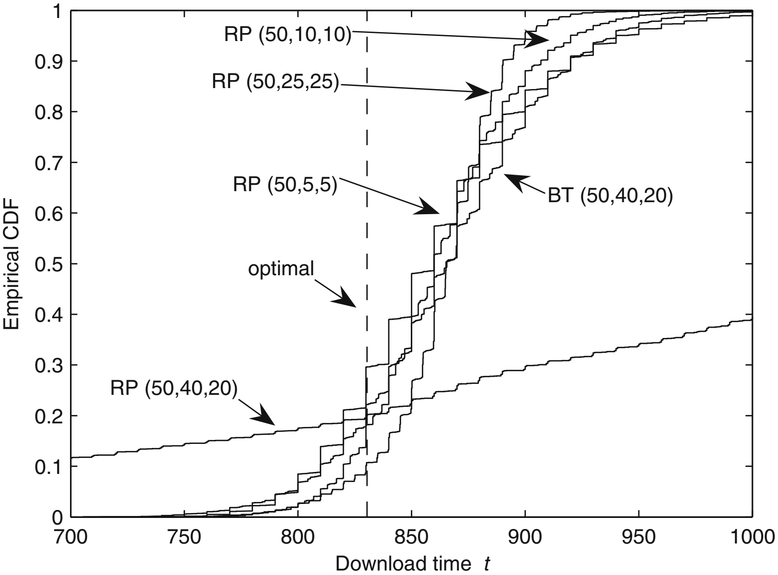


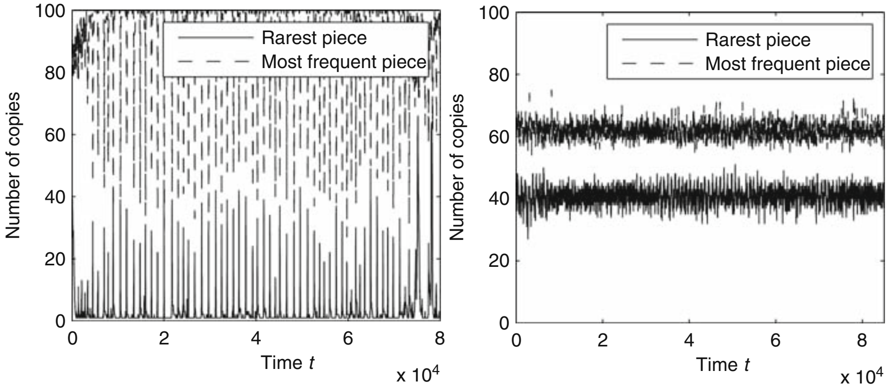
Fig. 6 CDF of the download time for Resource Pricing (RP) and BitTorrent (BT) with different parameters (N_R,N_{MI},N_{MP})

capacity of $C = 1$ Mbps the optimal download time t is $t = S_F/y_l \approx 830$ s using (19). Figure 6 compares the optimal download time with BitTorrent using default parameters and resource pricing with varying parameters N_{MI} and N_{MP}.

The most striking result is the poor performance with resource pricing and parameters $N_R = 50$, $N_{MI} = 40$ and $N_{MP} = 20$, which will be discussed in detail shortly. The other simulation results are close to the optimal value. The mean download time with BitTorrent is 866 s and for resource pricing with $N_{MI} = N_{MP} = 5$, $N_{MI} = N_{MP} = 10$ and $N_{MI} = N_{MP} = 25$ around 861 s, 863 s and 864 s, respectively. Looking at the distribution, 90% of the peers have a download time between 800-940 s, 810-920 s and 815-900 s with resource pricing and $N_{MI} = N_{MP} = 5$, $N_{MI} = N_{MP} = 10$ and $N_{MI} = N_{MP} = 25$, respectively. BitTorrent shows a range of 796-940 s. Hence, fairness among peers increases for resource pricing with an increasing value for the parameters N_{MI} and N_{MP} up to a limit of 25. Although differences are small for homogeneous peers fairness is better for these cases as with BitTorrent.

As opposed to BitTorrent, where peers upload by default to five other peers, resource pricing uploads data over all the connections of a peer. This can cause inefficiency as seen in Fig. 6 for $N_R = 50$, $N_{MI} = 40$ and $N_{MP} = 20$. By requesting data from a large number of other peers the piece selection is not efficient and rare pieces emerge. This can be seen from Fig. 7, where the number of copies for the rarest and the most frequent piece is depicted for the inefficient case ($N_R = 50, N_{MI} = 40, N_{MP} = 20$) in comparison to an efficient case ($N_R = 50, N_{MI} = 10, N_{MP} = 10$).

In Fig. 7a the rarest piece is sometimes only available at the seed. Although the number of copies increases fast in the following another piece becomes rare.



(a) Resource Pricing (50,40,20)

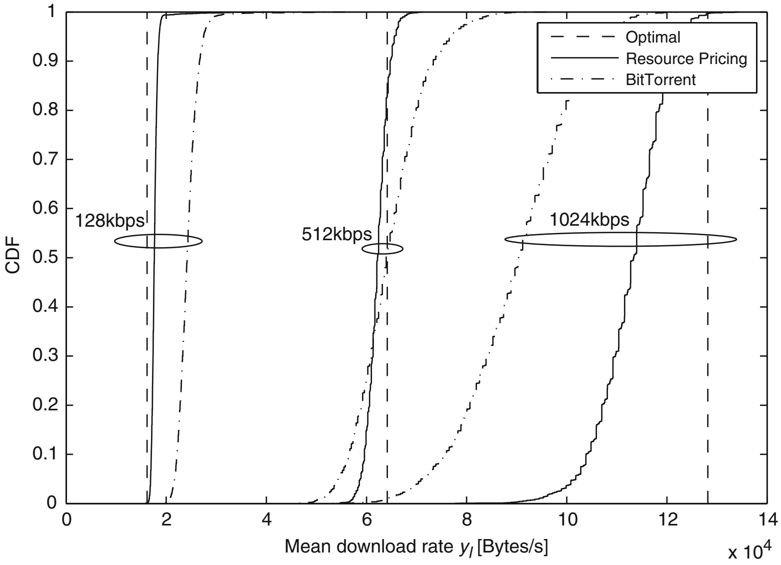
(b) Resource Pricing (50,10,10)

Fig. 7 Number of copies of rarest and most frequent piece in the network for parameters (N_R, N_{MI}, N_{MP})

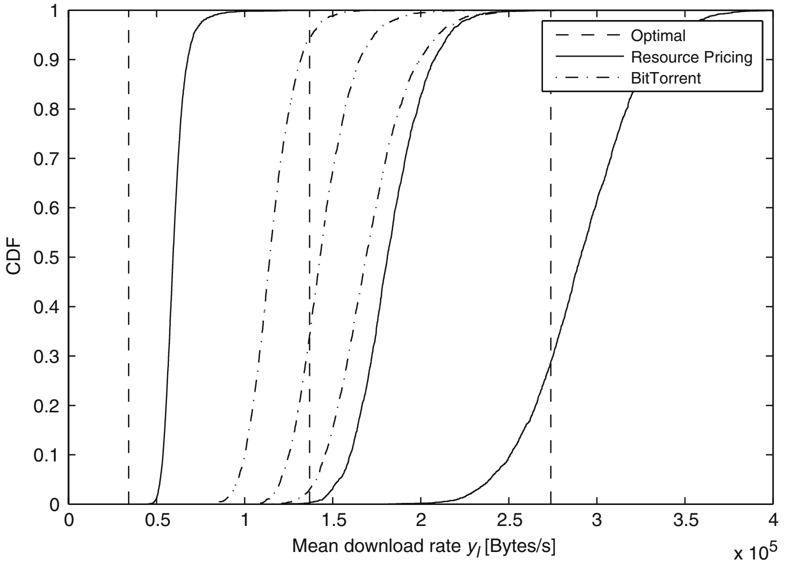
The piece availability in the network is unstable and peers have highly different download times. This is different in Fig. 7b. Here, the piece selection works fine and the network is time invariant. This problem is not specific to resource pricing. Also BitTorrent shows similar behaviour when peers unchoke a larger number of remote peers (e.g. with 20 unchokes for the discussed simulation scenario). The reason for inefficiency lies in the time it takes until a full chunk is uploaded. With many upload connections the rate per connection decreases and it takes longer to complete a specific chunk. Hence, when a chunk becomes rare it takes longer with many connections until further peers offer copies of it. Furthermore, the piece availability depends on the number of peers in the network. For this simulation setup all pieces are distributed randomly at the beginning. Hence, on average each piece is available at half of the peers in the network. With an increasing number of peers also the piece availability increases in absolute numbers. And, simulations with 1000 peers show that resource pricing is efficient also with $N_R = 50$, $N_{MI} = 40$ and $N_{MP} = 20$.

The differences between resource pricing and BitTorrent are small for networks where most of the peers are leechers and have homogeneous upload capacities. This changes when peers have different upload capacities and altruistic behaviour is common. Figure 8 shows the CDFs for the mean download rate of peers with different upload capacities of 128 kbps, 512 kbps and 1024 kbps, which are assigned randomly to the peers. In Fig. 8a 1 seed and 99 leechers are in the network. The optimal download rates are computed with (19). With resource pricing the download rate is closer to the optimal values than with BitTorrent. Identical to the results from Section 4 the download performance with BitTorrent is better for peers with lower capacities than with higher capacities. Also the range for the same capacity is larger for BitTorrent than with resource pricing.

When a large number of seeds is present in the network BitTorrent fails to provide a strong incentive. This is shown in Fig. 8b, where half of the peers are seeds. As in Fig. 8a the curve to the left, middle and right for each method corresponds to an upload capacity of 128 , 512 , and 1024 kbps, respectively. Since the three curves of



(a) 1 seed and 99 leechers



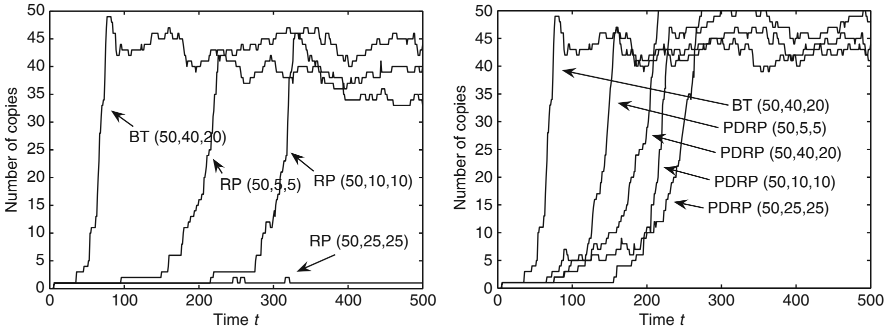
(b) 50 seeds and leechers

Fig. 8 Network with heterogeneous capacities for resource pricing (50,10,10) and BitTorrent (50,40,20) with parameters (N_R, N_{MI}, N_{MP})

BitTorrent are close together, peers do not gain much by contributing higher upload bandwidth to the network. With resource pricing the incentive for peers is much stronger and the curves are closer to the computed optimal values.

Up to this point we restricted our attention to a scenario where pieces are frequently available at different peers in the network. This is a desirable state of the network since download times are near to the optimal values. In the following we study a scenario where one chunk is rare and observe if the network gets out of this state and attains a state with balanced piece availability. Assume the rarest chunk is uploaded by the seed to U other peers. Furthermore, when a peer completes the rarest chunk it uploads it to others. Hence, for homogeneous peers with capacity C the number of copies of the rarest chunk increases by $(U+1)^i$, where i is the number of time intervals it takes to upload a full chunk to U peers. With a chunk size of S_C this time interval can be computed with $U \cdot S_C/C$. Hence, the increase of the rarest chunk is exponentially and at fastest rate with $U = 1$. For example, with the default chunk size and homogeneous peers with a capacity of 1 Mbps it takes around 15 s and 31 s to disseminate a chunk to over hundred peers with one and five current uploads, respectively.

In simulations we assume the chunk with id 0 is only available at the seed and all other peers have a random set of chunks except chunk 0. Results for BitTorrent and resource pricing are depicted in Fig. 9a for a specific run and are summarised for 10 runs in Table 1. The average time until chunk 0 is no longer the rarest chunk in the network is given in Table 1 along with the 95% confidence intervals.



(a) RP

(b) PDRP

Fig. 9 Dissemination of one rare chunk

Table 1 Time until chunk 0 is not the rarest chunk (and 95% confidence intervals)

BT (50,40,20)	RP (50,5,5)	RP (50,10,10)	RP (50,25,25)	RP (50,40,20)
85 ± 11	206 ± 37	358 ± 46	1393 ± 423	2120 ± 579

Although the simulation results are much larger than the analytical numbers also the measured dissemination time for the rare chunk increases with the number of upload connections. With BitTorrent a peer uploads to five other peers at a time and after around 85 s the chunk 0 is available frequently in the network. With resource pricing and $N_{MI} = 5$ each peer has more than five connections because it accepts remote connection requests. Hence, it takes around 206 s to disseminate the chunk in the network. For larger values of N_{MI} the time for dissemination increases considerably.

In summary, the results for resource pricing show that fairness is increased with the number of connections up to a point where the network becomes inefficient. Furthermore, dissemination of a rare chunk is better for a small number of upload connections. Hence, there is a trade-off between fairness and efficiency. To adapt dynamically to the network conditions we extend resource pricing in the following and make the price offer dependent on the piece availability.

5.1 Piece-Dependent Resource Pricing

We use a linear approach to weigh the price offer λ_c on every connection depending on the requested chunk. If the chunk is rare then higher prices are offered. Thus, the server will increase its rate on this connection. However, this works only as long as a server uploads and a client downloads different chunks. When all peers request the same piece also the price offers are similar and roughly the same upload rate is allocated to them. This extension of resource pricing is denoted as Piece-Dependent Resource Pricing (PDRP) in the following.

Each client estimates the availability of chunks based on the chunk information of its neighbours. Denote the number of neighbours of client c that completed chunk i as f_c^i. The price offer of client c for chunk i is

$$\lambda_c^i = \alpha_c \frac{\lambda_c}{f_c^i}. \tag{20}$$

Here, α_c normalises the price offers such that each peer does not offer higher prices than allowed with its willingness-to-pay. To ensure that $W_c = \sum_{s \in \mathcal{S}(c)} x_{sc} \lambda_c^i$ holds we set

$$\alpha_c = \frac{W_c}{\sum_{s \in \mathcal{S}(c)} \frac{x_{sc} \lambda_c^i}{f_c^i}}. \tag{21}$$

With (20) the weighting factor increases linearly with the rareness of a chunk. For example, a client offers the double price for a piece that is half as frequent as another piece.

Table 2 Time until chunk 0 is not the rarest chunk for PDRP

PDRP (50,5,5)	PDRP (50,10,10)	PDRP (50,25,25)	PDRP (50,40,20)
153 ± 15	205 ± 25	238 ± 18	208 ± 15

We repeat the simulation with the rare chunk for piece-dependent resource pricing. The results are given in Fig. 9b and Table 2. For resource pricing with piece-dependent price offers the rare chunk is disseminated faster for all studied parameter sets. The differences for different parameters with $N_{MI} \geq 10$ are small. However, BitTorrent provides also in this case the best results because peers upload to a small number of other peers. This means, weighting the price offers by the piece availability improves the dissemination time to some extent but also the number of uploads should be decreased when some pieces are rare.

We also re-run the other simulations for PDRP. Here, all pieces are frequently available in the network. Hence, PDRP produces very similar results as compared to the original implementation. Also for many connections ($N_R = 50$, $N_{MI} = 40$ and $N_{MP} = 20$) the weighting of the price offer is too slow and the performance is as poor as with resource pricing.

In conclusion, BitTorrent networks are a very efficient way for the fast dissemination of large content in the Internet. Although it provides an incentive to contribute upload bandwidth to the network it does not avoid unfairness among the peers. Especially, for peers with different upload capacities and for altruistically contributed resources the performance becomes unfair. Resource pricing and its extension piece-dependent resource pricing increases fairness. For a medium number of uploads (e.g. for the parameter set (N_R,N_{MI},N_{MP}))=(50,10,10)) the proposed algorithms outperform BitTorrent and give peers a strong incentive to contribute resources. Only for parameter sets where the upload bandwidth is divided among many peers it takes long until a full piece is provided to another peer. Hence, rare chunks might emerge. To some extent this can be avoided by higher price offers for rarer chunks. However, as in BitTorrent, limiting the number of concurrent uploads is a more efficient way. To trade off between fairness and dissemination time of rare chunks piece-dependent resource pricing is a good compromise.

6 TCPeer: A TCP Variant for P2P CDNs

The prevalent mechanism to avoid congestion in IP networks is the control of the sending rate with TCP. Dynamic routing strategies at the IP layer are not deployed because of problems like route oscillations and out-of-order packet deliveries. However, with the adoption of P2P technology, routing is done also in these overlay networks. With multi-source download protocols peers upload and download to/from other peers in parallel. Hence, by controlling the sending rates between the peers we may achieve two desirable properties: load balancing at the peers, which upload data, and the avoidance of congested links in the IP network. Considering load balancing, P2P technology has found already its way to content distribution networks [22]. But most of the work assumes that a file request is routed to a single server (in a sophisticated way). Multi-source download protocols adapt faster to changes at the servers or in the network because data is requested at multiple peers in parallel. Hence, the total bandwidth of all serving peers can be allocated in a fair manner to the client peers. Furthermore, also congestion can be avoided. When a connection between two peers becomes overloaded, the uploading peer can favour another peer on an uncongested route. Thus, it uses its upload bandwidth more efficiently. On the other hand the downloading peer could ask for a higher download rate at other providers of the file. Thus, it could receive the same download rate as in an uncongested network.

Based on the resource pricing algorithm for P2P networks, in Section 2.2, we propose in [9] a rate control algorithm for P2P over IP networks. Here, only the basic idea is outlined: The approach is denoted as *TCPeer*, because a peer adopts the functionality of TCP and extends it with information from the overlay network. A distributed algorithm is proposed, where sending peers control the rate depending on the price offers by the remote peers and the path prices charged by the routers in the IP network. Since sending rates depend on the price offers of the remote peers the algorithm ensures fairness among peers. Furthermore, indicated by the path price a serving peer is aware of the level of congestion in the underlying IP network. Thus, it can shift upload bandwidth from congested to uncongested flows. Because of the change in price offers at the receiving peers other senders will change their sending rates accordingly. The rate allocation of all peers converges to an efficient and fair allocation of bandwidth in the network. Furthermore, P2P traffic avoids congested links when uncongested routes between some peers exist. In [9] we validate the functionality of the new approach analytically and by simulations for a small network, where explicit price information is assumed.

7 Conclusion

In P2P content distribution networks, a peer chooses freely the peers it uploads to and allocates bandwidth for it. However, peers act frequently on their own interest. Hence, an incentive to contribute resources to the P2P network should be given to peers. In this chapter resource pricing for P2P networks is proposed to ensure fairness among peers. Fairness is of great importance in P2P networks, because it ensures that the download performance of a peer depends on its allocated upload bandwidth. Hence, free-riding is decreased and/or avoided in the network.

From a P2P network model two distributed pricing algorithms are derived, which are denoted as resource pricing and reciprocal rate control. Both increase fairness among peers as compared to the tit-for-tat scheme of BitTorrent. When the piece availability is taken into account, a trade-off between efficiency and fairness has to be done. With an increasing number of uploads, fairness is improved. However, rare pieces might emerge because the upload bandwidth is divided between many peers. One alternative is to set higher prices for rare pieces. This will fasten the dissemination of rare pieces.

The proposed algorithms are based on the resource pricing model used for IP networks. Hence, P2P and IP network can be modelled in a single optimisation problem and TCP variants optimised for the use with multi-source download can be derived.

References

1. A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and improving BitTorrent performance," Microsoft Research, Tech. Rep. MSR-TR-2005-03, 2005. [Online]. Available: http://www.research.microsoft.com/~padmanab/papers/msr-tr-2005-03.pdf
2. D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice-Hall. Inc., 1989.
3. "Bittorrent protocol specification v1.0." [Online]. Available: http://wiki.theory.org/BitTorrentSpecification
4. B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. 1st Workshop on Economics of Peer-to-Peer Systems*, Berkeley, Jun. 2003.
5. W. Chehai, L. Xianliang, and D. Hancong, "Analysis of content availability optimization in BitTorrent," in *Proc. 2006 International Conference on Hybrid Information Technology (ICHIT'06)*, Los Alamitos, CA, USA, Nov. 2006, pp. 525–529.
6. K. Eger, "A pricing approach to distributed resource allocation in IP and peer-to-peer networks," Ph.D. dissertation, Hamburg University of Technology, 2008.
7. K. Eger, T. Hoßfeld, A. Binzenhöfer, and G. Kunzmann, "Efficient simulation of large-scale P2P networks: Packet-level vs. flow-level simulations," in *2nd Workshop on the Use of P2P, GRID and Agents for the Development of Content Networks (UPGRADE-CN'07)*, Monterey Bay, CA, USA, Jun. 2007, pp. 9–16.
8. K. Eger and U. Killat, "Resource pricing in peer-to-peer networks," *IEEE Communications Letters*, vol. 11, no. 1, pp. 82–84, Jan. 2007.
9. K. Eger and U. Killat, "TCPeer: Rate control in P2P over IP networks," in *LNCS 4516: 20th International Teletraffic Congress (ITC20)*, Ottawa, Canada, Jun. 2007, pp. 618–629.
10. K. Eger and U. Killat, "Bandwidth trading in BitTorrent-like P2P networks for content distribution," *Computer Communications, Special Issue: Foundation of Peer-to-Peer Computing*, vol. 31, no. 2, pp. 201–211, Feb. 2008.
11. S. Floyd, "Measurement studies of end-to-end congestion control in the Internet," Last modified in April 2007. [Online]. Available: http://www.icir.org/floyd/ccmeasure.html
12. C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE INFOCOM*, Miami, Mar. 2005.
13. B. Hubert, T. Graf, G. Maxwell, R. van Mook, M. van Oosterhout, P. Schroeder, J. Spaans, and P. Larroy, *Linux Advanced Routing & Traffic Control HOWTO*. [Online]. Available: http://lartc.org/
14. F. Kelly, "Charging and rate control for elastic traffic," in *European Transactions on Telecommunications*, vol. 8, Jan. 1997, pp. 33–37.
15. F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," in *Journal of the Operational Research Society*, vol. 49, Mar. 1998, pp. 237–252.

16. A. Liotta and L. Lin, “IP multimedia subsystem - the operator’s response to P2P service demand,” *IEEE Communications Magazine*, vol. 45, no. 7, pp. 76–83, Jul. 2007.
17. A. Odlyzko, “Data networks are lightly utilized, and will stay that way,” *Review of Network Economics*, vol. 2, no. 3, pp. 210–237, Sep. 2003.
18. M. J. Osborne, *An Introduction to Game Theory*. Oxford University Press, 2004.
19. D. Qiu and R. Srikant, “Modeling and performance analysis of BitTorrent-like peer-to-peer networks,” *Computer Communication Review*, vol. 34, no. 4, pp. 367–378, 2004.
20. S. Shenker, “Fundamental design issues for the future Internet,” *IEEE Journal on Selected Areas in Communication*, vol. 13, no. 7, Sep. 1995.
21. R. Thommes and M. Coates, “BitTorrent fairness: analysis and improvements,” in *Proc. Workshop Internet, Telecom. and Signal Proc.*, Noosa, Australia, Dec. 2005.
22. E. Turrini and F. Panzieri, “Using P2P techniques for content distribution internetworking: A research proposal,” in *Proc. IEEE P2P 2002*, 2002.

Part VI
Trust, Anonymity, and Privacy

Reputation-Based Trust Management in Peer-to-Peer Systems: Taxonomy and Anatomy

Loubna Mekouar, Youssef Iraqi, and Raouf Boutaba

Abstract Trust is required in file sharing peer-to-peer (P2P) systems to achieve better cooperation among peers and reduce malicious uploads. In reputation-based P2P systems, reputation is used to build trust among peers based on their past transactions and feedbacks from other peers. In these systems, reputable peers will usually be selected to upload requested files, decreasing significantly malicious uploads in the system. This chapter surveys different reputation-based P2P systems. We will breakdown a typical reputation system into functional components. We will discuss each component and present proposed solutions from the literature. Different reputation-based systems will be described and analyzed. Each system presents a particular perspective in addressing peers' reputation.

1 Introduction and Motivation

The most important challenge of online environments is to assure positive interactions and satisfactory transactions. People will usually back up from dealing with strangers that they did not know before. Therefore, people minimize their interactions and transactions and tend to remain in their comfort zone. Peer-to-Peer file sharing systems provide a large collection of files available for download. In traditional systems, little information is given to the user to help in the peer-selection and file-selection processes. For example, if a user wants to download a file, the user is given a list of peers that have the requested file. The process of selecting the right peer with no a priori information is frustrating and risky. To foster positive

Loubna Mekouar
University of Waterloo, Waterloo, Canada, e-mail: lmekouar@bbcr.uwaterloo.ca

Youssef Iraqi
Dhofar University, Salalah, Oman, e-mail: y_iraqi@du.edu.om

Raouf Boutaba
University of Waterloo, Waterloo, Canada, e-mail: rboutaba@bbcr.uwaterloo.ca

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_24, © Springer Science+Business Media, LLC 2010

interactions and reduce the risk involved in P2P file sharing systems, peers need to reason about trust and reputation systems are used to this end. Reputation systems are based on collecting information about peers' past transactions and computing a reputation value for these peers. The reputation values will be the basis for identifying trustworthy peers.

In a P2P system, peers communicate directly with each other to exchange information and share files. P2P systems can be divided into several categories. Centralized P2P systems (e.g., Napster [14]), use a centralized control server to manage the system while downloading a file is achieved directly between peers. These systems suffer from the single point of failure, scalability and censorship problems. Decentralized unstructured P2P systems try to distribute the control over several peers. They can be divided into completely-decentralized and partially-decentralized systems. Completely-decentralized systems (e.g., Gnutella [2]) have no hierarchical structure between the peers and all peers have equal role and responsibilities. Partially-decentralized P2P systems (e.g., KaZaa [4], Morpheus [5], and Gnutella2 [3]) occupy the middle ground between centralized and completely decentralized systems. These systems have been proposed to reduce the control overhead needed to run the P2P system by the use of "superpeers" or "supernodes". Supernodes are peers that have extra capabilities and assume more responsibilities than regular peers. A supernode act as a centralized server for the peers connected to it. Decentralized structured P2P systems like CAN [46] and CHORD [54] are based on Distributed Hash Tables (DHT) in that they use a hash function to deterministically map keys such as file names into points in a logical coordinate space. Peers are responsible for storing (key, value) pairs that are hashed into a point that is located within their region. Most P2P systems deployed on the Internet are unstructured.

In an open P2P system, peers often have to interact with unknown peers and need to manage the risks involved in these interactions. For example, if a user wants to download a file, the user is given a list of peers that can provide the requested file. The user has then to choose one peer from which the download will be performed. This process is frustrating to the user as no help is given on how to choose the right peer. After the download has finished, the user has to check the received file for malicious content and that it actually corresponds to the requested file (i.e., the requested content). If the file is corrupted, the user has to start the process again. In traditional P2P systems, little information is given to the user to help in the selection process. To solve this problem, peers need to be able to reason about trust in order to avoid untrustworthy peers and reduce risks. Trust management [13] is any mechanism that allows to establish mutual trust which allows peers to cooperate, and obtain in the long term an increased utility for the participating peers.

In [55], the authors classify trust management systems into three categories:

- Credential-based trust management systems: in these systems, service providers and the provided services are trusted, but service requesters are not. Service providers use credentials to evaluate the trustworthiness of service requesters and services may be granted or not.
- Reputation-based trust management systems: in these systems, service providers and provided services are not trusted. Service requesters select service providers

based on their reputation values. Reputable service providers are selected to provide the service.
- Social network-based trust management systems: these systems are based on social networks. Reputation is computed based on social relationships.

The focus in this work is on reputation-based trust management in P2P file sharing systems. Reputation-based P2P systems [13, 17, 20, 28, 29, 32, 36, 39, 42, 61] were introduced to build trust among peers. These systems try to evaluate the transactions performed by the peers and associate a reputation value to each peer. The reputation values will be used as selection criteria among peers.

In this chapter, we will investigate the existing research work proposed to address reputation in P2P systems. We will present existing reputation management schemes in centralized, completely decentralized and partially decentralized P2P systems. We will describe these schemes and identify key properties for each reputation system to summarize the efforts of researchers in addressing peers' reputation.

2 Traditional Systems Versus Reputation-Based Systems

The following is the life cycle of a transaction in a traditional P2P system (Fig. 1a):

1. Send a file request (either to the supernode or to other peers depending on the P2P system)
2. Receive a list of peers that have the requested file
3. Select a peer
4. Download the file

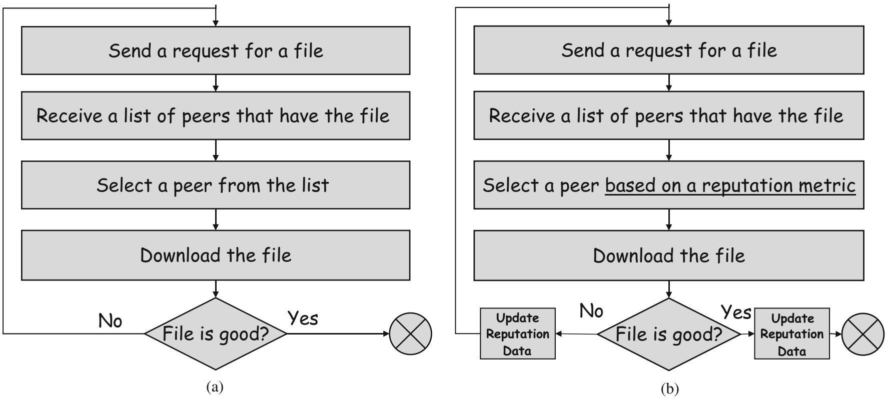

Send a request for a file
Receive a list of peers that have the file
Select a peer from the list
Download the file
No
File is good?
Yes
(a)
Send a request for a file
Receive a list of peers that have the file
Select a peer based on a reputation metric
Download the file
Update Reputation Data
No
File is good?
Yes
Update Reputation Data
(b)

Fig. 1 (**a**) Life cycle in a traditional P2P system, (**b**) Life cycle in a reputation-based P2P system

The following is the life cycle of a transaction in a reputation-based P2P system (Fig. 1b):

1. Send a file request
2. Receive a list of peers that have the requested file
3. Select a peer or a set of peers, based on a reputation metric
4. Download the file
5. Send feedback and update the reputation data

Most reputation management schemes try to achieve the following goals:

1. Make informed decisions about transaction partners and choose appropriate peers to download files
2. Isolate malicious peers from the network by downloading files only from reputable peers, hence reducing malicious uploads
3. Increase users' confidence and satisfaction
4. Help in the bootstrapping process to select peers to connect to in the overlay network.
5. Motivate peers to exhibit good behavior
6. Use the network resources more efficiently

3 Trust and Reputation

3.1 Trust Definition

Trust is as old as the existence of human beings on this earth. People were grouped in tribes and within the same tribe, they trusted each other. The concept of trust has a significant role in the surviving of human beings. We experience and rely on trust on daily basis. However, trust is difficult to define clearly and precisely.

Researchers from different fields such as psychology, sociology, philosophy, history, law, business and economics have tackled the concept of trust from different views. According to the Oxford dictionary [6], *trust is a firm belief in the reliability, truth, ability, or strength of someone or something.*

In 1973, Deutsch [21] has specified that *trust is the confidence that an individual will find what is desired from another, rather than what is feared.*

In 1990, Diego Gambetta [25] defined trust as follows: *Trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent will perform a particular action, both before we can monitor such action (or independently of his capacity of ever to be able to monitor it) and in a context in which it affects our own action.*

In 1996, McKnight and Chervany defined trust as *Trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security even though negative consequences are possible.*

In 2000, Grandison and Sloman [26] defined trust as *The firm belief in the competence of an entity to act dependably, securely and reliability within a specified context.*

In this chapter, we adopt the following definition proposed in 2006 by Chang et al. [15]. Trust is defined as *The belief the trusting agent has in the trusted agent's willingness and capability to deliver a mutually agreed service in a given context and in a given time slot.*

Based on our experience in the physical world, we extract the necessary information that can help us build trust in the virtual world to increase users' confidence and reduce the risk. Marsh [35] is one of the first researchers to give a formal model of trust that can be used in computer science. This model is based on social properties of trust taken from sociology.

3.2 Reputation Definition

Reputation has been widely used in different disciplines such as psychology, sociology, business and economics. From the Oxford dictionary, *reputation is the beliefs or opinions that are generally held about someone or something*. Abdul Rahman et al. [12] define reputation as "*an expectation about an agent's behavior based on information about its past behavior*". Sabater et al. [49] define it as an "*opinion or view of one about something*". Mui et al. [41] define it as "*the perception that an agent creates through past actions about its intentions and norms*". In service-oriented environments, Chang et al. [15] define reputation as *"an aggregation of the recommendations from all of the third-party recommendations agents and their first, second and third hand opinions as well as the trustworthiness of the recommendation agent in giving correct recommendations to the trusting agent about the quality of the trusted agent*".

3.3 Trust Properties

Trust is personal, subjective and it is based on various factors (i.e., endogenous and exogenous [15]). In [29], trust can be measured objectively by tracking peers' contribution to the system.

Trust is fuzzy since trust is imprecise and vague. Trust is dynamic since it is not stable and it changes as time goes by. Trust is also complex since different ways are possible for determining trust [15].

The trust relationship is usually asymmetric. The transaction between the trusting peer and the trusted peer results in a trust value assigned by the trusting peer to the trusted peer. This value shows the strength of the trust relationship.

The trust relationship can be transitive. In transitive trust [32, 33], if Alice trusts Bob and Bob trusts Claire and Alice wants to interact with Claire. Alice asks Bob. Bob will refer Claire to Alice. Alice derives a measure of trust in Claire based on Bob's trust to Claire and Alice's trust to Bob.

3.4 System Model

3.4.1 Modeling the Network

The success of file sharing applications has been behind the tremendous attention given to P2P technology. As a consequence, this technology has been applied to different distributed applications (e.g., Skype [8], PPLive [7], UUsee [9]). P2P applications have witnessed tremendous success among end users.

Centralized P2P systems use a centralized index for the files shared by each peer. It simplifies the direct exchange and the sharing of files between peers. However, it represents a single point of failure which reduces the reliability of the system. In completely decentralized P2P systems, a central authority for storing data and handling all the queries is not available. Interconnected peers are able to participate in transactions by interacting with each other and make local autonomous decisions to achieve their objectives. Peers are responsible for storing, sharing information and handling the queries. Peers act as clients and request services from other peers as well as servers and provide services to other peers. These systems provide improved robustness and enhanced scalability compared to centralized systems. In [55], the authors indicate the interplay between the degree of autonomy and the requirement of trust. In these systems, peers are autonomous which leads to loosing control over peers. Since no global view of the system is available, peers will get a partial and limited perspective of the system by receiving information from other peers. Trust is more needed to facilitate the interaction among peers. In partially decentralized P2P systems, some peers (named “superpeers”, “supernodes”) act as local central indexes for files shared by local peers. These systems provide lower discovery time as the discovery process involves only the supernodes.

In P2P systems, the following terms are used:

- Peers, users, nodes, agents: peers issue queries and reply to them
- Files, resources, items: the requested items from requester peers.
- Transactions: interactions and experiences between peers. In P2P systems, these interactions can be sharing a file, CPU cycles, or disk storage. A transaction in a P2P system involves:
  - The requester peer: the peer that requests the service.
  - The provider peer: the peer that has the requested service and is available to provide it to the requester peer.
  - Intermediary peer: the peer that is involved in a transaction. This peer participates to accomplish the transaction and to fulfill the request (e.g., forwarding messages). The intermediary peer is also referred as a third party peer or a mediator or a queried peer.
  - Recommender peer: peer that sends a feedback to the requester peer.
  - Feedback, opinion, rating, experience, appreciation: a peer sends a feedback as a result of a transaction. A rating is one trust value, while an opinion is

considered as an aggregation of all trust values based on previous transactions with a provider peer.

The following is a list of most important and representative reputation systems that have been proposed in the literature and will be used throughout the chapter: eBay [1], DistributedTrust [11], BinaryTrust [13], P2PBasic and P2PEnhanced [17], EigenTrust [32], Nice [33], DCRC and CORC [29], Travos [56], XRep [18], Regret [50], MDNT [15], MLE [20], LimitedReputation [36], CredibilityRecords [51], IDA and MDA [39], FineGrainedTrust [61], and PowerTrust [62].

3.4.2 Modeling the Nodes

In a P2P file sharing system, peers are expected to exhibit good behavior. Ideally, peers are expected to be trusted, that they will share good quality files, that they will upload requested files, and that they will send honest feedbacks. Unfortunately, real life P2P systems have proved that a mechanism is needed to measure explicitly trust in order to deal only with trustworthy peers.

There are two types of users' behavior: Good peers and malicious peers.

Good Peers

Those are the well-behaved peers that will usually send an authentic file. The following is the expected behavior from good peers:

- Availability: available to share a file, to forward a query, to reply to a query.
- Contribution: a peer contributes positively to the system by uploading authentic files.
- Credibility/Honesty: Upon receiving a reputation query, a recommender peer sends an honest feedback.

Malicious Peers

Peers that misbehave to impact badly the system. The adversary is another term to refer to malicious peers as used in [37]. The maliciousness of peers can appear in different ways and malicious peers can act individually or by forming collectives. Malicious peers can be one or more of the following:

- Senders of inauthentic data: these peers upload inauthentic and corrupted data and send spurious information to pollute the system.
- Liar peers: these peers lie in their feedbacks.
- Free riders: take advantage from the system without contributing to it. Although these peers do not harm directly other peers, they do have an impact on the performance of the system. In the literature, they are also referred to as selfish peers, free-loaders [29] or rogue peers [52].
- Traitors: peers that induce the milking phenomenon. These peers provide authentic files for a while to get a high reputation value before starting uploading inauthentic files. These peers abuse from this reputation by harming other peers, defecting on transactions and increasing the reputation of malicious peers.

- Colluding Peers: these peers know each other and work together to form a malicious collective. These malicious peers can upload inauthentic files, decrease the reputation of good peers (i.e., defaming), or increase the reputation of other malicious peers (i.e., flattering).
- Whitewashers: these peers join the system, act maliciously or misbehave then leave and re-join with new identities to get a new reputation value and eliminate the bad reputation received previously. In general, it is difficult to trace these peers and prevent the system from considering them as regular new comers. Systems with weak identities are vulnerable to whitewashing. Examples of these systems include KaZaA, Gnutella, and EigenTrust.
- Sybil Attacks: the malicious attacker creates multiple identities allowing the attacker to control a whole portion of the network.
- Denial of Service (DoS) attackers: these peers consume large amount of resources from the system to subvert the whole system and bring it down. In P2P systems, detection, management, and the prevention from these attacks is still an open problem.
- Unreliable: they can promise availability of specific services and do not deliver them.
- Imposter: they can pose as other peers.

Malicious peers act maliciously for different reasons, including personal characteristics, psychological, social, and economic factors. For example, movie and music industry pollutes P2P networks to minimize the use of these systems and to push users to buy their products rather than sharing them for free using P2P file sharing applications. Polluting these systems is achieved by distributing files with legitimate titles but with silence or random noise.

Peers Identity

To distinguish between peers, identity is required. Each peer in the system is identified by its identity that must be kept the same during its lifetime. As an example, users in Nice choose a PGP style identifier. The identifier includes a plain text identification string and a public key. Desirable characteristics of peers' identity related to trust include:

- Longevity: peers need to keep the same identifiers during their lifetime [47]. It should be difficult for peers to change their identity easily (i.e., whitewashing) and start over by obliterating their past behavior.
- Anonymity: according to the application used, peers' identity can belong to different levels of anonymity. Most P2P applications use a simple, user-generated pseudonyms. In some applications (e.g., freenet), onion routing is used to provide the anonymity of peers. However, since the peer identity is protected, it will be difficult to track the actions done by this peer. In general, there is a tradeoff between anonymity and trust.
- Unforgeability: Unforgeable identities are generated by a central trusted entity or a set of trusted entities and are given to new comers when they join the system. Each user is given only one identity. In TrustMe, for example, users are assigned

unforgeable identities. Unforgeable identity is used to protect from whitewashing (i.e., change of identity to eliminate peer's history) and sybil attacks (i.e., the use of multiple identities).

- Spoof-resistance: to prevent malicious peers from impersonating other peers identities.
- Sybil-attack resistance: peers identity should be resistant to sybil attacks. Some mechanisms have been proposed to slow the creation of new identities such as imposing an entry cost or a registration fee (e.g., Nice), requiring from users to solve a puzzle that can not be done by a computer, or read some text from a JPEG file.

New Comers Policy

When new comers join the system, these peers do not possess any local information regarding other peers in the system. In addition, all the peers in the system do not know yet the reputation of these new comers. Since new comers to the system are strangers and their behavior is not known yet, two policies can be adopted:

- The optimistic approach: new comers are trusted. However, other peers may be disappointed.
- The pessimistic approach: new comers are not trusted. However, in this case, new comers may not be able to build their reputation.

Reputation systems should welcome new comers and give them a chance to increase their reputation gradually. At the same time, these systems should protect peers from a new comer in case it turned out to be malicious.

New comers have to assume their responsibility in establishing trust with other peers by uploading authentic files. They should build and maintain good reputation to become reputable peers.

3.4.3 Modeling the Reputation Management Infrastructure

In centralized P2P systems, there is a central entity where information regarding peers' actions can be gathered and accumulated. Peers are controlled and malicious peers can be identified somewhat easily. Peers can be protected against malicious actions performed by malicious peers. For these systems, reputation management is also centralized since the central entity can handle the trust information in addition to the lookup mechanism. The use of a centralized reputation management infrastructure simplifies significantly the management of trust information. In completely decentralized systems, malicious peers have more freedom to act maliciously and perform variety of attacks and it is more difficult to gather all the actions performed by the peers. In these systems, peers should protect themselves from malicious peers since they can not be identified easily and hence isolated from the system. Peers are required to gather information to assess the reliability of peers in providing the requested service. In these systems, a centralized reputation management entity is possible. [29] is an example where the Reputation Computation Agent RCA is used to collect reputation information and compute peers' reputation. However, the use of a centralized entity will exhibit a single point of failure and will be easy to attack. A natural possibility is to use a decentralized reputation management infrastructure to improve the robustness and the scalability of the system. In partially decentralized systems, a centralized reputation management entity is possible. To avoid the drawbacks of the centralized entity, a partially decentralized reputation management system is proposed as in IDA and MDA [39]. In these schemes, the authors take advantage from the use of supernodes in the partially decentralized architecture. Since supernodes are already used to handle search requests on behalf of the peers connected to them, supernodes are used to handle trust data and compute peers' reputation.

The followings are advantages and drawbacks of different types of trust and reputation management infrastructures:

- Centralized systems: use a trusted centralized server for managing reputation information. This server will handle and manage all the reputation data
  - Advantages:
    1. Easy to use and manage
    2. Simplifies the mechanism design
    3. Offers efficiency since all the reputation data received for peers will be handled to the requester peer.
    4. Helps keeping the data consistent and coherent
  - Drawbacks:
    1. Difficult to achieve in case not all the peers can trust one entity.
    2. Represents a single point of failure and a bottleneck since millions of peers can send queries to this entity.
    3. Represents a single point of attacks by misbehaving peers such as DoS attacks, sabotage and subversion
    4. Expensive to achieve high performance and robustness
    5. Not scalable
    6. Not resistant to lawsuit
- Completely decentralized systems: information regarding peers' reputation is stored at the peer's level and this information is scattered throughout the network.
  - Advantages:
    1. No single point of failure
    2. No need for a globally trusted entity
    3. Provides robustness and scalability
    4. Resistant to lawsuit
  - Drawbacks
    1. Message overhead as many messages are generated in order to maintain and manage reputation data

2. Flooding is required to get the required information from peers in order to aggregate different local trust values
3. The topology changes frequently due to the transient nature of peers that can join or leave the system at any time. Some reputation data may be lost or inaccessible.
4. Reputation data is more vulnerable to tampering with.

- Partially decentralized systems
  - Advantages:
    1. Efficient search for reputation data of the provider peers compared to completely decentralized systems
    2. Easier to manage
    3. Resistant to lawsuit
    4. Scalable
    5. Small number of supernodes to manage reputation data
    6. Easy access to reputation data by other supernodes
    7. Alleviating the peer from the burden of replying to unnecessary queries as it is the case in completely decentralized P2P systems. Peers are contacted by their supernodes only if they have the requested file.
    8. Efficient enforcement of service differentiation
  - Drawbacks:
    1. Supernodes should be trusted to handle reputation data
    2. Additional load for supernodes

4 The Anatomy of Reputation Systems

The goal from file sharing applications is to share files, and sharing experiences between peers will ultimately lead to providing peers with the adequate environment by identifying good peers and isolating malicious ones. Reputation-based P2P systems create the appropriate environment for peers to rely on each other for downloading authentic files. Since we can not predict future peers actions, peers' past actions are used to measure its reputation.

Open systems need robust reputation management. A good reputation system should [30]:

- Identify malicious peers in order not to be selected as transactions partners: this is a precaution measure before starting the transaction.
- Spread information regarding a malicious peer in case that a negative transaction occurred: this is a revenge measure after ending a transaction to help other peers in future interactions.

The survey of different reputation systems reveals the important mechanisms used to achieve good reputation management. The peer looking for a file is the requester peer, and peers that have the requested file are the provider peers. The components and issues involved in finding a file and downloading it in the case of a reputation-based P2P file sharing application are the followings:

1. The Local Trust: Here important issues include:
   a. What kind of trust information is gathered?
   b. Where to store trust information?
   c. Is the local trust information sufficient?
2. The Reputation Query: The requester peer sends a reputation query regarding the potential provider peers. Related issues include:
   a. To whom the requester peer should send a reputation query?
   b. How many peers, should the requester peer contact?
   c. What kind of information is sent to the requester peer?
3. The Reputation Computation
   a. How to deal with liar recommender peers?
   b. How to deal with fraudulent recommender peers?
   c. How to compute the reputation?
4. The Use of Reputation
   a. How to choose a peer based on the reputation value?
   b. How to evaluate a transaction after downloading a requested file?
5. Credibility Assessment
6. Incentives, Rewards and Punishment

In the following, we will explain and describe each one of these important components and present important solutions proposed in the literature.

4.1 The Local Trust

The requester peer sends a file search query and gets a search reply from the system containing a list of peers. The requester peer may or may not have previously interacted with some of the peers in the list. If the requester peer is not a new comer, and has already downloaded files from other peers, local trust information based on previous transactions can be used. The requester peer may:

- Eliminate from the list malicious peers that it has already interacted with.
- Accept to deal with good peers that have already provided him a good service.
- May not have any trust information for some of the peers if it did not interact with them in the past.

4.1.1 What Kind of Trust Information is Gathered?

Some reputation management schemes uses the number of negative and positive downloads (e.g., EigenTrust), other schemes use the negative downloads only (e.g., complaints in Binarytrust). For other schemes the size of the download is more important than the number of uploads (e.g., IDA, MDA). In debit-credit reputation computation DCRC and credit-only reputation computation CORC [29], peers contribution is tracked and is considered as its reputation.

For each transaction, the following information can be gathered for each peer:

1. The type of trust information:

   - The quality of the transaction: positive, negative or both (e.g., satisfied and unsatisfied transactions in EigenTrust, complaints for negative transactions in BinaryTrust).
   - The quantity of the transaction: the size of the files downloaded as positive for authentic files or negative otherwise (e.g., satisfactory downloads and uploads in IDA and MDA).

   Relying only on positive transactions will lead to dealing with malicious peers that conducted few successful transactions. While relying only on negative transactions may eliminate good peers. A combined approach is more efficient in identifying the real behavior of the peers.
2. A trustworthiness value which represents the outcome of the interaction. This value can be a binary value 0 or 1 (e.g., XREP, Travos, CredibilityRecords) which means that a requester peer is satisfied from the transaction or not, the provider peer is trusted or not trusted. The trustworthiness value can also be a discrete value (e.g., Excellent, Good, Fair and Poor) as in Amazon and DistributedTrust, a scaled integer (e.g., 1–5) (e.g., MDNT) or a real value on a continuous scale from [0,1] (e.g., Nice, PeerTrust), a probability value (for EigenTrust and PowerTrust). While a binary value does not allow partial trust, a continuous value expresses better how much trust is given. However, assigning $\{0,1\}$ is much easier. The trustworthiness value can be the result of a transaction (e.g., a cookie which is a signed receipt of successful transactions in Nice) or computed based on either the quality or the quantity of a transaction or a combined approach. In DCRC and CORC, peers' willingness to share the content they have and forward queries in addition of staying connected in case they are chosen to upload the requested file is considered for computing the peers' score. In some reputation systems, a value of 0 from [0,1] does not make a distinction between a malicious peer and a new comer to the system.
3. The time of the interaction: this can be used when computing reputation values (e.g., FuzzyTrust, Regret, MDNT, FineGrainedTrust).
4. The context of the interaction: the context of the interaction can differ from an application to another. An application may have different contexts. In the file sharing application, we assume to concentrate on one context which is providing authentic, not corrupted files.

Peers may keep track of all the records with every trusted peer (e.g., eBay) or only one record that summarizes all the transactions (e.g., TrustModel). Adopting this approach will reduce the storage cost. Keeping track of trust data will allow to benefit from the followings [15]:

- Recency: giving more weight to the recent values of the trust values. In this case, the use of *time of interaction* is important.
- Trend: it is preferable to know not only the trust value at a specific time slot but also the trend line over the last few time slots. Understanding the trend is important.
- Cyclical poor performance: keeping history of trust values according to their time slots may help in detecting such behavior.
- Variability in trustworthiness: it is important to know the variance value to detect the variability of the service provided.

It is important to note that the focus is more on the trustworthiness of peers rather than files since malicious peers can generate a large number of inauthentic files. Thus, identifying and isolating malicious peers will improve significantly the quality of the shared files. In [18], a reputation scheme is proposed for both peers and resources (e.g., files) to overcome the limitations of reputation schemes that tackle only the problem of peers' reputation. In the XRep protocol, each peer maintains two experiences repositories to keep track of authentic and inauthentic resources in addition to good and malicious peers, the peer had direct transactions with. *Credence* is another reputation scheme for identifying content that is not authentic (e.g., damaged, corrupt, missing contents, dangerous content, misleading metadata designed to confound user searches). However, computing reputations for files is usually time consuming when dealing with a large collection of files.

4.1.2 Where to Store Trust Information?

Distributing the trust data history is a challenging task [47]. Provider peers may change their pseudonyms easily to erase prior history. Some mechanisms can be deployed to avoid this by imposing an entry fee (e.g., Nice) or by using real names. However, the anonymity of peers may be affected. But, it is also important to protect the identities of the peers from malicious peers.

While in centralized P2P systems, the central entity will be used to store trust information (e.g., eBay, amazon), in completely decentralized P2P systems, the trust information regarding a trusted peer can be stored at:

- The trusting peer's level: in this case, it is necessary to contact all the trusting peers that have interacted with the trusted peer to compute the trusted peer's reputation. This task is very challenging in completely decentralized systems where trust information is scattered throughout the network (e.g., CredibilityRecords).
- The trusted peer's level: in this case, trust information is gathered at the trusted peer's level (e.g., EigenTrust, Nice). This makes the reputation computation process easier, however, malicious peers may manipulate this information. A

solution for this problem is to store the trust information at another peer's level. In DCRC and CORC, each peer is responsible of storing its own reputation for fast retrieval. To prevent malicious peers from thwarting the reputation system, reputation scores are signed by the RCA private key. To avoid that trusting peers drop negative reputation scores resulting from downloading files, the RCA keep the negative scores until these peers send credits for contributing to the system.

- Both the trusting peer's and the trusted peer's levels: since negative transactions may be deleted from the trusted peers records, the trusting peers keep this information (e.g., Nice).
- Third-Party peers: trust data is stored at another peer instead of the trusting or the trusted peers. If the trusted peer is responsible to store its reputation data, it can easily manipulate it. To prevent against such threat, another peer is selected to be responsible for storing this data. Another problem that may occur is that this intermediary peer may report false trust information. To solve this issue, a set of peers is responsible to store trust information of each peer. This way, minimizing the impact of false trust reports. As an example, in secure EigenTrust, a distributed hash table is used to assign to each peer, score managers that are responsible to store and compute peers' global trust values. Also in BinaryTrust, the storage of the output of negative transactions is realized by other peers.

Since the local trust information stored at each peer's level in completely decentralized P2P systems is proportional to the size of the network, the storage cost increases linearly as the number of peers increases. Trust management schemes should be able to minimize the storage cost.

In partially decentralized P2P systems, trust information can be stored at the supernode level and sent regularly to the trusted peer (e.g., IDA). To avoid tampering with this data, reputation information is encrypted and signed by a shared key known only to supernodes.

When storing reputation data at peer's level, reputation-based P2P systems need to provide mechanisms to ensure that reputation data is stored securely to prevent malicious peers from thwarting the reputation system and also to ensure the availability of trust information even with the transient nature of peers in P2P systems.

4.1.3 Is the Local Trust Information Sufficient?

At this stage, it is important to know if it is sufficient to choose the trusted peer according to local personal trust information or maybe it is necessary to get feedbacks form other peers in the system. In case where a few well-behaved peers know each other and always exchange files between them, local trust information is sufficient for making decisions. In systems where millions of peers are available to share files, local trust information is very limited. Therefore, relying on external advice from other peers is imperative. Reputation systems help in predicting peers' future behavior based on their past behavior. Since reputation is based on peers' opinions, the credibility of the peers must also be assessed.

4.2 The Reputation Query

To gather more information regarding the peers that have the requested file, the requester peer needs to ask other peers about their opinions. The requester peer queries about the requested peers' reputations in a specific recommendation context.

In centralized reputation systems, there is no need to gather reputation data from other peers since the information is already collected and stored at the central entity. In [29], the RCA facilitates the collection of trust data and the computation of reputation scores. In partially decentralized reputation systems, supernodes are used to collect reputation data (e.g., IDA, MDA). While in completely decentralized reputation systems, on demand processing of peers' reputation is needed by querying peers and collecting reputation data to compute peers' reputation.

In reputation-based systems, the reputation scheme collects, distributes and aggregates feedbacks about peers' past behavior. This scheme will help peers to identify trustworthy peers and decide whom to trust. However, for reputation systems to operate effectively is very challenging. These challenges will be detailed in the following sections.

4.2.1 To Whom the Requester Peer Should Send a Reputation Query?

In the physical world, when recommendations are needed for a person for example, it is not practical to send a query to any person. It is preferable to target the people who know better this person and ask for their feedbacks (e.g., co-workers, friends). The same process can be applied for the virtual world, however, in completely decentralized P2P systems, it is difficult to find the peers who interacted previously with the provider peer.

The followings can be the receivers of the reputation query:

1. The neighbors of the requesting peer: the requester peer can use its neighbors in the overlay network to get reputation information. For example, in the basic polling (P2PBasic) and the enhanced polling (P2PEnhanced) algorithms (P2PRep protocol), the requester peer broadcasts a reputation query, asking for the reputation of the provider peers. In LimitedReputation, the requester peer sends a reputation query to its neighbors (Neighbor-voting).
2. The peers that have already interacted with the requester peer and that are trustworthy: the requester peer will not send a reputation query to a malicious peer. Most probably, the malicious peer's recommendation is not credible. Another alternative in this approach is to ask the trustworthy peers to ask the trustworthy peers that they have interacted with (e.g., EigenTrust). Using this transitive trust, the requester peer gets more recommendations. In FuzzyTrust, recommender peers are chosen based on their number of performed transactions and local trust values. In LimitedReputation, the requester peer can also select recommender peers from its friends (Friend-voting), where friends are peers who have proved to be reputable.

3. The peers that have already provided the requester peer with accurate recommendations in previous queries (e.g., MDNT): the requester peer keeps track of past recommendations and uses metrics to identify good recommender peers. These recommender peers are knowledgeable peers and their opinions/ratings are accurate.
4. Specific selection of peers: as an example of social-based trust systems, Regret groups peers according to their relationships with the trusted agent. The most representative agent is selected as a recommender peer based on fuzzy rules.
5. Random selection of recommender peers: the requester peer can choose randomly some peers for their opinion.

4.2.2 How Many Peers Should the Requester Peer Contact?

The requester peer can send a reputation query asking for recommendations to the followings:

- A small number of peers: for example, a Time To Live (TTL) can be used to limit the number of queried peers and reduce the generated communication overhead. The forwarding of the query continues until the TTL is exhausted. However, with the TTL, the requester peer will not get a wide view of the peers' reputation, but the communication overhead will be reduced significantly. In case of Nice, a query is forwarded to only 5 users instead of following all the paths from the requester peer.
- A large number of peers: the more information gathered, the more precise is the reputation of the provider peers. However, this approach incurs additional communication overhead. Each forwarded message consumes bandwidth and processing at each peer it visits. This approach may impact badly the performance and the scalability of the system.

Sending a reputation query and asking for a feedback means that the requester peer is asking for peers' vote on the provider peers. Reputation-based systems can be divided into the following categories:

- Excessive voting systems: these systems require the voting process to collect and aggregate reputation values for each provider peer. This will incur high amount of message overhead and will introduce additional latency. Examples of these systems are: DistributedTrust, BinaryTrust, P2PRep, XRep and EigenTrust
- Moderate voting systems: these systems require the voting process in a moderate way. The voting process is restricted only to some of the peers in the network. This mechanism reduces significantly the traffic overhead and hence, using the network resources more efficiently. Examples of these systems are: the work done by Marti et al. LimitedReputation [36] and Selcuk et al. CredibilityRecords [51],
- No voting system: these system require no voting mechanism for computing reputation values. However, a high volume of traffic overhead is generated due the

use of a central entity for computing reputation values. As an example of these systems: DCRC and CORC in [29].

In general, there is a tradeoff between accuracy and communication overhead. The larger the number of feedbacks received from recommender peers, the more accurate peer's reputation is. The number of recommender peers is determined according to the requirements of the P2P application. In addition, if providing a feedback is manually performed and a rewarding mechanism is not provided, the system may not be able to collect sufficient number of feedbacks to compute reputation scores accurately.

The feedbacks sent to the requester peer include information regarding previous interactions with some of the provider peers. It is important to know what kind of information is conveyed in feedbacks. The content of feedbacks is needed for both credibility analysis and reputation computation.

4.2.3 What Kind of Information is Sent to the Requester Peer?

For each provider peer, local trust information stored at the recommender's peer level can be sent to the requester peer in addition to the following information:

1. A confidence value: this value confirms how confident the recommender peer is in the trust value.
2. the time of recommendation: this time is provided when the feedback is generated (i.e., eBay, Nice, MDNT, EigenTrust, Travos, Regret).

The confidence value is usually based on the number of interactions that occurred between a recommender peer and a provider peer. A small number of interactions indicates uncertainty and does not reflect the real peer's trust value, while a large number of interactions increases the accuracy of the provided trust information. The confidence value can also be a subjective value given by the recommender peer based on its own experience. The concept of the confidence value is similar to reviewing a submitted paper and getting a feedback from a reviewer who indicates explicitly its expertise in the field. This represents how confident is the reviewer in the review. In Travos, the confidence value is a metric that represents the accuracy of the trust value based on the number of transactions. If the requester peer has a low confidence value in its assessment of trust for a provider peer, then seeking recommendations from other peers is necessary while, having a high confidence value means that the requester peer does not need to ask for recommendations. In fact, it could be that the recommendations received will add more misleading information than giving more accuracy to the peer's reputation.

At the queried peers, the trust information is filtered and only transactions corresponding to the context of recommendation are chosen to be sent to the requester peer. In case that recommendations pass through intermediary peers (e.g., PeerTrust, Nice, BinaryTrust, MLE) [48], additional information regarding these peers may be added to ensure the transparency of this process. In case of TrustModel, each intermediary peer adds some comments on the credibility of the recommendation received.

The identity of recommender peers can be revealed or not. In P2PBasic, the identity of peers sending the feedback is not required, while in P2PEnhanced, the identity of these peers is included in the feedbacks. This way, the requester peer is able to identify the peer sending the feedback and gives a weight to its feedback according to the trust given to this peer.

In some of the proposed reputation systems, the history of all trust transactions for a specific context are sent to the requester peer. In other systems (e.g., EigenTrust), only an aggregated trust value is sent to reduce the communication overhead and provide better scalability, but it lacks transparency. Aggregating feedbacks is a difficult task. The reputation score must represent the real behavior of a peer. The aggregation method of the trust values was not clearly described in most reputation systems. In Nice and TrustModel for example, any aggregation method can be used by each peer according to its requirements or by all the peers in the system to ensure homogeneity and avoid conflict of interests. Different approaches have been proposed for the feedback aggregation method: in Regret, the weighted average of ratings is computed with the use of recency while, in EigenTrust, the difference between positive and negative transactions is normalized. In Travos, the feedback is an aggregation of the number of successful transactions and unsuccessful ones. The feedbacks received from recommender peers must provide sufficient information for credibility assessment. Incentive mechanisms are required to encourage peers to send recommendations.

4.3 Reputation Computation

Eliciting feedbacks from the peers that received the reputation query and had already interacted with some of the provider peers rises several issues. These peers may:

- Reply to the query honestly: sending the right feedback is the expected behavior from a well behaved recommender peer.
- Reply to the query dishonestly by lying: sending the wrong feedback in order to defame competitors or flatter conspirators. Reputation management schemes should be able to minimize the impact of this threat.
- Ignore the query: a free rider will ignore the need of the requester peer and will not take into account the reputation query. Ignoring the query by peers that have already interacted with the provider peers is an act of unreliability.

According to the reputation management scheme, the intermediary peers may transmit or not the feedback to other peers. Upon reception of feedbacks from other peers to get transmitted to the requester peer, the intermediary peers could omit the feedbacks or maliciously manipulate the content. Some security mechanisms are required to make sure that feedbacks are received intact by the requester peer. As an example, the recommender peer's digital signature can be added to ensure the recommendation integrity. Reputation management systems should provide mechanisms to enforce good contribution and cooperation from the peers that receive the reputation query.

The requester peer receives feedbacks from recommender peers that can be one-hop or multi-hop from the trusted peer or even all the peers in the system. Different types of opinions may result:

- First hand opinion: is a direct opinion from the queried peer to a provider peer. The queried peer has already interacted directly with the trusted peer.
- Second hand opinion: is an indirect opinion from a peer, that has been contacted by the queried peer, about a provider peer.
- Third hand opinion: is a public opinion

Different weights can be given to the received feedbacks according to their type. Typically, a first hand opinion is more trustworthy and then more important than the other ones.

This is one of the most important issues in reputation management systems and unfortunately, only few research works have focused on solving this problem. If the feedbacks received are not credible, the reputation can not be computed accurately. The majority of reputation schemes assume that peers are well-behaved, honest and do not provide any mechanism to check the accuracy of the gathered opinions which will reduce significantly the reliability of the reputation system. This problem is also referred to as mis-representation. In eBay, for example, the credibility mechanism is left to members who can go through all the ratings and decide if a specific member is credible or not. Recently, the credibility assessment becomes an important issue since in e-commerce transactions where money plays a vital factor, accurate reputations are necessary in making the right decisions.

It is imperative to distinguish between two important issues when dealing with liar and fraudulent recommender peers:

- Liar recommender peers: the recommender peer had actually a transaction with the provider peer. The recommender peer may reply negatively even if the output of the transaction was positive. The recommender peer may lie and provide a dishonest feedback for different reasons (e.g., to increase the reputation of a colluding peer).
- Fraudulent recommender peers: the recommender peer did not interact with the provider peer before and it provides the requester peer with a feedback. The feedback can be positive if the provider peer is a colluding peer or negative if the provider peer is a competitor (i.e., a good contributor peer).

Detecting malicious recommender peers is vital to the accuracy of the reputations and hence, the reliability of the reputation system. A credibility analysis is required to filter out recommendations, eliminate the suspicious feedbacks and select the accurate ones. The content of recommendations form the basis of reputation computation.

4.3.1 How to Deal with Liar Recommender Peers?

According to [30], there are two basic approaches that have been proposed: Endogenous and Exogenous methods. In Endogenous methods, the statistical properties of the reported feedbacks are used to detect unreliable feedbacks. In these methods, most of the proposed mechanisms assume that liar peers are the minority among peers and so the ratings that are different from the majority of peers are inaccurate. Exogenous methods rely on other information such as the reputation of the recommender or its feedback accuracy given its past recommendations or its relationship with the provider peer.

The following are some mechanisms that have been proposed to minimize the impact of liar recommender peers:

- Keeping track of past recommendations (e.g., number of accurate and inaccurate recommendations) and weight the feedbacks according to the credibility of the recommender peers (i.e., MDNT, Travos, MLE, CredibilityRecords).
- The multivariate outlier detection technique: this technique is used to detect liar peers in FineGrainedTrust [61]. Outlier detection is an important task in data analysis. The outliers describe the abnormal data behavior which means data that is deviating from the natural data variability.
- The use of suspicious transactions to measure the credibility of recommender peers [39]: a suspicious transaction is one in which the feedback sent by the requester peer is different from the one expected knowing the reputation of the provider peer. Peer's credibility is computed based on the ratio of the number of suspicious transactions over all the transactions performed by the peer.
- The use of trust and reputation values as credibility metrics for the recommender peers (i.e., EigenTrust, Nice, FuzzyTrust, P2PEnhanced). Trustworthy peers are considered as credible while untrustworthy peers are not. This is based on the fact that if peers are behaving correctly by sending authentic files, these peers will most probably be honest in their recommendations. In Regret, social relationships are also taken into account.
- Redundancy: the use of different score managers to compute the trust value and use a majority vote to eliminate the false reports by malicious score mangers (e.g., EigenTrust)
- The use of Beta distribution based on previous recommendations as in Travos
- To let only one rating count from any single IP address
- Enforcing policies: in eBay, when a seller receive multiple feedbacks from the same buyer within the same week, the net effect on that seller's feedback score is based on the number of negatives, neutrals and positive received [22].

Many reputation systems (e.g., EigenTrust, fuzzyTrust, P2PEnhanced) use a reputation value for a recommender peer as a credibility value which is not totally accurate since some peers could behave well while sending authentic files and provide at the same time inaccurate recommendations. The credibility of recommender peers is based on local trust values. These values are not accurate enough to measure the trustworthiness of recommender peers and it is not practical to collect feedbacks regarding the credibility of the recommender peers. This approach will incur additional overhead and waste of network resources. Moreover, if a peer is malicious according to its reliability in sending authentic files (i.e., its reputation) it will be difficult to trust its recommendation and there is no guaranty that a reputable peer will provide an honest opinion.

4.3.2 How to Deal with Fraudulent Recommender Peers?

The followings are some mechanisms that have been proposed to minimize the impact of fraudulent recommender peers:

- Only peers who were actually involved in a transaction are permitted to provide ratings (e.g., eBay).
- The use of proof of interactions: in TrustMe for example, both entities participating in a transaction sign a transaction certificate. This proof is needed to report the output of the interaction.
- The use of proof of processing: in DCRC and CORC, the RCA sends proof of processing for peers contribution to the system by processing, forwarding queries, staying online and serving files. To prevent malicious peers from getting credits without actually participating, the RCA maintains a transaction state where the credit_processed_list contains the list of peers who have already received the credit. The RCA ensures that a peer will collect credit only once for the same upload.

It is important to indicate that even with the presence of proof of transactions in some reputation management systems, it is difficult to prove that effectively there was a transaction between the two parties. In case of a file transfer for example, it is needed to prove that the file transfer has actually occurred. This proof must also indicate all information regarding the file transferred (e.g., size), the peer ID uploading the file, the peer ID downloading it and the time of interaction.

4.3.3 How to Compute the Reputation?

After filtering the feedbacks and getting rid of dishonest reports and fraudulent ones, the gathered data from different recommender peers is used for reputation computation in addition to the local trust data available at the requester peer.

Some of the proposed reputation schemes assign more importance to:

- Bad transactions versus good transactions: in some reputation schemes, it is proposed that the negative impact of bad behavior on reputation should outweigh the positive impact of good behavior.
- Local trust data versus the reputation information gathered from other peers: since the local trust data is more credible than the data collected from other peers.

However, in P2P systems with millions of users, local trust data may not be very helpful in taking decisions.

- Recent transactions versus old ones: to take into account the recency of interactions, the time of interaction is needed. This will help in identifying the traitors. In some reputation systems, old transactions are deleted and the focus is more on the recent transactions. In DCRC, a time stamp is used each time the RCA sends the reputation scores to peers, while, CORC time-stamps the reputation scores for expiration.

In some proposed reputation systems, a peer's reputation is equivalent to the group of peers it belongs to. A group's reputation can be, for example, the average of all the members reputations within this group. If the group has a high reputation, all its members are reputable. A peer can be rejected from the group in case it turned out to be acting maliciously.

Reputation systems should be exigent in terms of accuracy of the computed reputation according to the risk involved in the transactions. However, several factors affect the accuracy of reputation values such as: network overheads, congestion and loss of trust data during peers' communication. In some circumstances, this inaccuracy may be acceptable since the goal from reputation systems is to provide an approximation of the real peer's behavior.

Different approaches have been proposed used to aggregate trust values received from recommender peers and synthesize them to generate a reputation value for a provider peer [15, 30]:

1. Deterministic approach: In this approach, peers' reputation is based on a simple summation or average of collected ratings. Even in the most popular e-commerce applications that involve a huge amount of money, reputation values can be computed easily by simple operations. The reputation scheme used in eBay, for example, is based on the sum of the number of positive and negative ratings, while in amazon the reputation is computed based on the average of all the ratings. A weighted average of all the ratings is also possible based on different factors such as the credibility of recommender peers. This approach is easy to use and can be easily understood by users. Many proposed reputation systems used this approach for reputation computation. In MDNT, the reputation value is computed based on the weighting factor for the first, second and third opinions of third-party agents, the reputation opinion given by a recommender agent, the credibility of the recommender agent, and the time weighing factor that represents the importance of the opinion depending on the time of the last interaction between the recommender agent and the provider agent. In PeerTrust [59], different factors are taken into account such as transaction and community factors. In DCRC, reputation scores are computed based on Query-response Credit, the Upload Credit, the Download Debit and the Sharing Credit. In BinaryTrust, reputation is computed based on the number of negative complaints.

2. Probabilistic approach:
    a. Bayesian Networks: The Bayesian approach uses a probabilistic approach to the determination of the reputation. It is based on the Bayes formula. Bayesian systems take binary ratings as input, and are based on computing reputation scores by statistical updating of beta probability density functions (PDF). Several works have used this approach including [41], [58], and [62]. In Travos, reputation is computed based on Beta Probability Distribution. However, the Bayesian approach suffers from strong assumptions of independence that are made.
    b. Maximum Likelihood Estimation: MLE [20] uses a probabilistic approach to compute the reputation value based on the probability of recommender peers to provide inaccurate information. In MLE, the reputation value is the probability of a peer to cooperate and it is chosen to maximize the probability of the available ratings.
3. Fuzzy logic: Trust and reputation can be represented as linguistically fuzzy concepts. The information received from recommender peers is characterized by being imprecise and not accurately quantified. Fuzzy systems are used to deal with such situations by providing rules for reasoning with fuzzy measures. Different factors (e.g., the credibility of recommender peers) can be represented by fuzzy sets and membership functions are used. Several works have been proposed to compute reputation scores based on fuzzy logic systems including [50, 53]. Regret considers the following dimensions in reputation computation: the *individual dimension* which is the direct trust obtained by previous transactions, the *social dimension* which refers to a trust of an agent in relation with a group and finally the *ontological dimension* that includes the particularity of each agent. In FuzzyTrust, fuzzy inference is used to produce local trust values and aggregate them to global reputation values.
4. Flow models: Systems that compute trust or reputation based on transitive iteration through looped or arbitrarily long chains. As an example, EigenTrust, and FineGrainedTrust.

4.4 The Use of Reputation

4.4.1 How to Choose a Peer Based on the Reputation Value?

After computing the reputation value, the following approaches can be adopted:

- The ranked-based approach: peers are ranked according to how reliable they are likely to be. The more reliable they are, the more trusted they are, the more the requester peer is expecting to get the required file. In this approach, the peer with the highest reputation value will be selected to upload the file. Highly reputable peers will handle almost all the uploads, yielding to an increase in their reputation. However, these peers will be overwhelmed by download requests. This approach suffers from unbalanced load share among reputable peers. This approach has been adopted by EigenTrust and Travos.

- The threshold-based approach: The computed reputation scores may be compared against a trust threshold value. A random peer from the provider peers whose reputation values are greater than this trust threshold is selected. This approach spreads the load between these peers. This approach has been adopted by Nice, MDNT, MLE, and BinaryTrust.
- The Probabilistic-based approach: choose the peer that will upload the file probabilistically based on its trust value. This probability is proportional to its normalized trust value and with a probability of *p%* select a new comer (e.g., 10% for EigenTrust). This approach distributes better the load share among reputable peers, giving them a chance to increase their reputation in addition to increasing the reputation of new comers.

Once the requester peer finds a provider peer and it is willing to download from it the requested file, the requester peer becomes the trusting peer and the provider peer becomes the trusted peer. Both the trusting and the trusted peers will participate in a transaction.

4.4.2 How to Evaluate a Transaction After Downloading a Requested File?

It is important at this stage that a peer is able to ascertain when a transaction is successful. In case of failure, it could be due to network congestion or network failure, and sometimes it is due to the maliciousness of the transaction partner. Assigning a trust value based on the quality of the transaction is subjective. The subjectivity of trust values is inherent in most trust systems.

According to a detailed study on eBay, recommenders peers may not send negative feedbacks by fear of reprisals or may also tend to reciprocate in both a positive and a negative way. To deal with the problem of reprisals, some reputation systems provide mechanisms such as:

- A peer can have an identity as a regular peer and another identity as a recommender peer. Recommender peers will not be easily recognized.
- Recommender peers can remain anonymous as in BinaryTrust. Recommender peers fill complaints and their identity is not provided. In this case, keeping records for assessing the credibility of recommender peers may not be possible. However, the identity of the mediators storing the recommendations are known.
- The privacy of recommender peers is protected by using symmetric-key cryptographic functions as in FineGrainedTrust.

4.5 Credibility Assessment

After assessing the transaction output, it is imperative to update the credibility of recommender peers. In MDNT, the credibility of the recommender peer is computed based on the distance between the transaction result and the trust value provided by this recommender peer. Updating the credibility of recommender peers will be beneficial for the assessment of feedbacks in future transactions. In Travos, the credibility of recommender peers in terms of accuracy of the provided recommendation is updated according to the transaction output. In MDA, the feedback sent to the supernode of the trusting peer is used to check if the transaction is considered suspicious or not. The credibility of the trusting peer is updated accordingly. This credibility will be used in computing the reputation of the trusted peer.

In completely decentralized P2P systems, credibility values could be stored at the trusting peer's level for easy access in case this information is needed. In partially decentralized systems, peers' credibility could be also stored at the supernode level. MDA counts the number of suspicious transactions to measure the credibility of peers.

4.6 Incentives, Rewards and Punishment

In P2P systems, if all peers receive the same service regardless of their behavior, peers will not be motivated to strive for high reputation values since they will be always asked to upload files without receiving any special benefit or reward. This is why service differentiation is needed.

The system may reward reputable peers with increased connectivity to other peers, greater bandwidth, and/or higher priority/probability of performed requests. Rewarding these peers will give users an incentive to get a high reputation value by sharing authentic files. Peers can also be rewarded for providing feedbacks and sharing trust data with other peers. Several incentive mechanisms have been proposed to motivate peers to contribute to the system [23, 24, 27, 43–45, 60].

While good peers are rewarded for exhibiting a good behavior, malicious peers can be punished by sending them lists of peers with low reputation values to download from them. In [43], the authors introduce a reputation-based mechanism that assigns better service to higher performing peers. The reputation-based policies are classified into two dimensions: *Provider Selection* and *Contention Resolution*. In *Provider Selection* policies, a peer among peers providing a service is chosen to provide the service. Three schemes have been proposed for *Provider Selection*: *Highest Reputation* where peer with the highest reputation is selected, *Comparable Reputation* where peers can request services only from peers with reputation values comparable to their reputation and *Black List* where peers with low reputation are not providing any service. However, in *Comparable Reputation* policy uses the concept of "Layered Communities" and provides the requesting peer with a list of peers having similar reputation values. This approach will incur an important increase of malicious uploads. Indeed, if a peer receives a service from a lower reputable peer, it will most probably receive a bad service (e.g., malicious file) and hence does not help the peer in providing good service to others. *Contention Resolution* policies help in selecting a peer among all peers requesting a service from the same provider peer. Two policies are presented: *Highest Reputation* policy where the peer with the highest reputation is selected to get the service and Probabilistically Fair w.r.t Reputation policy where a peer is selected with a probability according to its reputation.

In [45], the authors analyze the effectiveness of different incentives mechanisms to motivate peers to share files. The authors present a *reputation-based peer-approved* scheme. The scheme uses a reputation mechanism based on rating peers according to the number of files they are advertising. Peers are allowed to download files only from peers with lower or equal rating. The results show that the scheme can be used to counter the selfish behavior. However, this scheme will allow malicious peers to advertise a high number of corrupted files. According to this scheme, these peers will still receive good service. Even non malicious peers may advertise a large number of non popular or useless files and still benefit from the system.

In [60], the authors propose a reputation scheme that combines trust and incentive mechanisms. The proposed scheme uses explicit and implicit evaluations such as files' vote and retention time, download volume and users' rank to construct direct trust relationships. Based on the reputations, service differentiation is used to motivate users to share, vote on files, rank users and remove fake files. However, performance evaluation is needed to assess the performance of the proposed scheme.

Since the peers differ from each other in the type of services and resources they contribute to the system, [27] proposes a Service Differentiation Protocol (SDP) for service differentiation in completely decentralized unstructured P2P networks. This protocol works as follows:

- During the *search* phase, a peer sends its reputation score along with the Query message. Each peer that receives this query extracts the reputation score and maps this value to a Level of Service (LoS). This peer will provide this LoS to the requester peer.
- During the *content download* phase, the peer requesting the file sends its reputation score to the peer uploading the requested file. This peer will send the file with a rate of transfer according to the reputation score of the requester peer.

In [38], the contribution of peers rather than the reputation of peers is used as a guideline for service differentiation. Peers' contribution is based on their availability to share files and their involvement in the system (i.e., upload of authentic files).

In [34], the authors propose a service differentiation based on the amount of services each node has provided to a P2P community. A resource distribution mechanism is proposed to increase the utility of the whole network and provide incentive for nodes to share information. A generalized mechanism that provides incentives for nodes having heterogeneous utility functions is also described.

5 Design Requirements

Reputation systems should have minimal overhead in terms of infrastructure, computation, storage and message complexity. The design of reputation management schemes must consider the followings issues [30]:

- Enforcing Local Control: as trust data can be stored by peers in the system, it is important not to allow these peers to arbitrarily manipulate trust data. Local control mechanisms are needed to protect trust data.
- Minimizing Storage Cost: this can be achieved by minimizing the trust data that need to be stored at peers level.
- Minimizing Bandwidth Cost: this can be realized by minimizing the messages exchanged between peers.
- Fault Tolerance: the topology of P2P networks is changing frequently due to the transient nature of peers that join and leave at any time. Leaving the system without any notice, may result in unavailability of trust information stored at peers level. Trust information need to be replicated to assure having the required information.
- Scalability: scalability means the ability of the reputation scheme to scale with an increase in the number of peers. This increase will yield to an increase of transactions among the peers. Peers will need to handle more trust data by collecting, storing, generating more computations and replying to queries. To investigate the scalability of a reputation system, it is needed to address the load placed on peers, especially on highly reputable, due to high demand for upload transactions and on the network as a result of exchanging messages between peers.
- Reliability: reputation systems should provide mechanisms to protect peers from malicious threats. Based on peers' reputation, peers can be identified as good or malicious. Peers need to rely on reputation systems to efficiently identify malicious peers and isolate them from the system. The more reliable reputation systems are, the more trust is given by peers to these systems.

6 Centralized Reputation Systems

6.1 e-Commerce Applications

The Internet opened up new opportunities for millions of people to interact with each other through applications such as electronic markets. However, in e-Commerce sites, customers do not have enough information about the sellers and the products/services offered. Trust systems create a platform between different parties to learn from each other and build trust [19, 47].

eBay [1] uses the feedback profile for rating sellers and computing the sellers' reputation. Buyers rate their transaction partners with a positive or negative feedback, and explain why. The reputation is computed by assigning 1 point for each positive comment and –1 point for each negative comment. The reputation score for a seller is the sum of the received ratings.

The reputation system used in eBay has contributed significantly to increase eBay's revenues. eBay recognizes that the collected ratings to compute their members' reputations are one of the company's main assets. eBay relies on a central trusted server to maintain the reputation system that has been improved recently. For members identity, eBay members use pseudonyms and personal information is used to identify members and kept confidential by the system. No special benefits are given to new users to increase their reputation.

The eBay feedback system suffers from the following issues:

- Easy to attack
- A seller may have a good reputation value by satisfying many small transactions even if he did not satisfy a transaction with a higher amount
- No mechanism is used to detect liar members that send wrong feedbacks. This task is left to users to detect such behavior.

eBay's reputation concept has been widely used to increase the trust level of online users. Similar reputation systems have been proposed in e-Commerce applications [15]:

- BizRate.com: is a shopping search engine which lists stores and products. BizRate.com has an index of over 30 million products provided by more than 40,000 stores. BizRate.com claims that they use feedbacks collected from more than on million online users each month. BizRate.com uses ShopRank which is a proprietary shopping search and rating algorithm. This algorithm weights up price, popularity and availability of products against the reputations of sellers. BizRate.com provides:
  - Rating of products: a rating scale of 1–5 stars (awful, poor, average, very good, excellent). In addition a breakdown of the overall product rating is also available (e.g., ease of use, portability, sound quality) and the reviews provided by users. Based on this information, a user can decide to buy or not the product.
  - Rating of merchants: for each product, a list of stores that sell the product, along with the price, the availability of the product and the rating of the store. This rating is based on a smiley scale (poor, satisfactory, good, outstanding) in addition to a "Customer Certified" logo for stores who satisfy specific criteria. This logo increases the users' trust in these stores. Only collected data during the last 90 days is considered for computing the merchants' reputation.

- Amazon.com: started as online bookstore and expended to sell different kind of products. Amazon.com uses ratings for products and for merchants. In merchant rating, sellers are rated based on the quality of the service provided on a scale from 1 to 5 stars in addition to comments left by buyers. In addition, the "Safe Buying Guarantee" increases the buyers confidence and encourage them to trust the sellers. Products in Amazon.com are also rated by users. Electronic products are only rated while other products are rated and reviews are provided. Users check reviews and vote whether the reviews are helpful or not.

- Elance.com: is a web-based project marketplace that provides small businesses easy access to projects such as: web design, development, software, administrative services, writing, etc. Service providers are rated by service buyers while service providers leave only comments to service buyers.

  - Rating of service providers: each project is rated by the service buyer according to different criteria (e.g., quality of work, responsiveness, professionalism, subject matter expertise, adherence to schedule, adherence to work). The rating is based on a five-point scale from *Extremely Unsatisfied* to *Extremely Satisfied*. In addition to this rating, the service buyers can leave comments to describe the service provider's performance. For each criterion, a numerical score is computed for all projects. The reputation of a service provider is an overall numerical score based on the projects scores
  - Service buyers: Service providers can leave only comments to service buyers and no rating is provided

- MoneyControl.com: is used to track the stock market. People can read different posted opinions and rate them. These opinions inform the readers about the stock market. These opinions help the readers make informed decisions regarding the stock they want to invest in. Opinions given by users are rated on a five stars scale. The average of all the ratings provided by users for a specific opinion is its overall rating and is expressed by a number between 0 and 5. Rating of reviewers is also available. A user can track the users who always provide good opinions. These users form the trusted agents network. A reviewer's reputation is based on the number of the trusted agents who track this reviewer.
- Yahoo.com: provides a variety of products to customers. Both rating of products and merchants are available. In product rating, users can rate the products in addition to ratings provided by experts. Customers prefer more this latter rating since the expert rating is more trusted. Merchants are also rated based on the feedbacks collected from customers. Customers who made the purchase and rated the merchant are given more weight than other users.
- Alibris.com: provides information regarding sellers of books, music and movies. It offers access to more than 50 million books. Both merchants rating and product rating are available. Merchants are rated based on the number of Alibris orders delivered by the seller. The rating scale is based on 1–5 stars and new sellers belong to the 4 stars category during the first 90 days. Products are not rated based on the content but on the quality of the products since Alibris.com is used to sell used products.
- CNET.com: provides an up-to-date information regarding technology products. Ratings of products is realized by editors. These editors are experts and their opinion is more trusted by customers. Editors use the products and rate them according to different criteria (e.g., set up, design, features, performance, service and support) according to the type of the product (e.g., camera, computer). To

compute the quality of a product, a specific weight is given to each criterion. The rating generated is a numerical score between 1 (Abysmal) and 10 (Perfect). Users can also provide their opinion regarding a specific product to exchange information between each other. Ratings for merchants is also provided based on four criteria: site functionality, store standards, order fulfillment, and customer feedback.

The feedbacks received from online users represents a rich set of collected information (e.g., ratings, comments, opinions, feedbacks) that helps the business managers in discovering and identifying the needs of customers and improving the services provided to them and hence, increasing the customers' trust in the business. As a consequence, the profit generated is increased in addition to the business reputation and value. The feedbacks received strengthens the relationship between customers and the business in addition to providing to online users the information needed to reduce the risk involved in virtual environments. While in physical environments, customers can really feel the products before buying them, exchanging information by sending feedbacks allow customers make informed decisions.

6.2 P2P Systems

Although these P2P systems use a distributed overlay for search and control messages, a centralized reputation system is used.

6.2.1 Reputation Management Using DCRC and CORC

In [28, 29], the reputation system uses objective criteria to track peers' contribution in the system. Each peer stores the reputation value locally for a fast retrieval. Two mechanisms are proposed to compute the reputation:

- The Debit Credit Reputation Computation (DCRC)
- The Credit Only Reputation Computation (CORC)

By using a Reputation Computation Agent (RCA), peers' reputations are updated periodically in a secure and distributed manner. The DCRC mechanism credits peers for serving content and debits for downloading. The second mechanism (CORC) credits peers for serving content with no debits for downloading. The two mechanisms credit peers for query processing, query forwarding and for staying online. RCA is responsible for computing peers' reputation and certain amount of accuracy is lost due to the periodic update of reputation scores and the loss of data during the communication with RCA. Peers can be differentiated according to their *behaviors* and their *capabilities*. The behavior of a peer depends on its contribution to forwarding, processing requests and uploading files. The capability of a peer depends on the processing power, memory, bandwidth and storage capacity.

In the DCRC scheme, a peer's reputation score is computed using the following components:

- Query-Response Credit: peers are credited for being online and processing query-response messages.
- Upload Credit: each peer uploading a file gets a credit for serving the file and contributing to the good functioning of the system
- Download Credit: each peer downloading a file, will get a debit
- Sharing Credit: a peer gets credit for sharing hard-to-find files.

The CORC mechanism is similar to the previous scheme except that the download will not be debited. This score will only increase. In order to prevent peers from acting maliciously once they get a high CORC score, CORC time stamps the reputation scores for expiration.

Drawbacks:

- RCA is contacted periodically by peers for updating peers' reputation scores. This is potentially a central point of failure.
- RCA needs to be replicated to ensure system robustness. No schemes have been provided to explain replication mechanisms.
- Malicious downloads are not taken into consideration in peers' reputation.

In [28], two methods for tracking reputation are proposed: *Strong Reputations* and *Weak Reputations*. In *Strong Reputations*, the RCA is expected to have a copy of all the content served by peers to ensure content reliability. In addition, the RCA crawls periodically the P2P topology to maintain snapshots of the topology. This assumption is not practical in real P2P systems even with the fact that RCA is not required to serve the content. This proposed scheme for reputation tracking incur higher overheads compared to *Weak Reputations*.

6.2.2 A Fine-Grained Reputation System for Reliable Service Selection: FineGrainedTrust

In [61], the authors propose a reputation system built upon the multivariate Bayesian inference theory. Reference [61] defines the reputation of a server (i.e., file provider, peer) as the probability that he is expected to demonstrate a certain behavior, as assessed by a client based on self experiences (i.e., the output of direct transactions) and users' feedbacks.

For reputation management, a centralized server is used as:

1. Account Manager: crediting/rewarding users and maintaining social groups.
2. Query Processor: for reputation queries
3. Feedback Collector: collects feedbacks
4. Reputation Engine: computing reputation scores

FineGrainedTrust has the following characteristics:

- Reputation data: After each transaction with a server, the user will increment the corresponding Quality of Service QoS level by one. This way, each user keeps only one vector for each server he interacted with.
- Reputation data storage: QoS information is stored across system users either in a random fashion or through a distributed hash table. In this latter case, the central server let λ users store the reputation feedback to improve system tolerance in case users are not available. The larger this value, the higher fault tolerance, the larger the communication overhead and the average storage cost.
- Reputation computation: Users keep QoS experiences with servers after each transaction. They return QoS experiences after receiving a query from the centralized server. Users may inquire the centralized server about servers' reputations. Collecting all QoS information by the centralized server simplifies significantly the computation of peers' reputations. Old experiences are not as important as recent ones.

Drawbacks:
The centralized server is a single point of failure and is easy to attack (c.f. Section 3.4.3).

7 Decentralized Reputation Systems

Since decentralized P2P applications are characterized by the absence of a central authority that coordinates the reputation management. Peers store information about past experiences with other peers and may be required to get other peers' opinions for reputation computation. Several decentralized reputation management schemes have been proposed for completely decentralized systems [13, 17, 20, 32, 36, 42].

7.1 The Distributed Trust Model: DistributedTrust

Abdul-Rahman et al. [11] proposed a model for trust based on distributed recommendations. This work is one of the first works on distributed trust models and that can be used in P2P systems. The proposed approach is based on four goals:

- Decentralization: each agent is responsible for managing trust information.
- Generalization of the notion of trust by using trust categories and trust values for different levels of trust in each category.
- The use of explicit trust statements in order to reduce ambiguity.
- Recommendations are used to get trust information regarding other agents in the system.

In this trust model, a trust relationship is between two entities, is non symmetrical and is conditionally transitive. The model has two types of trust relationships:

- Direct trust relationship: when an agent has trust in another agent.
- Recommender trust relationship: when an agent trust another agent to give recommendations about another agent's trustworthiness.

In this approach, each agent stores its trust data regarding other agents. Key-based encryption is used to protect recommendation messages from malicious agents. Network traffic is generated only for recommendations in case that the agent has not a direct trust value for another agent in the system.

Drawbacks:

- Each agent needs to store all history of past experiences and received recommendations. Storing this information will provide for the user a kind of global view of the whole network, however, a large storage capacity is needed.
- Updating this information can be time consuming and difficult.
- Increase of network traffic due to message exchange between agents in order to get reputation information.

7.2 The Binary Distributed Trust Model: BinaryTrust

In [13], the trust model is based on binary trust. An agent can be trustworthy or not. Each transaction between two agents can be either performed correctly or not. When an agent cheats, this agent becomes untrustworthy and a complaint is sent to other peers. In [13], only dishonest transactions are considered, based on the fact that malicious behavior is the exception. Reputation of a peer is based on the global knowledge of the complaints. PGrid that is a data storage, is used to store complaints. This trust model works as follows:

- When a peer file a complaint about another peer, the peer will send complaints to other peers using *insert* messages
- When a peer wants to get the trustworthiness of a peer, the peer will search for the complaints on that peer and will identify the peers that store the complaints. In order to avoid additional traffic overhead, once the peer requesting the trustworthiness of another peer receives similar trust information from a certain number of peers, there is no need for further search.

Drawbacks:

- Some peers may receive complaints about themselves. Conflict of interest may occur and peers can delete this information to drop the complaints.
- No mechanism is provided to prevent from inserting arbitrary complaints about peers
- Maintenance of PGrid is required.

7.3 Reputation Management by Choosing Reputable Servents: P2PBasic and P2PEnhanced

In [17], the distributed polling algorithm *P2PRep* is used to allow a servant *p* (i.e., the resource requester) looking for a resource to ask about the reputation of offerers (i.e., resource providers) by polling peers. After receiving a response from all the resource providers available to provide peer *p* with the requested resource, peer *p* selects a set of peers from the offerers and broadcasts a message asking other peers to give their opinion about the reputation of the offerers. Two variants of the algorithm are provided: in *the basic polling*, peers send their opinion and peer *p* uses the vote to determine the best offerer. In *the enhanced polling*, the peers provide their own opinion about the reputation of the offerers in addition to their identities. This latter will be used by peer *p* to weight the vote received.

Credibility Management is considered in *the enhanced polling* algorithm. Trust data of the peers sending their opinion to the resource requester will be considered in computing the reputation of the resource providers. Trustworthy peers are given more weights than untrustworthy peers in the reputation computation. In *the basic polling* algorithm, credibility of peers is not taken into consideration.

Drawbacks:

- The proposed schemes incur considerable overhead by polling peers for their votes. The *basic polling* algorithm checks whether the voters have provided the vote by sending *TrueVote* and *TrueVoteReply*. In the *enhanced polling* algorithm, *AreYou* and *AreYouReply* messages are used to check the identity of the voters. This will further increase the network overhead.
- Each peer has to keep track of past experiences with all other peers. The reputation of the peers is used to weight their opinions.

In [16], the authors propose SupP2PRep which is a protocol for reputation management via polling in P2P networks with superpeers. In [16], when a servent is looking for a resource, it broadcasts a *Query* message and receives a list of provider peers. The requester peer polls its peers by broadcasting a message *PollRequest* requesting their opinion about the selected provider peers. These peers reply with a *PollReply*. The superpeers collects the *PollReply* messages into one message *CumulativePollReply* and then sent to the requester peer. This proposed protocol incurs additional messages overhead for direct communication between peers. No performance analysis nor performance evaluation are presented.

7.4 Reputation Management by the XRep Protocol: XRep

In this paper [18], the authors propose an approach that uses combined reputations of servents and resources. The authors describe the XRep protocol used for maintaining and exchanging reputations and its advantages against security attacks in

P2P systems. In XRep, each servent maintains information based on its own experiences on resources and servents and can share this information with other peers. Each peer maintains two experience repositories:

- A resource repository: for storing resources identifiers and a binary value to describe each resource if it is good or bad
- A servent repository: for storing the servent identifier for each peer it interacted with, in addition to the number of successful and unsuccessful downloads.

The XRep protocol is based on the following phases:

- *Resource searching* phase: search the peers that have the requested resource.
- *Resource selection and Vote polling* phase: select the resource from the received list. The requester peer broadcasts *Poll* messages to enquire about the reputation of the resource and the peers providing this resource.
- *Vote evaluation* phase: evaluate the reputation of the requested resource based on the received votes.
- *Best servent check* phase: check that the best servent provides really this resource and that the resource digest is in fact reliable.
- *Resource downloading* phase: decide from which servent to download the resource and contact this servent directly for the download operation.

Drawbacks:

- Inquiring about the reputation of resources and the resource providers incurs considerable traffic overhead.
- No performance results are provided to convince the reader that the reputation management based on both the resources and the servents outperform the schemes that are based only on the reputation of the servents.

7.5 Reputation Management Using EigenTrust

In [32], the EigenTrust algorithm assigns to each peer in the system a global trust value based on peer's history of uploads. This trust value reflects the experiences of all peers with the peer. The authors propose a distributed and secure method to compute global trust value based on power iteration.

EigenTrust has the following characteristics:

- Reputation data: local trust values that represents the number of both satisfied and unsatisfied transactions. Local trust values are normalized and are between 0 and 1.
- Reputation computation: EigenTrust is based on transitive trust. The global reputation of a peer *i* is given by the local trust values assigned to peer *i* by other peers, weighted by the global reputations of the assigning peers. The use of transitive trust leads to a system where global trust values correspond to the left principal eigenvector of a matrix of normalized local trust values. Since each peer *i*

computes and reports its own trust value, malicious peers can easily report false trust values. In secure EigenTrust, a distributed hash table is used to assign score mangers that are responsible to compute global trust values. Each score manager is responsible for a set of peers. For each peer, the score manger learns about the peers that downloaded files from this peer and gets trust assessments from them.

- Credibility mechanism: different score managers are used to compute the global trust value for a peer. A majority vote is used on the trust values to reduce the impact of malicious score managers that report false trust values.
- Malicious peers policy: EigenTrust is robust to malicious collectives of peers who know each other and try collectively to subvert the system. In secure EigenTrust, using the one-way hash function, it is not possible for a score manager to know from whom the global trust value is computed. Malicious peers can not increase the reputation of each other. In addition, peers can not know their location in the hash space, thus, peers are unable to manipulate their own trust value. The system break up malicious collective through the presence of pre-trusted peers.

Drawbacks:

- Normalizing local trust values will not make the distinction between peers who the requester peer did not interact with and peers that performed more unsatisfied transactions than satisfied ones.
- The scheme requires reputations for each provider peer to be computed *on-demand* which requires cooperation and collaboration from a large number of peers in performing computations.
- The scheme introduces additional latency and requires long periods of time to collect data and compute a global trust value for each provider peer.
- The use of a distributed hash table and score managers for each peer in the system in order to collect trust local information and compute global trust values increases significantly the communication overhead.

7.6 Limited Reputation Sharing in P2P Systems: LimitedReputation

In [36], the proposed algorithms use only limited reputation sharing between peers. Each peer records statistics and ratings regarding other peers. As the peer receives and verifies files from peers, it updates the stored data. In the proposed voting reputation system, the requester peer receives ratings from other peers and weights them accordingly to the ratings that the requester peer has for these peers to compute a reputation value. The peers can be selected from the neighbor list (Neighbor-voting) or from the friend list (Friend-voting). In the latter case, friends are chosen from peers who have proved to be reputable. Note that a peer can be reputable, but not credible. No mechanism is given to detect liar peers.

7.7 Reputation Management Using Trust and Credibility Records: CredibilityRecords

In [51], a reputation-based distributed trust system is proposed. In this system, the outcomes of past transactions are stored in trust vectors. These vectors are maintained by peers that perform the downloads. Trust vectors are constant length, binary vectors of m bits. Each bit represents the result of a transaction: 1 if successful, 0 if not. A number is associated with each vector to indicate the number of significant bits. When the requester peer receives a list of provider peers, the peer will compute the reputation values for these peers based on local information. If this information is not available, a trust query is issued in order to inquire about the reputation of the provider peers. The responses are weighted by the credibility ratings of recommender peers. The credibility vectors are similar to the trust vectors. Each bit represents the result of a previous judgment: 1 if the judgment was right, 0 if not.

7.8 Reputation Management Using CCCI Methodology: MDNT

In [15], the authors presented a conceptual framework for measurement of quality and trust. They presented quality assessment through CCCI metrics (Correlation of delivered quality against defined quality, quality Commitment to each of the defined Quality Assessment Criteria, Clarity of each criterion from both parties' views and Influence of each criterion on the overall quality assessment).

The CCCI methodology for a trustworthiness measure provides four metrics, and defines the maximum possible correlation value for a business service interaction, $Corr_{qualities}$ and the maximum possible values of $Commit_{criterionc}$, $Clear_{criterionc}$, $Inf_{criterionc}$. The ratio of the correlation value and the maximum possible correlation value determines the relative correlation value. By using this value, the trustworthiness value can be easily computed.

7.9 Cooperative Peer Groups in Nice

NICE system is a platform for implementing cooperative applications over the Internet. A cooperative application allocates a subset of its resources (e.g., processing, bandwidth and storage) to be used by peers. In Nice, the resource provider is trusted. To access a remote resource, the trustworthiness of the resource requester is inferred. A trust value represents how likely a user consider another user to be cooperative.

In a successful transaction between two users Alice and Bob where Alice consumes a set of resources from Bob, Alice signs a cookie. This cookie states that she has successfully completed a transaction with Bob. A cookie can be stored by Bob to prove his trustworthiness to others. However a negative cookie may be destroyed by Bob. In this case, Alice may store it.

In Nice, each user stores a set of signed cookies. In case Alice wants to get resources from Bob, there are two possibilities:

- Alice has cookies from Bob: Alice gives these cookies to Bob and Bob computes a trust value for Alice
- Alice has no cookies from Bob: Alice uses the cookies that she has. The users who signed these cookies are contacted to check if they have cookies from Bob. For example, Alice has a cookie from Carol, and Carol has a cookie from Bob. Alice gets a copy of the cookie that Carol has and presents the two cookies to Bob. This means that Bob trusts Carol and she trusts Alice. Based on these cookies, Bob infer a trust value for Alice. Based on this trust value, the requested resource will be granted or not to Alice.

In Nice, a query from the requester peer is forwarded to only 5 users instead of following all the paths. Each time, a user is giving a cookie to another user, a copy of its cookies are also given. The users will be selected based on the digests that show a cookie from the resource provider. A digest of negative cookies can also be sent to identify uncooperative users. A preference list is also used to keep track of trustworthy resource providers for future use.

8 Partially Decentralized Reputation Systems

While the centralized systems suffer from the single point of failure, the major challenge in completely decentralized systems is how to collect feedbacks and perform reputation computation efficiently. The high amount of traffic generated from sending a query and getting back results causes inefficiency in using network resources. Moreover, peers in completely decentralized systems have the same role although these peers have different capabilities in terms of processing power, bandwidth, memory and storage capacity, in addition to peers' uptime. Peers that have less capabilities and short uptime should not assume the same role, at least not at the same level. In contrast, the communication protocol used in partially decentralized systems reduces significantly the number of messages exchanged during the search phase. The supernode architecture offers an intermediate design, minimizes the weaknesses of both centralized and completely decentralized systems and combines the advantages of these systems. Several reputation-based systems have been proposed for completely decentralized systems. However, all proposed research works in this field have completely focused on these systems. Almost no attention was directed toward partially decentralized systems that have received a tremendous interest from the online community. KaZaA Media Desktop (KMD) a proprietary partially-decentralized P2P system has introduced a *Participation Level* for rating files and peers. This *participation level* is considered as peers' reputation in the literature. Priority is given to peers with high *participation level*, however the exact process of how this priority is given is not known. In KaZaA, malicious peers will still have a high value of *participation level* even if their participation is affecting badly other peers if they are uploading corrupted content. KaZaA has no mechanism to detect malicious peers.

8.1 BitTorrent

BitTorrent, is a widely used second generation P2P protocol that adopts the *tit-for-tat* strategy. Using this strategy, peers are able to optimize their download and upload rates. Recent studies [10, 31, 57] have shown that the *tit-for-tat* strategy does not effectively reward good peers and punish free riders. Selfish peers can get more bandwidth while honest peers can receive low download rates. In [52], trust has been incorporated to the BitTorrent protocol, however, in this work trust is defined in terms of uploads compared to the downloads not in terms of maliciousness of the provider peer.

8.2 The Inauthentic Detector Algorithm (IDA) and the Malicious Detector Algorithm (MDA)

In [38, 39], trust is addressed according to the following dimensions: (1) *Authentic Behavior*, (2) *Credibility Behavior*, and (3) *Contribution Behavior*.

- *Authentic Behavior*: represents peer's reliability in sending authentic file in terms of accuracy and technical quality. To measure the *Authentic Behavior* of peers, the *Inauthentic Detector Algorithm* is proposed to identify malicious peers and isolate them. This scheme takes into account the size of the file uploaded. Malicious uploads are significantly reduced in addition to distributing the load uniformly among reputable peers.
- *Credibility Behavior*: represents peers' sincerity in providing an honest feedback. The concept of *Suspicious Transaction* is introduced and used to compute the credibility of a peer. A *Suspicious Transaction* is a transaction in which the appreciation is different from the one expected knowing the reputation of the peer uploading the requested file. The *Malicious Detector Algorithm* is proposed to identify peers that send the wrong feedback and minimize their impact. Peers' *Authentic Behavior* is computed according to the credibility of peers downloading the requested files.
- *Contribution Behavior*: represents the positive contribution of a peer to the system. Peers' contribution is based on:
  - Peers' *Availability*: being available for uploading requested files.

– Peers' *Involvement*: non-malicious uploads performed versus downloads received by a peer.

A *contribution-based service differentiation* scheme is proposed to differentiate between peers that contribute positively to the system (i.e., altruistic), and free riders and malicious peers (i.e., egoistic). This service differentiation is based on peers' contribution rather than peers' reputation.

According to [40], service differentiation is divided into two categories: *implicit* and *explicit*. The *Implicit* service differentiation results from the normal evolution of the system. *Explicit* service differentiation, is the one that results from the explicit decision of supernodes or peers. Using peer's contribution as a guideline for service differentiation will provide better service to peers that contribute positively and will reduce the level of service provided to free riders and malicious peers. Peers are also forced to continuously contribute to benefit from the services provided hence, minimizing the impact of the milking phenomenon.

9 Conclusion

Reputation-based trust management in peer-to-peer systems is an interesting research area and very challenging. Reputation is used to build trust among peers, minimize the risk involved in the transactions and increase users' confidence and satisfaction. Reputation is based on evaluating the transactions performed by peers. In this chapter, we surveyed reputation-based trust management P2P systems. We investigated the existing research work proposed to address peers' reputation. To understand better the functioning of reputation systems, we divided a typical reputation system into components. A description is given to each component to help in analyzing and summarizing the efforts of researchers in addressing peers' reputation. We also presented some of the existing centralized, completely decentralized and partially decentralized reputation management systems. The survey of different reputation systems reveals the important mechanisms used to achieve efficient reputation management.

The success of reputation-based e-Commerce applications has fostered the advancement of reputation management and its use in a wide area of applications. The study of P2P reputation management systems in this chapter exposes the different mechanisms used to solve the various challenges faced by these systems. Since reputation management is a risk assessment device, different constraints and parameters should be taken into consideration while designing a reputation system according to the risk involved in the transactions. In particular, file sharing systems are different from e-Commerce applications. The stability of these reputation systems and their robustness against malicious attacks, and security threats are of paramount importance and represent the main future research directions.

References

1. eBay Feedback. Http://www.ebay.com
2. Gnutella Protocol Specification v0.4. Http://www9.limewire.com/
3. Gnutella2 Specification. Http://www.gnutella2.com/
4. KaZaA. Http://www.kazaa.com/
5. Morpheus. Http://www.morphus.com/
6. Oxford Dictionary. Http://www.askoxford.com/
7. PPLive. Http://www.pplive.com/en/about.html
8. Skype. Http://www.skype.com/intl/fr/welcomeback/
9. UUsee. Http://newteevee.com/2007/03/07/chinas-uusee-scores-235-million/
10. Bharambe, A., Herley, C., Padmanabhan, V.: Analyzing and Improving a BitTorrent Network's Performance Mechanisms. In: IEEE 25th INFOCOM, pp. 1–12 (2006)
11. Abdul-Rahman, A., Hailes, S.: A distributed trust model. In: The 1997 workshop on New security paradigms, pp. 48–60. ACM Press (1997)
12. Abdul-Rahman, A., Hailes, S.: Supporting Trust in Virtual Communities. In: Proceedings of the 33rd Hawaii International Conference on System Sciences, p. 6007. IEEE Computer Society, Washington, DC, USA (2000)
13. Aberer, K., Despotovic, Z.: Managing Trust in a Peer-2-Peer Information System. In: 9th International Conference on Information and Knowledge Management, pp. 310–317 (2001)
14. Oram, A.: Peer-to-Peer: Harnessing the Power of Disruptive Technologies, pp. 21–37. O'Reilly Books (2001)
15. Chang, E., Dillon, T., Hussain, F.K.: Trust and Reputation for Service-Oriented Environments. Wiley (2006)
16. Chhabra, S., Damiani, E., di Vimercati, S.D.C., Paraboschi, S., Samarati, P.: A Protocol for Reputation Management in Super-Peer Networks. In: DEXA Workshops, pp. 979–983 (2004)
17. Cornelli, F., Damiani, E., di Vimercati, S.D.C., Paraboschi, S., Samarati, P.: Choosing Reputable Servents in a P2P Network. In: 11th International World Wide Web Conference, pp. 376–386 (2002)
18. Damiani, E., di Vimercati, S.D.C., Paraboschi, S., Samarati, P., Violante, F.: A reputation-based approach for choosing reliable resources in peer-to-peer networks. In: 9th ACM conference on Computer and communications security, pp. 207–216. ACM Press (2002)
19. Dellarocas, C.: The Digitization of Word of Mouth: Promise and Challenges of Online Feedback Mechanisms. Management Science **49**(10), 1407–1424 (2003)
20. Despotovic, Z., Aberer, K.: P2P reputation management: probabilistic estimation vs. social networks. Computer Networks **50**(4), 485–500 (2006)
21. Deutsh, M.: The resolution of Conflict:Constructive and Destructive. Tech. rep., New Haven, Yale University Press (1973)
22. eBay: http://www.ebay.com
23. Feldman, M., Lai, K., Stoica, I., Chuang, J.: Robust Incentive Techniques for Peer-to-Peer Networks. In: Proceedings of the 5th ACM conference on Electronic commerce, pp. 102–111. ACM, New York, NY, USA (2004)
24. Feldman, M., Papadimitriou, C., Chuang, J., Stoica, I.: Free-riding and whitewashing in Peer-to-Peer systems. In: Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems, pp. 228–236. ACM, New York, NY, USA (2004)
25. Gambetta, D.: Can We Trust Trust? In: Trust: Making and Breaking Cooperative Relations, chap. 13, pp. 213–237. Published Online (2000)
26. Grandisan, T., Sloman, M.: A survey of Trust in Internet Applications. In: IEEE Communications Surveys, vol. 3 (2000)
27. Gupta, M., Ammar, M.: Service Differentiation in Peer-to-Peer Networks Utilizing Reputations. In: ACM Fifth International Workshop on Networked Group Communications, pp. 70–82 (2003)

28. Gupta, M., Ammar, M.H., Ahamad, M.: Trade-offs between Reliability and Overheads in Peer-to-Peer Reputation Tracking. Computer Networks **50**(4), 501–522 (2006)
29. Gupta, M., Judge, P., Ammar, M.: A Reputation System for Peer-to-Peer Networks. In: ACM 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video, pp. 144–152 (2003)
30. Josang, A., Ismail, R., Boyd, C.: A Survey of Trust and Reputation Systems for Online Service Provision. Decis. Support Syst. **43**(2), 618–644 (2007)
31. Jun, S., Ahamad, M.: Incentives in BitTorrent Induce Free Riding. In: Workshop on Economics of Peer-to-Peer Systems, pp. 116–121 (2005)
32. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The EigenTrust Algorithm for Reputation Management in P2P Networks. In: 12th International World Wide Web Conference, pp. 640–651 (2003)
33. Lee, S., Sherwood, R., Bhattacharjee, B.: Cooperative Peer Groups in NICE. In: IEEE Infocom, pp. 1272–1282. San Francisco, USA (2003)
34. Ma, R.T.B., Lee, S.C.M., Lui, J.C.S., Yau, D.K.Y.: Incentive and Service Differentiation in P2P Networks: a Game Theoretic Approach. IEEE/ACM Trans. Netw. **14**(5), 978–991 (2006)
35. Marsh, S.: Formalising Trust as a Computational Concept. Ph.D. thesis, University of Stirling (1994)
36. Marti, S., Garcia-Molina, H.: Limited Reputation Sharing in P2P Systems. In: ACM Conference on Electronic Commerce, pp. 91–101. New York, USA (2004)
37. Marti, S., Garcia-Molina, H.: Taxonomy of trust: categorizing P2P reputation systems. The Computer Networks Journal, Special Issue on Management in Peer-to-Peer Systems: Trust, Reputation and Security **50**(4), 472–484 (2006)
38. Mekouar, L., Iraqi, Y., Boutaba, R.: Free Riders under Control through Service Differentiation in Peer-to-Peer Systems. In: IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2005), pp. 10–20 (2005)
39. Mekouar, L., Iraqi, Y., Boutaba, R.: Peer-to-Peer Most Wanted: Malicious Peers. The Computer Networks Journal, Special Issue on Management in Peer-to-Peer Systems: Trust, Reputation and Security **50**(4), 545–562 (2006)
40. Mekouar, L., Iraqi, Y., Boutaba, R.: A Contribution-Based Service Differentiation Scheme for Peer-to-Peer Systems. Submitted (2008)
41. Mui, L., Mohtashemi, M., Halberstadt, A.: A Computational Model of Trust and Reputation for E-businesses. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences, pp. 2431–2439. IEEE Computer Society, Washington, DC, USA (2002)
42. Papaioannou, T.G., Stamoulis, G.D.: Effective Use of Reputation in Peer-to-Peer Environments. In: IEEE/ACM CCGrid: International Symposium on Cluster Computing and the Grid, pp. 259–268 (2004)
43. Papaioannou, T.G., Stamoulis, G.D.: Reputation-based Policies that Provide the Right Incentives in Peer-to-Peer Environments. the Computer Networks Journal: Special Issue on Management in Peer-to-Peer Systems: Trust, Reputation and Security pp. 563–578 (2006)
44. Ranganathan, K., Ripeanu, M., Sarin, A., Foster, I.: To Share or not to Share' An Analysis of Incentives to Contribute in File Sharing Environments (2003)
45. Ranganathan, K., Ripeanu, M., Sarin, A., Foster, I.: Incentive Mechanisms for Large Collaborative Resource Sharing. In: International Symposium on Cluster Computing and the Grid, pp. 1–8 (2004)
46. Ratnasamy, S.: A Scalable Content-Addressable Network. Ph.D. thesis, University ofCalifornia, Berkeley (2002)
47. Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E.: Reputation Systems. Commun. ACM **43**(12), 45–48 (2000)
48. Ruohomaa, S., Kutvonen, L., Koutrouli, E.: Reputation Management Survey. In: Proceedings of the The Second International Conference on Availability, Reliability and Security, pp. 103–111 (2007)

49. Sabater, J., Sierra, C.: REGRET: Reputation in Gregarious Societies. In: Proceedings of the fifth international conference on Autonomous agents, pp. 194–195. ACM, New York, NY, USA (2001)
50. Sabater, J., Sierra, C.: Reputation and Social Network Analysis in Multi-agent Systems. In: Proceedings of the first international joint conference on Autonomous agents and multi-agent systems, pp. 475–482. ACM, New York, NY, USA (2002)
51. Selcuk, A., Uzun, E., Pariente, M.: A reputation-based trust management system for P2P networks. In: Fourth IEEE International Symposium on Cluster Computing and the Grid, pp. 251–258 (2004)
52. Shah, P., Paris, J.: Incorporating Trust in the BitTorrent Protocol. In: International Symposium on Performance Evaluation of Computer and Telecommunication Systems, pp. 586–593 (2007)
53. Song, S., Hwang, K., Zhou, R., Kwok, Y.: Trusted P2P Transactions with Fuzzy Reputation Aggregation. IEEE Internet Computing **9**(6), 24–34 (2005)
54. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In: Proceedings of the ACM SIGCOMM '01 Conference, pp. 149–160. San Diego, California, USA (2001)
55. Suryanarayana, G., Taylor, R.N.: A Survey of Trust Management and Resource Discovery Technologies in Peer-to-Peer Applications. Tech. rep., ISR (2004)
56. Teacy, W.T., Patel, J., Jennings, N.R., Luck, M.: TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources. Autonomous Agents and Multi-Agent Systems **12**(2), 183–198 (2006)
57. Thommes, R., Coates, M.J.: BitTorrent Faireness: Analysis and Improvements. In: Workshop of the Internet Telecommunications and signal Processing (2005)
58. Wang, Y., Vassileva, J.: Trust and Reputation Model in Peer-to-Peer Networks. In: Proceedings of the 3rd International Conference on Peer-to-Peer Computing, p. 150. IEEE Computer Society, Washington, DC, USA (2003)
59. Xiong, L., Liu, L.: PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. IEEE Transactions on Knowledge and Data Engineering **16**(7), 843–857 (2004)
60. Yang, M., Feng, Q., Dai, Y., Zhang, Z.: A Multi-dimensional Reputation System Combined with Trust and Incentive Mechanisms in P2P File Sharing Systems. In: Proceedings of the 27th International Conference on Distributed Computing Systems Workshops, p. 29 (2007)
61. Zhang, Y., Fang, Y.: A Fine-Grained Reputation System for Reliable service Selection in Peer-to-Peer Networks. Trans. IEEE **18**(8), 1134–1145 (2007)
62. Zhou, R., Hwang, K.: PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. IEEE Transactions on Parallel Distributed Systems **18**(4), 460–473 (2007)

P2P Reputation Management Through Social Networking

Zoran Despotovic

Abstract Reputation systems offer a viable solution to the problem of risk reduction in online communities, in situation in which other mechanism such as litigation or security cannot help. Building on the assumption that its participating entities engage in repeated interactions, a reputation system can either signal what happened in the past or aggregate the past feedback in such a way as to influence the future actions of the concerned entity. In the former case, the concerned entity's behavior is seen as static, while the sent signal is expected to be indicative of the entity's future actions. In the latter case, behavior is dynamic in the sense that the entity can adjust it given the observed feedback, while the purpose of the reputation system is to induce adjustments according to the designer's needs. In this chapter, we discuss these two classes of solutions in detail. In particular, we investigate how they apply to P2P networks, what additional problems and difficulties the P2P environment introduces and what scalable solutions to these problems the current research offers.

1 Introduction

This chapter discusses P2P reputation systems and provides their categorization and classification. However, to set up the stage, we start with a general discussion on the phenomenon of reputation and its importance.

Reputation has always been important to humans. Its key role is as an enabler of interactions in situations when other assurance mechanisms are not viable. When contractual agreements or enforcement by third parties are not possible, say, due to unacceptably large transaction costs and the interacting parties have no previous experience with each other, then it makes sense for them to consult "public opinion" on their partners. This public opinion constitutes the partners' reputation. Although

Zoran Despotovic
DOCOMO Communications Laboratories Europe GmbH, Munich, Germany,
e-mail: despotovic@docomolab-euro.com

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_25, © Springer Science+Business Media, LLC 2010

the phenomenon was observed even in ancient times, it is the modern internet society that brought about the current strong need for reputation management. The main reason is that in a large scale online community a vast majority of interactions are between complete strangers. Take eBay (www.ebay.com) as an example: it is quite probable that any given buyer will purchase every new item from a seller she never met before. P2P systems offer even a more severe example: it is hard to believe that any two files downloaded by a given peer will come from the same source.

Before we start with technical details, we touch upon two important questions. The first one has to do with motivation for P2P reputation management: Are there evidences of the need for P2P reputation management? The answer is: Yes, there are many. As an example, a report from the Wired magazine reveals that as of September 2004 “forty-five percent of the executable files downloaded through Kazaa, the most popular file-sharing program, contain malicious code like viruses and Trojan horses.” The second question is: Would deploying reputation systems effectively decrease the threats associated with P2P systems. It is a bit harder to answer this question in the context of P2P systems, as there are no deployed reputation mechanisms which could provide the answer. However, we can rely on analyses of centralized systems, such as eBay. Recent empirical studies have shown that a great deal of the commercial success of eBay, the largest online auctioning site, can be attributed to its reputation mechanism (Feedback Forum) as a means of deterring dishonest behavior. The analysis of eBay data carried out by [36] has shown that “reputation profiles were predictive of future performance,” while more specific analyses of [22] and [28] brought the conclusion that Feedback Forum fulfilled its promises: the positive feedback of the sellers was found to increase their prices, while the negative one reduced them.

To start with, we repeat the informal definition of reputation systems from [37]. Resnick et al. [37] define reputation systems as “systems that help people decide whom to trust, encourage trustworthy behavior, and deter participation by those who are unskilled or dishonest through collecting, distributing, and aggregating feedback about the participants past behavior.” The last item in this list, i.e., strategy to aggregate feedback in order to help people make informed decisions prior to interacting with their partners, is the heart of any reputation system. Assume that all interactions are logged and interacting partners are presented with complete logs of what happened before they decide what to do. Most likely they will not find their way through such logs and the system will fail to fulfill its promise. But, if suitable statistics are computed from such logs, the situation may change radically.

This holds for any reputation system, irrespective of the properties of its target environment, e.g., distributed vs. centralized. However, relevant properties of the environment constrain the space of viable solutions. A solution perfectly acceptable for a centralized online market such as eBay, may not be acceptable for a decentralized environment such as a P2P network. For example, a reputation system that needs a large amount of feedback that is scattered throughout the network may be unusable, because it may take too long to retrieve relevant data from the network and then aggregate it. Or, consider another example. To enable fast lookup, data can be stored in a systematic way, e.g., in an underlying Distributed Hash Table (DHT), such as [1, 35, 39]. But, nodes storing the data may find it profitable to alter it. This is not the case when the storage is provided by a single trusted entity, without any interest in tampering with it.

The dot-com boom in mid nineties was the main driving force for work on online reputation management. Since then, a large body of work on reputation management emerged. However, we need to understand that the topic is much older than that. In particular, reputation has long been a subject of study in economics. An important part of this chapter deals with the way reputation is modeled in economics, i.e., game theory. Let us take a look at the following example to illustrate the point. Consider a market in which sellers sell goods of different qualities. Buyers cannot observe the quality of any good. Thus, they tend to undervalue high quality goods, as they may end up purchasing low quality. But then high quality sellers cannot achieve good prices and may withdraw from the market. So only low quality goods will be traded. This is what George Akerlof calls the "market for lemons" [4]. Information asymmetry between sellers and buyers is critical here. A mechanism to break it is needed. Reputation systems may be such a mechanism.

The chapter is structured as follows. Section 2 introduces the problem that reputation systems address. We first discuss the problem in general, and then also take a look at the P2P dimension of the problem in Section 2.1. Section 2.2 lists the main classes of solutions found in the literature. We make a distinction between signaling and sanctioning reputation systems. A further distinction is made between probabilistic and ad hoc signaling methods. Each of these classes is discussed in a separate section. Section 3 describes probabilistic signaling reputation systems, Section 4 discusses ad hoc signaling methods, while Section 5 looks at sanctioning reputation mechanisms in detail. Section 6 discusses some practical problems that originate in the possibility to change identities at no or low cost. We believe that these problems are so important for practical deployments that they deserve a section on their own. We conclude in Section 7 with a discussion and an outline of open research topic.

2 P2P Reputation Systems

We start this section by setting up the stage for the rest of the chapter. A central notion we are interested in is an online system in need of a reputation system, e.g., a P2P file sharing network or eBay. We assume that there are well defined participants in the system and interactions among them in the context of a specific application. Examples are eBay buyers competing in auctions to purchase goods from eBay sellers or peers downloading files from one another in a P2P file sharing application. An obvious way to record all interactions is making a graph. An arc in the graph would correspond to an interaction between the entities represented by the arc endpoints. We call such a graph social network as it simply encodes interactions among entities of the considered community, i.e., social interactions. Strictly speaking, a social network is a multigraph rather than a graph, as any two nodes may be joined by more than one arc. Nevertheless, we will continue to use the term graph, the term multigraph will be used only when we want to stress the multiplicity of arcs between pairs of nodes.

The arcs in a social network can be given another meaning. For any interaction, the involved entities can rate each other's behavior in the interaction and the rating can be added as a weight to the corresponding arc. If so, the social network effectively becomes a directed weighted graph: its nodes represent the entities (agents, participants) of the considered community, its arcs represent the interactions among them, while weights represent the rating (feedback) of the corresponding arcs' endpoints (service providers) as provided by the arcs' starting points, i.e., service consumers. Figure 1 (taken from [14]) gives an example of such a graph. The figure assumes that feedback takes the form of ordered pairs with the first component being a flag d or r and the second a real number from the interval $[0,1]$. The reasoning behind this decision is that it introduces another type of interactions, besides the direct service provisioning in the considered application context. The new type is interactions in the context of the reputation system itself. Assume that nodes can ask other nodes for their opinion about some other nodes. This looks like asking a friend to tell you what experience he had with an eBay seller. Once the service is provided, you can rate not only the seller but also the friend who provided you with the recommendation. The latter rating has a different context, it says how good your friend is in recommending other entities, rather than how good he is in providing the service (i.e., what quality goods he sells). Thus the way we should understand Fig. 1 is as follows. Node a had three interactions with node b: once node b acted as a recommender of other entities (flag r) and node a's contention with the recommendation was evaluated 0.8 and twice node b provided the service in question to node a (flag d) and a's evaluations of the service provisions were 1 and 0.9 respectively.

The main message of the discussion so far is that one has to decide what interactions are rated, what ratings can be provided, i.e., what is the assumed feedback set (we will denote it W) from which these ratings are drawn and, in particular, whether the feedback set explicitly takes into account that interactions belong to possibly different contexts? Examples of feedback sets are the set $\{r,d\} \times [0,1]$, as above, the interval $[0,1]$, the two element set $\{0,1\}$ or any discrete grading such as the four element set $\{very\ good, good, bad, very\ bad\}$. In any case, the set W as well as the semantics associated with its individual elements are assumed to be universally known and agreed upon. In particular, we are assuming that there is a binary partial ordering relation (call it "greater than") defined on W with the interpretation that "greater" elements mean better feedback.

A number of further questions immediately arise: Where is the graph stored? How is it used to help making decisions? In a centralized system, the entity running the system normally provides storage for reputation data graph. In case of a decentralized system, such as a P2P network, it is not so obvious what the best strategy for storing the graph is. We describe briefly two possible strategies, a more elaborate discussion follows in the next section. In the first one, every peer records ratings it has made about others. When data about a specific target peer is needed,

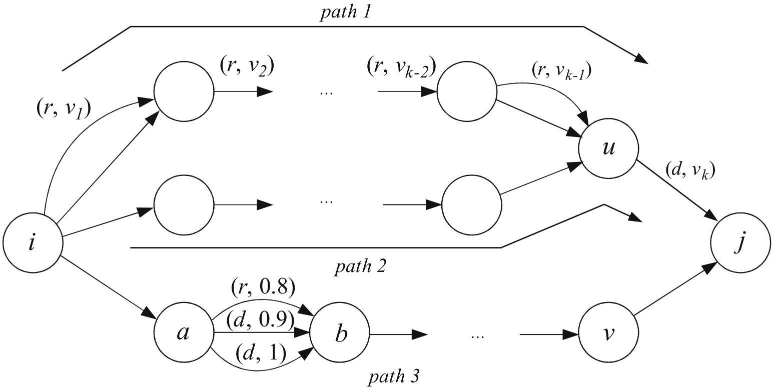


Fig. 1 A social network

i.e., ratings the peer has received from other peers, the requesting peer can navigate through the social network until it reaches the destination peer. Note that, this strategy does not make any assumption on the underlying P2P network. In fact, it is not aware of it at all. The second strategy assumes that the underlying P2P system is organized as a DHT. This DHT could be used to store reputation data in the form of $(providerID, rating)$ key-value pairs. Data about any given peer is in this way immediately available through using the DHT's search facility.

We now give a number of possible answers to the second question. They are discussed by means of examples from Fig. 1. To predict future performance of a peer, say, j, we could propagate direct experiences of the peers which interacted with j (peers u and v) through the graph down to the peer doing the assessment (say, peer i), filtering them out by the recommendation experiences among the neighboring nodes along the paths in order to decide on their credibility. Different works propose different strategies for doing this. We stress that most of the existing works do not model explicitly the context of recommendations but rather use direct experiences as credibility filters. This can be thought of as weighting one's reports by his trustworthiness to perform a service rather than his ability to recommend.

Another possibility might be that we simply take average of all reports about a peer in question. This should be an efficient strategy in terms of communication overhead spent on retrieving relevant data. However, it is not clear how good indication of the most likely performance of the target peer it would offer.

In a word, we need an algorithm, denote it A, that operates on the formed social network, aggregates the feedback available in it, and for any peer given it outputs a value $t \in T$ which represents an indication of the peers likely behavior. Note also the algorithm may output a recommendation for how the peer running the algorithm should behave in the interaction with the target peer.

Thus, the social network, the feedback set W, the algorithm to aggregate the data in the social network and the output set T constitute what we call a reputation management system.

2.1 P2P Systems Perspective

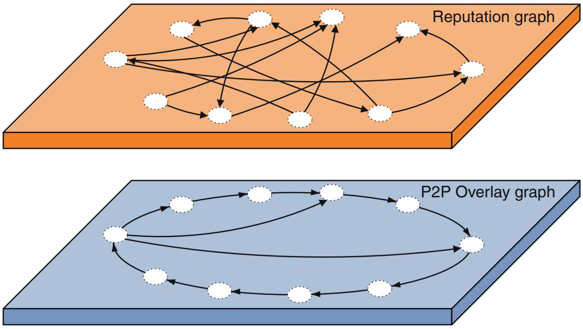


Fig. 2 Social network layered on top of a P2P overlay network

Figure 2 illustrates the difference between the social graph and the underlying P2P overlay network. The social graph can be seen as another graph layered on top of the P2P overlay network. Mapping between the nodes in the two graphs need not necessarily be one-to-one. More than one peer (i.e., P2P overlay node) can act as one entity in the social graph and vice versa. However, the figure assumes that this is not the case, i.e., each peer corresponds to one and only one application entity.

Recall that one of the steps in a reputation system operation is retrieving relevant reputation data. At minimum, this data includes feedback about the node whose reputation is aggregated. So the question is: how can this data be efficiently retrieved? A possible answer is immediately visible from Fig. 2. The social graph can be directly used for this purpose. The P2P overlay graph is irrelevant in this case. In such a solution, every node needs to maintain a list with its interactions with other nodes. Each entry in the list should include the information such as the physical (e.g., IP) address of the other interacting node and the feedback associated with the interaction. These lists at all nodes in the system would effectively constitute a social graph, that could be used also to retrieve the needed data. A discussion on the pros and cons of such a solution is given below, after we introduce another possible solution.

The second possibility is to store the social graph in the underlying P2P network. The underlying network, be it structured [1, 35, 39] or unstructured [9, 20, 27] can be seen just as a hash table storing a set of $(key, value)$ pairs. Reputation data that we need to store has exactly this form. For any reputation data item, *key* could be the identifier of the peer which was the target of the corresponding feedback (or, equivalently, the endpoint of the corresponding edge in the social graph). The actual feedback could play the role of *value*. Note that other organizations are also possible. For example, the *key* fields might be formed by concatenating the two endpoints of the arcs, while the *value* still holds the associated reputation information. When the underlying P2P network is unstructured, every peer can store its outgoing edges from the social graph. When the network is structured, the $(key, value)$ tuples are stored at peers as dictated by the underlying storage, i.e., each tuple is stored at the peer whose ID is closest the hashed value of the key.

Which of these two possibilities is better? We believe that the second one is always better, irrespective of the type of the underlying P2P network. Retrieving reputation data about any specific peer requires flooding if the network is unstructured. The overhead scales in this case linearly with the number of arcs. This number is typically higher in the social graph, as unstructured P2P networks are normally made to follow a power-law outdegree distribution, giving rise to a logarithmic degree on average. The conclusion is more obvious when the underlying P2P network is structured. The reputation data about any specific peer can be retrieved with $O(\log N)$ overhead ($O(N)$ to retrieve the entire trust network). So the underlying structured storage should be used to store the reputation data.

2.2 Classification of Existing Solutions

The main goal of a reputation system is inducing specific behavior, that the reputation system designer finds appropriate. Ideally, this behavior should be exerted in every interaction, no matter who the interacting entities are. In practice, a weaker goal is often acceptable: reduce the fraction of interactions with undesired outcomes as much as possible. Consider again the well known eBay scenario. The best would be if every purchased item is delivered by the respective seller, and if the quality of every item is as its buyer expected. If this is not possible the rate of purchases with unsatisfied buyers should be kept as low as possible. According to [13], there are two possible ways a reputation system can do this. One is just to *signal* to the interacting entities what can go wrong in the interactions if they behave in specific ways. It is then up to the entities to decide what strategies are best for them. For example, the system may just provide a prospective buyer with indications of the probabilities that an item the buyer is interested in will not be delivered by the sellers offering it. Then the buyer can select the seller with the minimum probability and purchase the item if this probability is not too small. However, whatever the outcome of the interaction and the report on it, they will not enforce the seller to change his behavior in the future.

If agents are capable of reasoning strategically about one another and adjusting their behavior dynamically, then another possibility exists. In this case, it is typically maximization of the long-run profit (over her whole lifetime) that an agent is interested in. The main task of a reputation system in this case is to *sanction* misbehavior through providing correlation between the feedback the agent receives and the long-run profit she makes. The main (albeit subtle) difference in comparison with signaling reputation systems is this should hold irrespective of whether the agent is the only one offering a specific service.

Put differently, the two approaches make different assumptions on the underlying behavior. Sanctioning mechanisms assume that behavior is dynamic, i.e., actions taken by an agent at two different time instants are correlated. Signaling mechanisms assume no such correlation across time. In the rest of the chapter we will use more specific terms to denote these two classes of behavior: probabilistic and rational, respectively. Rational behavior implies that there is an underlying economic model in which utilities are associated with various choices of the agents and that the agents act as to maximize their utilities. We will say more on this in Section 5, in which we discuss sanctioning reputation systems.

Probabilistic behavior assumes that a joint probability distribution is associated with the set of all peers, describing their innate characteristics and determining how they behave in the context of direct service provisioning and when reporting on others' performance. Thus any event in the form

$$(\ldots, p_k \text{ performes } w_k \ \ldots, p_l \text{ reports } w_l \text{ when } p_m \text{ performs } w_m \ \ldots),$$

where $w_k, w_l, w_m \in W$, should have an assigned probability. For instance, a given distribution may specify

$$P[p_1 \text{ misreports on } p_3 | p_2 \text{ misreports on } p_3] \neq P[p_1 \text{ misreports on } p_3],$$

meaning that peers p_1 and p_2 misreport on peer p_3's performance in a coordinated manner, forming thus a collusive group.

The true distribution in not known in advance. It is the goal of reputation systems to assess the marginal distribution of peer p_i performing in specific ways $P[p_i \text{ performs } w_m]$.

When discussing the two classes of reputation systems, we will also pay attention to the *implementation overhead* they incur. It is important because P2P networks normally involve large numbers, e.g., millions, of nodes. Thus, only scalable reputation systems are acceptable. Generally, the implementation overhead consists of:

- the communication costs associated with the process of retrieving the necessary feedback,
- the involved storage costs,
- and the computation overhead related to the feedback aggregation.

The communication costs are determined by the amount of the necessary feedback on which a given algorithm operates, while the storage costs become important primarily in the case a caching scheme is used to cut the communication costs. The employed feedback aggregation strategy determines the computation overhead.

The above discussion already indicated the division we will follow. We established two broad groups of approaches: *signaling* and *sanctioning* reputation mechanisms. Signaling reputation systems aim at reducing risks associated with interactions by signaling relevant parameters associated with the peer behavior. For example, peers could be assigned probability distributions and every interaction of a given peer could be considered as a trial drawn from the peer's distribution. A signaling reputation mechanism would try to output relevant parameters of the distribution associated with any peer. However, we stress that the true underlying behavior does not have to be static. It can change with time. A signaling reputation mechanism may even make explicit the assumption that behavior is dynamic, for example through making weighted averages of the available feedback data items with more weight put on more recent reports. What makes them signaling is the outputs they make, i.e., the fact that these outputs can be at best interpreted as indicators (signals) of the likely performance at any instant of time.

A further distinction can be made based on how plausible these signals are, i.e., how strong they are. *Probabilistic* signaling reputation mechanisms assume that the underlying behavior is probabilistic, just as we described previously. The signals they output have a clear interpretation, they are parameters of the corresponding distributions. *Ad hoc* signaling mechanisms fail to provide a clear interpretation of their output signals. On the one hand, they are not parameters of probability distributions characterizing peers, and, on the other, they often have relative meaning in the sense that knowledge of the signal of a specific peer is of little use unless we know the signals of many or all other peers.

Sanctioning reputation systems rely directly on the assumption that the underlying behavior is dynamic. In fact, they are even more specific, assuming that it is rational in the game-theoretic sense. This means that the involved agents are interested in maximizing their *long-run* profits, i.e., across their entire lifetime. More specifically, when selecting an action today, they reason about what impact the report on their current behavior can have on what actions their opponents select against them tomorrow.

This division is followed in the rest of the chapter so that each of the established classes will be discussed in a separate section. Section 3 deals with probabilistic signaling mechanisms, Section 4 discusses ad hoc signaling mechanisms, while sanctioning mechanisms are described in Section 5. Each section includes a general description of the corresponding solution class as well as a discussion on the implementation overhead in the context of P2P systems.

3 Probabilistic Signaling Reputation Mechanisms

The key property of probabilistic signaling reputation mechanisms is that they make an explicit assumption about probabilistic peer behavior, as we described it in Section 2.2. In its simplest form, every peer tries to assess parameters of probability distributions of other peers by considering only own experiences with those peers or, in the lack of (a sufficient number of) these, by taking into account the second level judgments, i.e., reports of the other peers about past behavior of the peers being assessed. More precisely, the starting assumption is that any peer is associated with a probability distribution that determines the peer's behavior in its interactions. Presumably, the distribution type is known, but its parameters are not. Further, when reporting its own experiences with others, each peer may lie with some, again unknown probability.

Given these assumptions, we can use well known probabilistic estimation techniques to learn all unknown parameters, each time we obtain new information, either a direct experience from an interaction with the peer in question or a new report from a peer. The main value of these mechanisms is that they offer interacting parties the possibility to estimate the risks of the interaction and decide whether to enter it or not, and how to behave if decision to perform the interaction has been made.

Bayesian estimation is a typical representative of statistical estimation techniques. Its main idea is to assign a prior probability distribution to an unknown parameter and calculate its posterior whenever new observations on the parameter value become available. We describe in short how it can be used for estimating the unknown parameter of a Bernoulli distributed random variable. Such a task can be encountered for example when the interactions among peers have two outcomes and can be treated as Bernoulli random variables. According to the notation from Section 2, this means $W = \{0,1\}$ and $T = [0,1]$. Any prior knowledge on the distribution of the unknown parameter can be conveniently modeled as a beta distribution with appropriately chosen parameters. Posterior realizations of the variable lead only to a change in the beta distribution parameters, they do not change the distribution type. Thus, the unknown parameter remains beta distributed, but now with different parameters. To be more concrete, assume that the unknown probability θ of a specific action A of a given peer has the prior distribution $Beta(a,b)$.[1] If we now observe n realizations of the peer's interactions, k of which were A then the posterior distribution of θ becomes $Beta(a+k, b+n-k)$. The Bayes' estimator of θ is the expected value of $Beta(a+k, b+n-k)$. It can be shown that this value equals $\frac{a+k}{a+b+n}$ and that the estimator is asymptotically unbiased and consistent.

All this applies only when own experiences are taken into account. Reference [30] presents an example in which this approach is used. Besides, it also provides the minimum bound on the number of encounters one has to have with another peer in order to remain within a specific probability of estimation error. It is given by the following inequality: $m \geq \frac{1}{2\varepsilon^2} \ln\left(\frac{\delta}{2}\right)$, where ε and δ are the estimation error and confidence level respectively. However, it was left unspecified how to apply the method to integrate reports from direct witnesses when no meaningful decision can be made based on own experiences only. Reference [8] makes a step towards this. However, even though the authors discuss a number of possibilities to deal with the "second hand" opinions they use an intuitive approach in which all second level information sources are given equal weights. To the best of our knowledge, there is no P2P reputation model extending the Bayesian estimation technique in the most natural way (so called "super Bayesian estimation") to take these or higher level beliefs into account as well. In comparison with the "one source" Bayesian models the only difference would be that the samples do not come from the same

[1] A random variable Θ is distributed $Beta(a,b)$ if its probability density function is $f(\theta) = \frac{1}{B(a,b)}\theta^{a-1}(1-\theta)^{b-1}$, $0 < \theta < 1$, $a > 0$, $b > 0$, where $B(a,b) = \int_0^1 x^{a-1}(1-x)^{b-1}dx$. a and b are the parameters of the distribution, the case $a = 1$ and $b = 1$ corresponds to the uniform distribution.

distribution representing the service provider's trustworthiness but from different ones as the original samples now pass through the second level sources who may misreport. But, the probability distribution of these misreports can be also estimated and updated with new experiences so that calculating the posterior distribution of the unknown parameters is not harder at all. This also enables an easy separation of the contexts of recommendations and direct service provisions. We will see later on that this can bring some benefits. The approach has also some points in common with works relying on similarity based computation [42], the key difference is that it has a stronger theoretical foundation. In a similar way it can be extended to cover third and even higher level experiences.

Maximum likelihood estimation presents another method to make statistical inference. Unlike Bayesian estimation, which provides a method to assess probability distributions of unknown parameters, maximum likelihood estimation gives most likely values of the parameters given a set of samples. More specifically, assume that we know the form of the probability distribution of a random variable but do not know the exact values of involved parameters. The main idea of the approach is to compute the likelihood of the observed sample for general values of the unknown parameters and fit the values that maximize it.

Consider an example taken from [15]. It starts with the assumption that peers are fully characterized by two parameters: the probability of performing honestly in their interactions with others (θ_j denotes the probability of peer j) and the probability of lying when asked to report on their experience with others (l_j denotes the probability of peer j). Assume that peer j interacted with peers $p_1, \ldots, p_n$ and its performances in these interactions were $x_1, \ldots, x_n$, where $x_i \in \{0, 1\}$ (1 denoting the honest performance and 0 the dishonest one). When asked to report on peer j' performances, witnesses $p_1, p_2, \ldots, p_n$ may lie and misreport. The probability of observing report y_k from peer p_k can be calculated as:

$$P[Y_k = y_k] = \begin{cases} l_k(1-\theta_j) + (1-l_k)\theta_j \text{ if } y_k = 1 \\ l_k\theta_j + (1-l_k)(1-\theta_j) \text{ if } y_k = 0. \end{cases}$$

Given a random sample of independent reports $y_1, y_2, \ldots, y_n$, the likelihood function of the sample is

$$L(\theta_j) = P[Y_1 = y_1]P[Y_2 = y_2] \cdots P[Y_n = y_n]. \quad (1)$$

The maximum likelihood estimation procedure implies simply finding θ_j that maximizes this expression. This number is the maximum likelihood estimate of the unknown probability. However, to compute θ_j that maximizes (1), one needs to know all l_k's. [15] proposes a simple trick to estimate these unknowns: every peer needs to compare the reports about own performances with reports about them. This provides a rough assessment of the level of lying in the community and can be understood as the probability of lying of an average peer in the system.

Reference [2] presents another approach to reputation management in P2P networks built explicitly on probabilistic assumptions. It does not provide any prediction of a likely future performance of the peers in terms of a probability distribution over the possible performances. Instead, it tries to assess whether a given peer has ever cheated in the past. Precisely, the interactions considered in this work are binary (the peers either cheat or cooperate) and so is the feedback. After any interaction its participants can file complaints against each other. The complaints are stored in the underlying DHT. It is assumed that after cheating in an interaction any peer will file a complaint against its partner, trying to hide in this way its own misbehavior. Then, the mentioned decision is made by analyzing and separating the probability distributions of the filed and received complaints of any peer. Though the technique provides a certain insight to risks associated with interactions, it is very questionable how precisely the information on whether a peer ever cheated or not can be used to make decisions on entering an interaction with it.

3.1 Discussion

3.1.1 Performance Analysis

It is somewhat hard to give precise judgment on the performance of the probabilistic estimation approaches because of the lack of informative simulation results in the papers we reviewed. Reference [15] assigns randomly probabilities of performing honest and misreporting own experiences, respectively. It tries to estimates the probabilities of honest performance and reports the average estimation error for various forms of collusion among peers. The error stays within several percents even when liars take up to 20–30% of the peer population and relatively small numbers of interactions per peer are considered (up to 30). As the most effective attack against the mechanism, [15] points out a simple collusive setting with two groups, liars and honest peers. Honest peers always report honestly, while liars invert the reports on the performances of the honest peers, and always report honest performance about the performances of the other liars.

In general, the assessment quality is highly dependent on the amount of information used by the mechanism. Probabilistic methods use relatively small amount of available reputation data, i.e., they do not pay attention to computing accurate estimates of the credibility of reports. This is generally different from what most of ad hoc methods do. We will return to this question in the next section, after we introduce ad hoc mechanisms.

3.1.2 Implementation Overhead

The reputation data on which all previously mentioned probabilistic approaches operate is localized around the target peer, i.e., the peer being assessed. Thus one can expect that the communication costs associated with the computation are low. [15] makes this precise by specifying that even a total of 30–40 reports on the performances of the target peer are enough for making good estimates. The computation overhead and storage costs are virtually negligible. This is to contrast with ad hoc methods, which typically use much larger amount of data, in many cases even the entire reputation graph.

However, probabilistic estimation techniques may span a larger fraction of the network if many levels of reports are considered. An interesting question to investigate with respect to this is evaluation of dependency of the estimation quality on the used fraction of available feedback.

4 Ad hoc Signaling Reputation Mechanisms

In this section, we describe a large group of works that approach the problem of P2P reputation management without making any explicit assumptions on the underlying peer behavior. This results in somewhat hard-to-understand semantics with respect to how the outputs of their corresponding algorithms should be used. Therefore, lacking a better term, we call this body of work ad hoc. Another important property of this class of solutions is that it typically aggregates all reputation information available in the formed social network (or at least a considerable amount thereof), as illustrated in Fig. 1. A natural interpretation of this process involves the following steps: (1) enumerating all paths from the trust computation source to the target node, (2) aggregating the trust values along the paths to give a path wide gossip and (3) merging these gossips into a final value. Thus particular attention is paid to computing credibility of available feedback and this is what brings serious benefits in terms of performance.

Reference [6] presents one of early examples.[2] It is one of rare works that makes explicit distinction between the contexts of recommendation and direct experiences, i.e., while many of the existing works judge on the credibility of feedback based on how well the feedback submitters provide the service itself, [6] introduces the context of recommendations to deal with credibility of feedback.

The feedback aggregation algorithm of [6] can be described as follows. Feedback is binary in the both contexts so that $W = \{r,d\} \times \{0,1\}$. The algorithm starts by aggregating all direct and recommendation interactions between neighboring nodes. Given its p "positive" direct experiences (i.e., p $(d,1)$ pairs) and no negative experience with peer j, peer i computes the "direct trust" toward j as $v_d^{ij}(p) = 1 - \alpha^p$, $0 < \alpha < 1$; having at least one negative experience with peer j, peer i should put $v_d^{ij} = 0$. On the other hand, given p positive and n negative experiences with a recommending peer k the recommendation trust of i toward k becomes $v_r^{ik}(p,n) = 1 - \alpha^{p-n}$ if $p > n$ and 0 otherwise. The exact value of parameter α was left unspecified. Thus the initial social network gets transformed so that for any ordered pair of nodes there are at most two edges between them, one per context, carrying these aggregated values. Further, the source node enumerates all paths to the destination node, selects only those such that the last hop carries direct trust (let us denote

[2] The work does not target large P2P networks. We selected it as an interesting illustrative example.

it v_k for a generic path *path i*) while all intermediate ones carry recommendation trust (denote them $v_1, \ldots, v_{k-1}$) and propagates the direct trust of the destination node through them according to the formula $v_{path\ i} = 1 - (1 - v_k)^{v_1 v_2 \cdots v_{k-1}}$. The path *path* 1 from Fig. 1 presents an example. Then, it groups together the paths with a common penultimate node (e.g., paths *path* 1 and *path* 2 from Fig. 1) and merges their trust values according to formula $v_{group\ j} = \sqrt[m]{\prod_{i=1}^{m}(1 - v_{path\ i})}$, where m is their number and $v_{path\ i}$ their values. Finally, it merges computed trust values of all groups $v = 1 - \prod_{l=1}^{s} v_{group\ l}$.A quick inspection of the described steps reveals that the output value $v \in [0, 1]$ and thus $T = [0, 1]$.

A simple analysis of the presented algorithm shows that, because all paths between the two concerned nodes must be explicitly taken into account, the algorithm complexity may be exponential in the number of the nodes in the network. This is not acceptable in P2P networks in which this number can easily reach the order of magnitude of millions. Note that we are not strict here in the sense that this explosion must happen if we consider *all* the paths. As we will see shortly, there is a way around this problem through a synchronous computation if the feedback is conveniently propagated and aggregated. However, the above algorithm does not fulfill this requirement and its computation complexity is in fact exponential, as the authors show.

Reference [43] offers a polynomial time feedback aggregation algorithm. Unlike [6], it does not consider the recommendation context separately. Again, direct experiences are rated binary (good and bad, $W = \{0, 1\}$). The feedback aggregation algorithm starts, just as in [6], with aggregating the interaction outcomes between the neighbors, which transforms the initial multigraph into a graph such that only one edge remains for any ordered pair of nodes. To this end, the authors propose a well known machine learning technique, delta learning namely, resulting in values between -1 and 1. To propagate the values over chains the authors use multiplication, with the constraint that trust over a chain with at least one negative link must be negative. When merging the values of multiple chains they choose only those chains carrying maximal values from the source to all the neighbors of the destination node. In the example from Fig. 1 only one of the two chains from peer i to peer u would be selected, the one having the larger associated value. The mean value of all these chains is selected as the algorithm output. Thus, the output values remain in the interval $T = [-1, 1]$. This computation is polynomial in the number of nodes (both finding all j's neighbors and selecting the maximum weighted chains toward those neighbors are polynomial operations and so is their union) and thus presents a considerable improvement as compared to [6].

4.1 A Synchronous Algorithm

Reference [38] offers interesting theoretical results relating the complexity of feedback aggregation to operations used in the aggregation.[3] It also proposes an

[3] [38] targets semantic web, but it is directly applicable to P2P networks as well.

algorithm to aggregate feedback. The algorithm is synchronous in the sense that it is done for all peers at once. It is important because it provides a feedback merging method with more efficient computation.

Assume that we have a social network with a feedback set $W \subset \mathbb{R}$, i.e., the recommendation context is left out, and that it is a pure graph, meaning that individual interactions between neighbors have been merged. At the moment it is not important how exactly this merging is done, any of the previously discussed strategies will do. The social graph can be also represented as a matrix $M \equiv [M_{ij}]_{i,j=1}^{N}$, where N is the number of peers. Assume further that the matrix has been normalized so that for any $1 \leq i, j \leq N$:

$$0 \leq M_{ij} \leq 1 \text{ and } \sum_{k=1}^{N} M_{ik} = 1.$$

Reference [38] defines two binary operators: $\circ$ and $\diamond$, denoting feedback concatenation (propagation) and feedback aggregation, respectively. The former is applied on two consecutive edges of a path in the social graph, while the latter applies to two distinct paths. One can think of ordinary multiplication and addition as examples of these operators. A new operator, denote it $\bullet$, is defined in terms of these two as follows: $C = A \bullet B$ such that $C_{ij} = \diamond(\forall k : \ A_{ik} \circ B_{kj})$, where A and B are matrices representing two social graphs. The easiest way to think of the new operation is to keep in mind that ordinary matrix multiplication is its special case provided that $\circ$ and $\diamond$ are ordinary multiplication and addition, respectively. When both A and B equal a matrix M, where M is the matrix representation of a given social graph, then C_{ij} is aggregated feedback on j as seen by i under the constraint that it is aggregated only across all paths of length 2.

Consider now the following simple algorithm:

$$Q^{(0)} = M, \ \ Q^{(k)} = M \bullet Q^{(k-1)} \ \text{ until } Q^{(k)} = Q^{(k-1)}. \tag{2}$$

It can be shown that the computation converges after a finite number of steps if the matrix M (or equivalently, the trust graph) is irreducible and aperiodic. Let $[Q_{ij}]_{i,j=1}^{N}$ be the limit. Further, [38] proves that if $\diamond$ is commutative and associative and $\circ$ is associative and distributes over $\diamond$, then the aggregated value of all paths (of any length) between any pair of peers i and j equals Q_{ij}.

Another interesting observation is that the computation can be performed locally, i.e., peers do not need to retrieve the whole matrix M first and then compute the needed power. After each iteration k, every peers can retrieve from its neighbors the currently computed opinions of those neighbors about all other peers and then do the computation of the step $k+1$. It turns out that this algorithm requires at most $O(N^3)$ computations per peer. However, it is not clear what should be the overall latency the algorithm introduces because it is determined by the number of iterations and this number is in turn affected by the network graph connectivity.

There are a number of other works which can be considered as special cases of [38]. Reference [33], used as Google's method for Web pages ranking, and [24], targeting P2P networks specifically, present two prominent examples. They both use the ordinary matrix multiplication as the matrix operation from (2).

Similar ideas are followed in [42]. Reference [42] avoids aggregation of the individual interactions (transforming the social network into an ordinary graph). Instead, it operates on the multigraph directly. It does not consider the recommendation context and uses ratings from the interval $[0,1]$, so again $W = [0,1]$. The main idea is to aggregate feedback on a given peer by computing a weighted average in which weights represent the aggregated feedback on the feedback originators themselves. To see this, consider the following formula:

$$t_j = \sum_{e \in incoming(j)} w_e \frac{t_{source(e)}}{\sum_{f \in incoming(j)} t_{source(f)}}, \tag{3}$$

where $incoming(j)$ is the set of all edges ending at node j, w_e is the feedback belonging to the edge e and $t_{source(e)}$ should be understood as credibility of the originator of this feedback. As the authors claim, this formula can be computed by using an iterative computation, similar (but still different) to (2). As such, it suffers from more or less the same problems: the computation is inefficient and the whole network must be retrieved. But, the authors also develop a simple caching scheme in which the credibility values of the feedback originators are taken from a cache (default values are used in the case of cache miss) and the computed value of a peer replaces its corresponding cache value.

Building on [12, 44], the same work ([42]) proposes another approach based on so called collaborative filtering technique. The idea is very similar to the previous one, the only change is that the weights in (3) are replaced with “similarity coefficients” between the raters and the computing peer. They are computed by finding a common set of peers that interacted with the computation source and a given rater and calculating the standard deviation between the two corresponding vectors.

4.2 Discussion

Consider algorithm (2) and assume that the operations $\diamond$ and $\circ$ are ordinary addition and multiplication so that $\bullet$ is the ordinary matrix multiplication. As well, assume that the graph is irreducible and aperiodic so that the computation converges for sufficiently large k. It turns out that the result is a matrix in which all the rows are the same and sum up to 1. In parlance of matrix calculus, this is the primary Eigenvector of matrix M. Applied to our problem, this result can be interpreted as follows. As the aggregated feedback of every peer is independent of the computation source, it can be seen as a value that the community as a whole assigns to the peer. The fact that all the values sum up to 1 has an even more interesting interpretation: It appears that the community distributed “a unit amount of trust” among the peers. An apparent problem with this is that if we have the value for only one peer (suppose that we used some approximate method to compute it or simply evaluated all the paths to the target) we will be in trouble with interpreting it. Does it mean that the value is low because the concerned peer has bad feedback on average or because it has excellent feedback but had to "share the trust" with other peers, which have similar feedback. There is another problem, even if we have the values for all the peers, but they are approximately close we are in doubt whether the whole network is good or bad.

A similar problem is present in other methods described in this section. Assume that we have a value, say 0.3, as the aggregated feedback value of a peer. The problem is how can we use this value to decide whether to interact with this peer or not? Higher values should imply better service providers, but is this particular value high enough for us? From the way how the value has been computed in any of the above methods it is clear that it cannot be interpreted as the (estimated) probability of specific behavior of the target peer. So, what does the value actually represent? The lack of a plausible answer to this question is what all the discussed approaches have in common. Precisely, the computed values lack a plausible interpretation on an absolute scale and therefore they can be used only in scenarios which involve ranking the trust values of many peers and selection of the most trustworthy one(s) among them.

4.2.1 Performance Analysis

Most of the works mentioned in this section do not build their feedback aggregation algorithms explicitly on the assumption that peer behavior is probabilistic. However, they all make this assumption when evaluating the algorithms. Thus judging on their performances requires evaluations in settings with various types of behavior represented by different probability distributions, including uncoordinated (i.e., independent) misbehavior of the peers as well as forming collusive groups of different sizes and varying collusion patterns. References [24, 42] offer informative simulations, a condensed view of which we now present. As presented aggregation algorithms output a single value, the only meaningful setting in which they can be applied requires two possible outcomes of interactions. This is reflected in the terminology we follow, peers can be either malicious or trustworthy.

Both [24, 42] report good performance when the fraction of malicious peers is small (below 45% approximately) and they act independently of one another. The fraction of misclassified peers remains almost constant and very low (within 5%) in this case. The approximate computation of [42] exhibits very similar behavior, the only difference is that the rise of the misclassification rate falls into a wider region starting at around 35%. Reference [42] reports the complete breakdown of both the original mechanism and the approximate one when malicious peers take more than a half of the population or when they collude. On the contrary, [24] claims almost full effectiveness of their mechanism even in this case.

This is partly due to the assumption that a number of peers exist each of whom is assigned some non-zero rating by the rest of the community, including the malicious peers. Thus, there must be an arc with a non-zero weight leading from any node in the social graph to each of the nodes representing these peers. While this assumption can be simulated when the computation is performed centrally at one peer, after entire feedback in the network has been retrieved, we do not see a clear way to enforce this behavior in a distributed computation. Reference [24] also analyzes various collusion scenarios and identifies the following as the most effective one. Malicious peers split into two groups. Peers from the first group cheat always, while the peers from the other group never cheat in direct interactions and give high ratings to the peers from the first group. We add that such collusion scenarios can be defeated only by separating the contexts of recommendations and direct service provisions.

Reference [42], the similarity based technique, may offer a solution to this problem. This scheme stays effective even when majority of the peers are malicious and they form collusive groups that make fake interactions. It would be interesting to check its performance in presence of the mentioned collusive behavior.

4.2.2 Implementation Overhead

In the most of the mentioned approaches, all available information in the network is used when aggregating the feedback about a single peer. This immediately implies that all the peers in the network are affected and that the associated communication costs are high. Further, some of the discussed methods require exploring all the paths between two given nodes incurring thus a high computation overhead. We see this as the most serious obstacle to their usability in practice in the context of P2P systems, although they can be applied to smaller settings without any change. The approximate method of [42] imposes acceptable overhead as only the direct witnesses of the target peer's performance are involved in the computation.

References [24, 38] propose a synchronous algorithm with a polynomial complexity. However, even if we accept its total overhead, it is unclear if the algorithm as specified is feasible in a P2P network. Peer dynamics may impose a serious problem. Peers go offline and come online at unpredictable time instants, the social network keeps changing constantly and the recomputation of the algorithm triggered by such changes may be highly impractical. Instead, we believe that an incremental computation is something worth further investigation. Caching schemes of [42] offer important insights with respect to this.

4.3 Bio-inspired P2P Reputation Systems

We now briefly touch upon a growing body of work that emerged in the recent several years and tries to transfer behavior observed among biological species to computing systems. In the light of the categorization of the P2P reputation systems established earlier, these works do not constitute a group on their own, but rather belong to the ad hoc methods. Therefore we discuss them in this section.

To illustrate these works, consider [21] as a typical representative. Reference [21] looks at the process of searching for reputable provider of a service in a P2P network as being equal to the process used by ants when searching for food. Ants leave pheromones at specific places to guide other ants finding better routes. The more visited a route is, the greater amount of pheromone it will have and consequently, more ants will follow that route. In a similar vein, when a peer requests a service it first contacts a neighbor that it believes is on the path toward a reputable service provider. This process is repeated at each hop, until a provider is found. Once the service has been provided and evaluated, feedback originating at the requesting peer gets disseminated along the same path the original request took, allowing the nodes on the path to update their beliefs about the neighbors.

We recommend [40, 41] as two further bio-inspired reputation systems and [11] as an extensive overview of bio-inspired algorithms in general.

5 Sanctioning Reputation Mechanisms

Sanctioning mechanisms assume that behavior is dynamic, i.e., actions taken by an agent at two different time instants are correlated. This correlation is naturally present in game theoretic reputation models, through the main game-theoretic assumption of rational behavior. Every player has a set of strategies and every strategy profile (i.e., combination of strategies selected by the players) generates an outcome of the game. A utility value is assigned to each outcome and the goal of each player is to maximize her own utility.

The game theoretic framework for analyzing reputation is that of repeated games in which some players are uncertain about payoff structures of their opponents. The right game-theoretic tool to model mentioned uncertainties is that of games with *incomplete information* (Bayesian games), in which different players may have different information about some important aspects of the game. Central to Bayesian games are *types* of the players by which the players' private information is modeled. In most of the models the types simply correspond to different payoff structures of the players, but this needs not always be the case.

Repeated play is modeled by well studied models of repeated games, in which a given stage game is played many (finitely or infinitely) times and the players maximize their long-run payoffs, i.e., in the course of the entire repeated game. We will focus here only on so called δ-discounted ($0 < \delta < 1$) repeated games in which any k-stage payoff is δ discounted as compared to that of the stage $k-1$. Generally, the set of players may vary among the stages, but the following two extreme cases are most studied in the literature: (1) models in which all the players stay active at each stage (we call these players long-run) and (2) the models in which one or more players stay active in each stage (long-run players) while the others play at one stage only (short-run players). The latter is more important to us, as large numbers of participants in today's online groups make repeated interactions between the same players highly improbable. Our main concern in the rest of the section will be to show how specific behavior of the long-run player can be induced by appropriately aggregating feedback on his past play and making it available to the other players.

An important assumption we make in this section is that the reader is familiar with the basics of game theory and the its main concepts such as Nash (as well as Nash-Bayesian) equilibrium, repeated games and so on.

5.1 Modeling Reputation

We now show how game theory formalizes the concept of reputation. Figure 3 (adapted from [26]) presents the working example. The figure shows two game trees corresponding to two types of one of the players (monopolist). Each game tree is an example of so called games in extensive form (or dynamic games). Nodes labeled *i* belong to player *i* and only that player moves at those nodes. Possible moves at each node are shown as labels of the edges leaving the node.

The game from Fig. 3 models a situation two firms might face in a market. Assume that player 1 is a firm already established in a market (monopolist hereafter) and that player 2 is another firm that decides on entering the market or staying out (entrant). The game is Bayesian as the monopolist has two types, it can be weak or strong, but the entrant is uncertain about which of these two types it is actually facing. Notice that the only difference between the two game trees is in the payoffs of the monopolist. In both cases the monopolist can choose to fight the entry or acquiesce (share the market peacefully), while the entrant chooses whether to enter or stay out. The payoffs of the two firms are as shown in Fig. 3. It is important to notice that the strong monopolist prefers fighting the entry ($1 > 0$, the right-hand side of the figure) and the entrant prefers staying out if he believes that the fight will occur ($0 > -1$). If the entrant stays out the monopolist gets the best payoff he can get in the game.

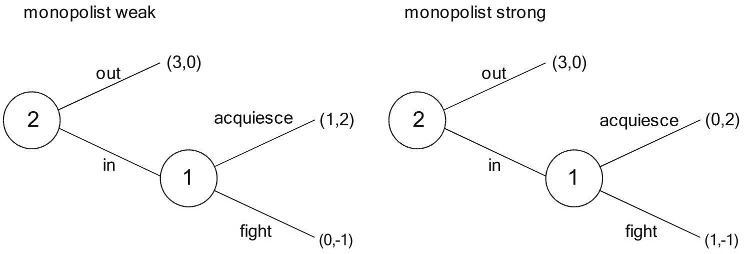


Fig. 3 Bayesian chain store game

Assume now that a *weak* monopolist plays this stage game against a finite number of entrants each of whom plays only once but is informed about the whole previous play. The entrants assign a common prior probability distribution to the set of the monopolist's types (weak and strong). After each stage the entrants update their beliefs on the types using Bayes' formula and the monopolist's that stage move. We are interested in the following problem: Does it pay off for the monopolist to develop reputation for being strong (remember that it is in fact weak), i.e., fight eventual early entries in order to deter later ones? Intuitively, this might make sense because not sharing the market in later stages may offset losses from fighting early ones.

[26] provide an elaborate analysis of the problem. First, if all the entrants know that the monopolist is weak (so the prior probability of being strong equals 0 – the game is no longer Bayesian), then there is only one equilibrium of the repeated game in which all entrants enter and the monopolist acquiesces at all stages. So, a monopolist that is known to be weak will never fight an entry under these conditions. Put differently, the monopolist cannot develop reputation for being strong when its opponents cannot believe that it can be strong. But, as soon as the prior probability on the monopolist being strong is non-zero there are equilibria in which the monopolist would prefer to fight some early entries and the affected entrants would consequently stay out (so fighting would not actually happen as the entrants stay out). This is exactly the effect of the monopolist's reputation, or more precisely the possibility to develop it. The point is that fighting these early entries would convince future entrants that the monopolist is strong so that losses incurred by these moves would be offset by not sharing the market at later stages. Therefore, the entries might occur only in a number of last stages. Interestingly, this number depends only on the initial probability that the monopolist is strong, i.e., it is independent of the total number of stages played. Thus, the payoff the monopolist receives when the total number of stages grows approaches 3, the payoff corresponding to the monopolist's most preferred stage game outcome, when the entrants stay out.

Reference [18] provides a generalization of this finding. Similar to [26], the setting involves one long-run player facing an infinite sequence of short-run opponents. The key notions [18] introduces are Stackelberg payoff and Stackelberg strategy. Stackelberg payoff is the best payoff the long-run player can get by committing himself to a specific strategy in the considered stage game (at this point, the stage game is not Bayesian). The strategy to which it would commit is called Stackelberg strategy. The authors form a Bayesian game setting in which the long-run player has two types: one is the ordinary long-run player with payoffs as given in the stage game, the other is a new type of the long-run player for which playing Stackelberg strategy is the dominant strategy. The main result of [18] is that, under these conditions, the limit of the long-run player's payoff in the repeated game when his discount factor δ approaches 1 (the long run player is sufficiently patient) is exactly his Stackelberg payoff. Applied to the above example, this means that a monopolist with a sufficiently high discount factor can always get his Stackelberg payoff (which equals 3) when playing against an infinite sequence of short-run entrants.

5.1.1 Perfect vs. Imperfect Monitoring

In the above models, the necessary ingredient to talk about reputation effects was incomplete information, modeled as Bayesian game. Interesting models, perhaps even closer to the real-world settings, can be obtained if uncertainties on the short-run players' side are modeled through so called imperfect monitoring. So far, the short-run players were perfectly informed about the history of play. Such games are said to have perfect monitoring. When players do not observe the past actions but only their imperfect signals, the game is said to have imperfect monitoring. If the signals are common to all the players (i.e., they are public), then the monitoring is said to be public, otherwise it is private.

Reference [13] presents an interesting model with private monitoring. The setting involves a long-run seller facing an infinite sequence of one-shot buyers. In each round, the seller sells an item and the buyers compete in an auction to buy it. The item is perceived on the buyers' side as either high or low quality. This perception is influenced by the seller's action, which can be either high or low effort. Both low and high quality perceptions of the buyers are possible under both high and low effort. Thus the game has private imperfect monitoring as the seller's actions are not observed by the buyers. Instead, their imperfect signals (high and low quality) are. The feedback on the seller's past play consists of the observed signals. Reference [13] proposes maintaining a fixed size window of such reports as the feedback aggregation strategy. Every new report replaces an old, randomly chosen one. The seller's optimal strategy in this setting is to always exert high effort if he has zero low quality reports, otherwise he should follow a mixed strategy in which the probability of exerting high effort is a linearly decreasing function of the current number of low quality reports. However, it turns out that there is an efficiency loss in comparison with the case where the seller can commit to exerting high effort. But there is no efficiency loss in comparison with the case where the entire feedback history of the buyer reports is publicly known. The author also shows that the whole mechanism is pretty robust in the presence of buyers' misreporting the seller's actions.

Even though the difference between the information structure games with perfect and imperfect monitoring seems small, it causes a large difference in the tractability of these two classes of games. While convenient mathematical tools for analyzing the imperfect public monitoring games have been found [3] and the long-run player's payoff characterization results have been obtained [19] there are no similar results for the imperfect private monitoring. Instead, only specific games have been analyzed and their equilibria derived. See [25] for more on this.

5.1.2 Truthful Feedback Submission

There are a number of problems with the above models that have to be addressed. In the models of this section, we have tacitly assumed that feedback is always provided. Each short-run player was assumed to submit her feedback on the long-run player's move at the concerned stage. Is this a valid assumption, given that all our players are rational and feedback submission is costly, i.e., it is not free? We have also assumed that short-run do not misreport on their experience with the long-run player. What is the impact of this assumption? Finally, to whom should the feedback be submitted in a P2P system, do we need to have a central authority for that purpose or we can solve the problem in a distributed manner? We give answers to the first two questions in the next paragraph, while we touch upon the last question in Section 5.1.4.

The approach taken in models from Sections 3 and 4 was that they permit misreporting and then verify how much it hurts the reputation system performance. Game-theoretic models take a different approach. They try to make truthful feedback provisioning the dominant strategy of the short-run players. [23, 29] analyze exactly this problem and propose a payment-based system based on the application of proper scoring rules to the reports. They prove that honest reporting is a Nash equilibrium and that, on the other hand, the budget is balanced by charging involved players for payments that are based on the reports of others.

Further references on the topic are [12, 34]. Particularly interesting is [34]. It assumes that both the service provider and the client rate the provider's performance after each interaction. In case of disagreement of ratings, it proposes fines that affect reputation of both interacting partners. Honest feedback submission has been found to be a Nash equilibrium.

5.1.3 Evolutionary Settings

Recently, there has been a growing interest in so called evolutionary game-theoretic models. In an evolutionary setting, a population of players play a given game repeatedly for many epochs. Each player is characterized by its strategy. Upon the completion of one epoch, the payoffs of all players are calculated and the population is adjusted for the next epoch so that good strategies reproduce while poor ones die out. The rates of reproduction depend on the exact scores.

Reference [5] analyzes such a setting with Prisoner's dilemma as the stage game. It has been found that the best strategy in the long-run is tit-for-tat. It is remarkably simple: it starts with cooperating and afterward does whatever the opponent did in the previous round. Thus reciprocation is one of its main properties. The setting for experiments in [5] was that in each epoch the same two players played against each other. Would the main result change if we drop this assumption and assume that the opponents are matched randomly? This would even better approximate real-world settings.

Reference [32] offers an answer (it is neatly summarized in [31]). It assumes that a public label is associated with each player. This label constitutes the reputation of the player. Everyone can read the label, and all players except the owner of the label are allowed to change it. Labels are changed after each interaction. Updating labels of opponents brings in another dimension in the strategy space. Reference [32] describes it by means of the assessment function. The assessment function takes the label of self, the label of the opponent and the action of the opponent and produces the new value for the opponent's label. On the other hand, the action function maps the label of self and the opponent to an action, either cooperate or defect. There are 16 possible action functions and 256 possible assessment functions, giving rise to 4096 possible strategies. Reference [32] identified 8 of them as evolutionary stable. Interesting enough, all of them are remarkably similar to tit-for-tat. This is why [31] points out indirect reciprocity as the keyword for this setting.

5.1.4 Critiques of Game-Theoretic Modeling

We now briefly discuss some pros and cons of game-theoretic modeling of reputation. First, feedback needs to be maintained and aggregated (labels of the players, fixed window of reports, and so on). We did not make it clear who does that and how. There are P2P settings in which this does not pose a problem. For example, distribution of every file in BitTorrent [10] is monitored by a tracker. This is an entity that represents a perfect candidate to perform feedback aggregation when a reputation system is implemented to support distribution. However, it is not clear how such an entity could be implemented in fully decentralized systems or for different tasks such as DHT routing. When a centralized entity is not available, the only alternative is to implement it in a distributed manner. However, this would lead to a substantial increase of model complexity as strategic choices with respect to feedback aggregation must not be viewed in separation from the rest of the system.

The next problem is related to the rationality assumption. Is that assumption valid in a typical P2P setting? There are studies demonstrating that humans do not behave as fully rational decision makers [7, 16]. So if humans (as opposed to software agents, for example) are expected to be the participants in the system, then a game-theoretic model of reputation may fail to be effective.

6 Identities: An Important Practical Problem

All the models discussed so far implicitly assume that identities are stable and unchangeable. Peers can be unambiguously authenticated and they never leave the system and reappear under new identities. This is highly unrealistic in P2P settings. As the following discussion shows, there is always a price to pay when stable identities cannot be enforced.

Reference [17] is one of rare works dealing with the problem of unstable identities. It develops a reputational game-theoretic model in which player play the prisoner’s dilemma repeatedly. After each stage, they can choose whether to replace their identities and erase the reputation they built until that point or just to continue to play with the old identity. The main result is that no equilibrium can be significantly more efficient than the equilibrium in which newcomers pay dues, i.e., they are treated poorly by the players with established good reputations. This means that if it is possible to change identities, then there must be an efficiency loss. When applied to signaling approaches, this translates to the following reasoning. The best a peer can do is setup a threshold of minimum reports other peers need to have in order to be selected as service providers. However, this creates a new problem of bootstrapping the system.

Another problem is indirectly related to identities. It deals with a range of simple strategies to seriously reduce the effectiveness of reputation systems [12]. For example, a peer may collude with another peer or a group of them in order to be given sufficiently many high ratings. The easiest way to realize this is setup a number of side identities and use them to rate the main identity. Reference [12] presents an effective way to reduce the effects of this problem. It implies taking only feedback from a set of raters which have rated a set of common peers as the concerned peer and also gave them similar ratings. To be effective in this case, the adversary needs not only make multiple identities but also rate sufficiently many other peers.

7 Discussion and Conclusions

In this chapter, we addressed the problem of reputation management through social networking. We started with a motivation for the work and pointed out that reputation management is needed in many settings as the only viable solution to promote trustworthy behavior. Further, we found that empirical studies confirm the effectiveness of reputation systems.

Social aspect is inherent to reputation systems. We formalized this through the notion of a social network. The interactions among peers together with the feedback the interacting peers make about one another form a social network. The data available in the social network as well as the algorithms to aggregate this data are the basis for building reputation systems. The P2P dimension of the problem lies in being aware that social networks can be huge and must be distributed among the peers. Therefore, algorithms to aggregate available feedback must be scalable and operate on a small fraction of available data.

We provided a classification of existing solutions. It is based on the assumptions various approaches make about underlying behavior and the interpretation of their outputs. Signaling reputation systems output signals that can be at best interpreted as probabilities of likely behavior of concerned peers. This is truly the case with probabilistic signaling approaches, that start with the assumption that peers are of certain types determining their probability distributions. A broad range of other methods, which we termed ad hoc, do not make this assumption explicit but the outputs they make can be most reasonably interpreted this way. Sanctioning reputation mechanism assume economic rationality as the defining property of the peers and build game theoretic models in which maximization of the long-run profits is the main goal. Different strategies to maintain and share the information about past play lead to different outcomes, i.e., equilibria. Finding those strategies that lead to desirable outcomes is the main problem.

However, there are no studies confirming that behavior in real-world online communities is probabilistic or rational. In fact, there are studies confirming that it does not belong to either of these two extremes. This is what we believe future research should focus on, i.e., deriving accurate approximations of the true underlying behavior and making models tailored specifically for that type of behavior.

References

1. Aberer, K.: P-Grid: A self-organizing access structure for P2P information systems. In: Proceedings of the Sixth International Conference on Cooperative Information Systems (CoopIS 2001). Trento, Italy (2001)
2. Aberer, K., Despotovic, Z.: Managing Trust in a Peer-2-Peer Information System. In: Proc. of the IX International Conference on Information and Knowledge Management. Atlanta, Georgia (2001)
3. Abreu, D., Pearce, D., Stachetti, E.: Toward a Theory of Discounted Repeated Games with Imperfect Monitoring. Econometrica **58**, 1041–1064 (1990)
4. Akerlof, G.: The market for lemons: Quality uncertainty and the market mechanism. Quarterly Journal of Economics **84**, 488–500 (1970)
5. Axelrod, R.: The Evolution of Cooperation. Basic Books, New York (1984)
6. Beth, T., Borcherding, M., Klein, B.: Valuation of Trust in Open Networks. In: Proceedings of the European Symposium on Research in Computer Security (ESORICS), pp. 3–18. Springer-Verlag, Brighton, UK (1994)
7. Bolton, G., Katok, E., Ockenfels, A.: How effective are online reputation mechanisms? an experimental investigation. Discussion paper 25, 2002, Max Planck Institute for Research into Economic Systems, Germany (2002)
8. Buchegger, S., Le Boudec, J.Y.: The effect of rumor spreading in reputation systems for mobile ad-hoc networks. In: Proc. of WiOpt ‘03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks. Sophia-Antipolis, France (2003)
9. Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S.: Making Gnutella-like P2P Systems Scalable. In: SIGCOMM 2003. Karlsruhe, Germany (2003)
10. Cohen, B.: Incentives Build Robustness in BitTorrent. In: 1st Workshop on the Economics of Peer-to-Peer Systems. Berkeley, CA, USA (2003)
11. Cordon, O., Herrera, F., Stützle, T.: A review of the ant colony optimization metaheuristic: Basis, models and new trends. Mathware & Soft Computing **9**, 141–175 (2002)
12. Dellarocas, C.: Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In: Proceedings of the 2nd ACM conference on Electronic Commerce. Minneapolis, USA (2000)
13. Dellarocas, C.: Sanctioning reputation mechanisms in online trading environments with moral hazard. Working paper 4297-03, MIT Sloan (2004)
14. Despotovic, Z.: Building Trust-aware P2P Systems: From Trust and Reputation Management to Decentralized E-Commerce Applications. Ph.D. thesis, École Polytechnique Fédérale de Lausanne (EPFL) (2005)
15. Despotovic, Z., Aberer, K.: Maximum Likelihood Estimation of Peers’ Performances in P2P Networks. In: 2nd Workshop on the Economics of Peer-to-Peer Systems. Cambridge, MA, USA (2004). Available at http://www. eecs.harvard.edu/p2pecon/
16. Fehr, E., Gächter, S.: Altruistic punishment in humans. Nature **14**, 137–140 (2002)
17. Friedman, E., Resnik, P.: The Social Cost of Cheap Pseudonyms. Journal of Economics and Management Strategy **10**(2), 173–199 (2001)
18. Fudenberg, D., Levine, D.: Reputation and Equilibrium Selection in Games with a Patient Player. Econometrica **57**(4), 759–778 (1989)

19. Fudenberg, D., Levine, D.: Maintaining a Reputation when Strategies are Imperfectly Observed. Review of Economic Studies **59**, 561–579 (1992)
20. Gnutella: Clip2. The gnutella protocol specification v0.4 (document revision 1.2). http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf (2001)
21. Gómez Mármol, Félix and Martínez Pérez, Gregorio and Gómez Skarmeta, Antonio F.: TACS, a Trust Model for P2P Networks. Wireless Personal Communications, Special Issue on Information Security and data protection in Future Generation Communication and Networking (2008)
22. Houser, D., Wooders, J.: Reputation in auctions: Theory and evidence from ebay. Working paper, University of Arizona (2001)
23. Jurca, R., Faltings, B.: Towards Incentive-Compatible Reputation Management. In: AAMAS 2002 Workshop on Deception, Fraud and Trust in Agent Societies. Bologna, Italy (2002)
24. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: EigenRep: Reputation Management in P2P Networks. In: Proceedings of the World Wide Web Conference. Budapest, Hungary (2003)
25. Kandori, M.: Introduction to Repeated Games with Private Monitoring. Journal of Economic Theory **102**, 1–15 (2002)
26. Kreps, D., Wilson, R.: Reputation and imperfect information. Journal of Economic Theory **27**, 253–279 (1982)
27. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: International Conference on Supercomputing, pp. 84–95. New York, USA (2002)
28. Melnik, M.I., Alm, J.: Does a Seller's Ecommerce Reputation Matter? Evidence from eBay Auctions. Journal of Industrial Economics **50**(3), 337–349 (2002)
29. Miller, N., Resnick, P., Zeckhauser, R.: Eliciting Honest Feedback in Electronic Markets. Working paper, SITE02 workshop (2002). Available at: http://www.si.umich.edu/ presnick/-papers/elicit/
30. Mui, L., Mohtashemi, M., Halberstadt, A.: A Computational Model of Trust and Reputation. In: Proceedings of the 35th Hawaii International Conference on System Science (HICSS). Hawaii, USA (2002)
31. Nowak, M., Sigmund, K.: Evolution of indirect reciprocity. Nature **437**, 1291–298 (2005)
32. Ohtsuki, H., Iwasa, Y.: How should we define goodness? - reputation dynamics in indirect reciprocity. Journal of Theoretical Biology **231**, 107–120 (2004)
33. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Tech. rep., Stanford University, Stanford, CA, USA (1998)
34. Papaioannou, T.G., Stamoulis, G.D.: Achieving Honest Ratings with Reputation-based Fines in Electronic Markets. In: Proceedings of the IEEE Infocom Conference. Phoenix, AZ, USA (2008)
35. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Proceedings of ACM SIGCOMM '01, pp. 161–172 (2001)
36. Resnick, P., Zeckhauser, R.: Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. In: M.R. Baye (ed.) The Economics of the Internet and E-Commerce, *Advances in Applied Microeconomics*, vol. 11. Amsterdam, Elsevier Science (2002)
37. Resnick, P., Zeckhauser, R., Friedman, E., Kuwabara, K.: Reputation systems. Communications of the ACM **43**(12), 45–48 (2000)
38. Richardson, M., Agrawal, R., Domingos, P.: Trust management for the semantic web. In: Proceedings of the Second International Semantic Web Conference, pp. 351–368. Sanibel Island, FL (2003)
39. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In: Proceedings of the 2001 ACM SIGCOMM Conference, pp. 149–160 (2001)
40. Tajeddine, A., Kayssi, A., Chehab, A., Artail, H.: Patrol: a comprehensive reputation-based trust model. International Journal of Internet Technology and Secured Transactions **1**(24), 108–131 (2007)

41. Wang, W., Zeng, G.: AntRep: A swarm intelligence based trust model in P2P networks. Journal of Universal Computer Science **13**, 1164–1174 (2007)
42. Xiong, L., Liu, L.: Peertrust: Supporting reputation-based trust in peer-to-peer communities. IEEE Transactions on Knowledge and Data Engineering (TKDE), Special Issue on Peer-to-Peer Based Data Management **16**(7), 843–857 (2004)
43. Yu, B., Singh, M.P.: A Social Mechanism of Reputation Management in Electronic Communities. In: Proceedings of the 4th International Workshop on Cooperative Information Agents (CIA), pp. 154–165. Boston, USA (2000)
44. Zacharia, G., Moukas, A., Maes, P.: Collaborative Reputation Mechanisms in Electronic Marketplaces. In: Proceedings of 32nd Hawaii International Conference on System Sciences. Hawaii, USA (1999)

State of the Art in Trust and Reputation Models in P2P networks

Félix Gómez Mármol and Gregorio Martínez Pérez

Abstract Ensuring security in a distributed environment such as P2P networks is a critical issue nowadays. Nevertheless, it is in those kind of scenarios in which entities can enter or leave the community whenever they want, where traditional security schemes can not always be applied. Specifically, the use of a PKI (Public Key Infrastructure) may be unacceptable within highly distributed systems. Therefore, modeling concepts like trust and reputation may result very helpful and useful when trying to gain a certain level of security and confidence among inter-operating entities. Thus, this chapter presents a review of some of the most representative trust and reputation models for P2P networks, discussing their main characteristics and also their weaknesses and deficiencies. Open issues and challenges associated with them will be also covered.

1 Introduction

P2P networks have been widely spread in the recent years. We find them in many scenarios and applications, from file sharing systems to even military environments. They have helped to improve and increase the accessibility to the information as well as the opportunity of communication or the performance of electronic transactions, for instance.

Consequently, many research works have been done and are still in progress in order to improve their robustness, applications, scalability or security among other features.

And it is just security one of the most critical issues when dealing with this kind of networks. Due to its intrinsic nature, where every peer can enter and leave the network whenever it wants and where most of the times there is no central entity controlling the community, many security threats have arisen during the development of P2P networks, since malicious behaviors must be managed by all the participants themselves.

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_26, © Springer Science+Business Media, LLC 2010

Some of these threats have their origin in the lack of knowledge a peer has (specially newcomers, who join the network for the first time) about the other peers belonging to the community, and their current and past behavior.

Without loss of generality we can assume a P2P network where some nodes offer certain services or carry out certain tasks and other peers apply for those services or tasks. In such a situation when a newcomer enters the network she will probably not know any or most of the rest of peers already joined, and vice versa, i.e., they will recognize her as a stranger.

In the last few years trust and reputation management has been proposed as a novel approach in order to overcome this problem. In fact, by counting with a system where every node could ask for the reputation hold by a peer in the community, more accurate and intelligent choices could be done when deciding which peer to interact with since, in many cases, fraudulent interactions could be avoided.

And this reputation values come from the trust relationships established among peers who have had interactions in the past and have evaluated and rated each others. However, the application of these trust and reputation schemes also has its security threats. For instance, a collusion could be formed among a set of peers who rated themselves with the maximum value, increasing this way their reputation in the community.

Therefore, in this work we are going to present and describe a set of some of the most representative trust and reputation models for P2P networks currently published in the literature and we will describe how each one of them deals with these and other problems.

The rest of the paper is organized as follows. The trust and reputation models review will be done in Section 2, while Section 3 will show a global analysis of all the previously described models, talking about common features and deficiencies detected in most of them, extracting the main steps followed by the majority of those models and presenting common issues related to trust and reputation management and how the current proposals tackle such problems. Finally Section 4 will present some conclusions of our analysis, as well as some future research directions.

2 Trust and Reputation Models

CuboidTrust

CuboidTrust [1] is a global reputation-based trust model for peer to peer networks which builds four relations among three trust factors including contribution of the peer to the system, peer's trustworthiness (in reporting feedbacks) and quality of resource. It applies power iteration in order to compute the global trust value of each peer.

A cuboid is built where each small cube with coordinates $(x,y,z,)$, denoted by $P_{x,y,z}$, represents the quality of resource z stored at peer y rated by peer x. Once peer

x has downloaded resource z from peer y, it may rate it as positive ($P_{x,y,z} = 1$) if the downloaded resource z is considered authentic, or negative ($P_{x,y,z} = -1$) if the downloaded resource z is considered inauthentic or the download is interrupted for any reason.

Two coefficient matrixes are defined, E and D, whose elements are:

$$D_{ij} = avg(P_{j,i,\cdot}) \in [-1,1] \qquad 1 \le i \le M,\ 1 \le j \le M$$

$$E_{ij} = avg(P_{i,\cdot,j}) \in [-1,1] \qquad 1 \le i \le M,\ 1 \le j \le N$$

where M is the number of peers, N is the number of distinct resources, $P_{j,i,\cdot}$ represents the vector with $X = j$ and $Y = i$ in the cuboid, and $P_{i,\cdot,j}$ represents the vector with $X = i$ and $Z = j$ in the cuboid. Therefore, each element D_{ij} stores the average score of peer i rated by peer j while each element E_{ij} stores the average score of resource j rated by peer i.

The first relation among the three mentioned factors effectively combines two of them (the trustworthiness score and the contribution score) as it is shown in the following equation:

$$C_i = \sum_{j=1}^{M} (D_{ij} \times T_j) \qquad 1 \le i \le M$$

where T_j represents the trustworthiness of peer j, and C_i reflects the contribution of peer i to the system by considering the experiences of all peers belonging to the network.

The second relation combines the quality of a resource with the trustworthiness of a peer as follows:

$$T_i = \sum_{j=1}^{N} (E_{ij} \times Q_j) \qquad 1 \le i \le M$$

where Q_j is the quality score of resource j, and T_i actually represents the trustworthiness of peer i.

The third relation takes into consideration the trustworthiness of the peers in the system as well as the quality of the resources being exchanged in the following manner:

$$Q_i = \sum_{j=1}^{N} (E^T_{ij} \times T_j) \qquad 1 \le i \le N$$

where E^T_{ij} is the element with coordinates (i,j) of the transposition of matrix E, denoted as E^T, and represents the average score of resource i rated by peer j. T_j indicates the trustworthiness of peer j and Q_i is the quality of resource i in the system.

And the last relation combines the contribution and the trustworthiness as follows:

$$T_i = \sum_{j=1}^{M} (D_{ij}^T \times C_j) \qquad 1 \leq i \leq M$$

where D_{ij}^T represents the average score of peer j rated by peer i, C_j indicates the contribution score of peer j, and T_i reflects the trustworthiness of peer i.

Combining now these four relations we can obtain:

$$\begin{aligned} &C = D \times T = D \times E \times Q = D \times E \times E^T \times T = \\ &D \times E \times E^T \times D^T \times C = (D \times E) \times (D \times E)^T \times C \end{aligned} \tag{1}$$

And applying power iteration over equation (1), we get

$$C^{(k)} = R \times C^{(k-1)} = R^{(1)} \times C^{(k-2)} = \cdots = R^{(k-1)} \times C^{(1)} = R^{(k)} \times C^{(0)}$$

where $R = (D \times E) \times (D \times E)^T$, and $C^{(k)}$ represents the global contribution score of every peer in the system, after k iterations.

Another combination of the four relations can give us the following expression:

$$\begin{aligned} &Q = E^T \times T = E^T \times D^T \times C = E^T \times D^T \times D \times T = \\ &E^T \times D^T \times D \times E \times Q = (D \times E)^T \times (D \times E) \times Q \end{aligned} \tag{2}$$

And applying again power iteration over equation (2), we get

$$Q^{(k)} = S \times Q^{(k-1)} = S^{(1)} \times Q^{(k-2)} = \cdots = S^{(k-1)} \times Q^{(1)} = S^{(k)} \times Q^{(0)}$$

where $S = (D \times E)^T \times (D \times E)$, and $Q^{(k)}$ represents the global quality score of every resource in the system, after k iterations.

In CuboidTrust the global contribution score of a peer in the system represents the global trust value of that peer.

EigenTrust

One of the most cited and compared trust models for P2P networks is EigenTrust [9]. It assigns each peer a unique global trust value in a P2P file-sharing network, based on the peer's history of uploads, achieving thus a decreasing in the number of downloads of inauthentic files.

The local trust value s_{ij} is defined as follows:

$$s_{ij} = sat(i,j) - unsat(i,j)$$

where $sat(i,j)$ is the number of satisfactory transactions peer i has had with peer j (equally, $unsat(i,j)$ is the number of unsatisfactory transactions).

A probability distribution $\mathbf{p}$ (with $p_i \in [0,1]$) is defined over pre-trusted peers. For instance, if some set of peers P are previously known to be trusted, then $p_i = 1/|P|$ if $i \in P$, and $p_i = 0$ otherwise. With a definition like this, a normalized local trust value $c_{ij} \in [0,1]$ can be defined as:

$$c_{ij} = \begin{cases} \frac{\max(s_{ij},0)}{\sum_j \max(s_{ij},0)} & \text{if } \sum_j \max(s_{ij},0) \neq 0 \\ p_j & \text{otherwise} \end{cases}$$

Therefore, if a peer does not trust anybody or does not know anybody, she will choose to trust the pre-trusted peers.

The global reputation of peer i is defined in EigenTrust in terms of the local trust values assigned by other peers to peer i, weighted by the global reputation of the assigning peers. So the aggregation of normalized local trust values is computed as:

$$t_{ik} = \sum_j c_{ij} c_{jk}$$

being t_{ik} the amount of trust that peer i places in peer k based on asking his friends. Let C be defined as the matrix $[c_{ij}]$ and $\mathbf{t_i}$ as the vector containing the values t_{ik}, then we have that $\mathbf{t_i} = C^T \mathbf{c_i}$.

Peer i may wish to ask her friends' friends in order to get a wider view. In such a situation we would have that $\mathbf{t_i}^{(2)} = (C^T)^2 \mathbf{c_i}$. If she continues in this way (i.e., $\mathbf{t_i}^{(n)} = (C^T)^n \mathbf{c_i}$), she will achieve a complete view of the network after n iterations.

The trust vector $\mathbf{t_i}$ will converge to the same vector for every peer i, if n is large enough. In other words, it will converge to the left principal eigenvector of C. Namely, $\mathbf{t}$ is a global trust vector in this model whose elements, t_j, quantify how much trust the system as a whole places in peer j.

Finally, in order to avoid malicious collectives in P2P networks, the global trust value is re-defined as:

$$\mathbf{t}^{(k+1)} = (1-a) C^T \mathbf{t}^{(k)} + a\mathbf{p}$$

where a is some constant less than 1 and $\mathbf{t}^{(0)} = \mathbf{p}$.

BNBTM

In BNBTM [20] multidimensional application specific trust values are used and each dimension is evaluated using a single Bayesian network.

Beta probability distribution functions are used in order to represent the distribution of trust values according to interaction history as follows:

$$\tau_i = \frac{\alpha_i}{\alpha_i + \beta_{\bar{i}}}, \quad (i \in \{G, L, C\}, \bar{i} \in \{\bar{G}, \bar{L}, \bar{C}\})$$

where $\alpha_i = r_i + 1$ and $\beta_{\bar{i}} = s_{\bar{i}} + 1$, r_i and $s_{\bar{i}}$ are the number of interactions with outcome i and $\bar{i}$, respectively, and G means shipping goods as described, L means shipping lower quality goods, C means not shipping any good, and $\bar{G}$, $\bar{L}$ and $\bar{C}$ the opposite.

Since $\beta_{\bar{G}} = \alpha_L + \alpha_C - 1$, then

$$\tau_i = \frac{\alpha_i}{\sum_{j\in\{G,L,C\}} \alpha_j - 1} = \frac{r_i + 1}{\sum_{j\in\{G,L,C\}} r_j + 2}$$

The trust value is then obtained by normalizing this beta function:

$$P_i = \frac{\tau_i}{\sum_{j\in\{G,L,C\}} \tau_j + 2}$$

This trust value P_i is considered to be reliable if its corresponding confidence γ_i is greater than a certain threshold θ_γ.

An entity can estimate another's reputation according to the received ratings about the latter. A Bayesian network is constructed to perform the estimation for each dimension of trust (G, L, C). This eases both to extend the model to involve more dimensions of trust and to combine Bayesian networks to form an opinion about the overall trustworthiness of an entity. Each entity can evaluate its peers according to its own criteria, and the dynamic characteristics of criteria and of peer behavior can be captured by updating Bayesian networks.

After obtaining reputation values for a seller agent, the buyer normalizes them to get (P'_G, P'_L, P'_C) which are then used to calculate the utility of dealing with the seller. Buyers can select a utility function $U_R(x)$ according to their attitude to risk, where R is the risk tolerance. Suppose the price of an item is q, its intrinsic value is v, and the intrinsic value of a lower quality item is v'. Then, the expected utility can be calculated as:

$$EU = P'_G \times U_R(v-q) + P'_L \times U_R(v'-q) + P'_C \times U_R(-q)$$

$EU > 0$ means that dealing with the seller is worthy, otherwise is too risky.

GroupRep

GroupRep [18] is a model where trust relationships are classified in three levels: trust relationships between groups, between groups and peers and only between peers.

Group i trust in group j, $Tr_{G_iG_j}$, is computed as follows:

$$Tr_{G_iG_j} = \begin{cases} \frac{u_{G_iG_j} - c_{G_iG_j}}{u_{G_iG_j} + c_{G_iG_j}} & \text{if } u_{G_iG_j} + c_{G_iG_j} \neq 0 \\ Tr^{reference}_{G_iG_j} & \text{if } u_{G_iG_j} + c_{G_iG_j} = 0 \text{ and } \exists\ Trust^{path}_{G_iG_j} \\ Tr_{G_iG_{strange}} & \text{otherwise} \end{cases}$$

Where $u_{G_iG_j} \geq 0$ and $c_{G_iG_j} \geq 0$ are the utility and the cost, respectively, that nodes of group j have given to nodes in group i.

Given a set of reference paths between G_i and G_j, the most trustworthy reference path is that one including the most trustworthy group. Therefore, $Tr^{reference}_{G_iG_j}$ is defined as the minimum trust value along the most trustworthy reference path.

The trust value for strange groups (groups that have not had transactions with G_i) is calculated with this expression:

$$Tr_{G_iG_{strange}} = \begin{cases} \frac{u_{G_iG_{strange}} - c_{G_iG_{strange}}}{u_{G_iG_{strange}} + c_{G_iG_{strange}}} & \text{if } u_{G_iG_{strange}} + c_{G_iG_{strange}} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Trust value of group G_i about peer j, $Tr_j^{G_i}$, is defined as follows:

$$Tr_j^{G_i} = \begin{cases} \frac{u_j^{G(j)} - c_j^{G(j)}}{u_j^{G(j)} + c_j^{G(j)}} & \text{if } u_j^{G(j)} + c_j^{G(j)} \neq 0 \wedge j \in G_i = G(j) \\ Tr_{strange}^{G(j)} & \text{if } u_j^{G(j)} + c_j^{G(j)} = 0 \wedge j \in G_i = G(j) \\ \min\{Tr_{G_iG(j)}, Tr_j^{G(j)}\} & \text{if } j \notin G_i \end{cases}$$

$$Tr_{strange}^{G(j)} = \begin{cases} \frac{u_{strange}^{G(j)} - c_{strange}^{G(j)}}{u_{strange}^{G(j)} + c_{strange}^{G(j)}} & \text{if } u_{strange}^{G(j)} + c_{strange}^{G(j)} \neq 0 \\ 0 & \text{if } u_{strange}^{G(j)} + c_{strange}^{G(j)} = 0 \end{cases}$$

Where $G(j)$ is the group peer j belongs to, and $u_j^{G(j)} \geq 0$ and $c_j^{G(j)} \geq 0$ are the utility and cost, respectively, that peer j gives to other peers in group $G(j)$.

Finally, the trust value between node i and j is expressed as:

$$Tr_{ij} = \begin{cases} \frac{u_{ij} - c_{ij}}{u_{ij} + c_{ij}} & \text{if } u_{ij} + c_{ij} \neq 0 \\ Tr_j^{G(i)} & \text{if } u_{ij} + c_{ij} = 0 \end{cases}$$

AntRep

AntRep [19] is a novel model where reputation evidences are distributed over a P2P network, based on the swarm intelligence paradigm [10]. Specifically, authors propose the use of an ant system [2, 3] for building trust relationships in P2P networks efficiently.

In AntRep each peer has a Reputation Table (RT) which is very similar with the distance-vector routing table [15], but differs from: (i) each peer in the RT corresponds to one reputation content; (ii) the metric is the probability of choosing each neighbor as the next hop instead of the hop count to destinations.

There are two kinds of forward ants sent out for a particular reputation:

1. Unicast ants are sent out to the neighbor with the highest probability in the reputation table.
2. Broadcast ants are sent out when there is no preference to neighbors. This happens either when no path to the reputation has been explored or the information the node has is outdated.

Once forward ants find the required evidence (reputation information), a backward ant is generated. When the backward ant visits each node i, it updates the reputation table at the same time. This updating is carried out due to the next reinforcement rule:

$$P_i(t) = \frac{[\tau_i(t)]^\alpha [\eta_i(t)]^\beta}{\sum_{j\in N}[\tau_j(t)]^\alpha [\eta_j(t)]^\beta}$$

where η_i is the goodness value of the link between current node and its neighbor node i. τ_i is the pheromone deposit, which is defined as follows: if at time $t+\Delta t$, current node receives a backward ant from node i, then

$$\tau_i(t+\Delta t) = f(\tau_i(t),\Delta t) + \Delta p$$
$$\tau_j(t+\Delta t) = f(\tau_j(t),\Delta t),\ j \in N,\ j \neq i$$

where $\Delta p = \frac{k}{f(c)}$, being $k > 0$ a constant, $f(c)$ a nondecreasing function of cost c and c could be any parameter revealing the information of evidence or the scenario of current network. $f(\tau_i(t),\Delta t)$ is the pheromone evaporation function:

$$f(\tau_i(t),\Delta t) = \frac{\tau_i(t)}{e^{\Delta t/k}}$$

Finally, α and β are constants varied in different network environments.

Another function of the pheromone is to decide when to send out broadcast forward ants. When node k receives a request at time t, it first searches if there is an entry for the desired evidence. If no such entry exists, it simply sends out broadcast ants. Otherwise, it finds the one with the highest probability.

Semantic Web

A model is presented in [24] where the trustworthiness between two agents is computed by searching all the paths that connect themselves; next, for each path the ratings associated with each edge are multiplied and finally all the paths are added (normalizing that aggregation).

Let N be the number of paths from agent P to agent Q. D_i denotes the number of steps between P and Q on the i_{th} path. The set of Q's friends or neighbors is called M. m_i denotes Q's immediate friend or neighbor on the i_{th} path. w_i denotes weight of the i_{th} path. The weight of each path is calculated as follows (giving a higher weight to shorter paths):

$$w_i = \frac{\frac{1}{D_i}}{\sum_{i=1}^{N}\frac{1}{D_i}}$$

If agent P and agent Q are friends, $P \rightarrowtail Q$, or neighbors, $P \leftrightarrow Q$, then P's trust in Q, $T_{P\to Q}$, can be got directly. Otherwise

$$T_{P\rightarrow Q} = \sum_{i=1}^{N} \frac{T_{m_i\rightarrow Q} \times \prod_{i\rightarrowtail j \cup i\leftrightarrow j} R_{i\rightarrow j} \times \frac{1}{D_i}}{\sum_{i=1}^{N} \frac{1}{D_i}} = \sum_{i=1}^{N} T_{m_i\rightarrow Q} \times \prod_{i\rightarrowtail j \cup i\leftrightarrow j} R_{i\rightarrow j} \times w_i$$

Where the reliable factor $R_{i\rightarrow j}$ denotes to which degree i believes in j's words or opinions.

Global Trust

In [23] a trust vector is created for each node i, $\overrightarrow{TT_i} = \alpha \cdot \overrightarrow{DT_i} + (1-\alpha) \cdot \overrightarrow{Rep_i}$, being $\overrightarrow{Rep_i}$ the addition of indirect trust $\overrightarrow{Rep_i} = \overrightarrow{IDT_i}^1 + \overrightarrow{IDT_i}^2 + \cdots + \overrightarrow{IDT_i}^m$ and where IDT_{ij}^m represents the path of m steps between i and j with the greatest indirect trust value, and $\overrightarrow{DT_i}$ is i's vector of direct trust values.

Therefore, IDT_{ij}^m is computed as follows:

$$IDT_{ij}^m = (R_{ik_1} \otimes R_{k_1k_2} \otimes \cdots \otimes R_{k_{m-1}k_m}) \cdot DT_{k_m j}$$

Since $A \otimes B = \min(A,B)$, in order to get the indirect trust IDT_{ij}^m what is done is to keep the minimum recommendation $R_{k_pk_q}$ along the path of m steps between i and j and multiply it by the direct trust value of the last node of the path about j, i.e. $DT_{k_m j}$.

The reputation of j from i's point of view is computed as:

$$Rep_{ij} = IDT_{i1j} \oplus IDT_{i2j} \oplus \cdots \oplus IDT_{irj}$$

And since $A \oplus B = \max(A,B)$, i obtains the reputation of j by keeping the maximum indirect trust value of its r acquaintances about j.

PeerTrust

PeerTrust [22] is a reputation-based trust supporting framework, which includes a coherent adaptive trust model for quantifying and comparing the trustworthiness of peers based on a transaction-based feedback system. It has two main features: on the one hand it introduces three basic trust parameters and two adaptive factors in computing trustworthiness of peers, namely, feedback a peer receives from other peers, the total number of transactions a peer performs, the credibility of the feedback sources, transaction context factor and the community context factor. On the other hand, it defines a general trust metric to combine these parameters.

Let $I(u,v)$ denote the total number of transactions performed by peer u with peer v, $I(u)$ denote the total number of transactions performed by peer u with all other peers, $p(u,i)$ denote the other participating peer in peer u's i_{th} transaction, $S(u,i)$ denote the normalized amount of satisfaction peer u receives from $p(u,i)$ in its i_{th}

transaction, $Cr(v)$ denote the credibility of the feedback submitted by v, $TF(u,i)$ denote the adaptive transaction context factor for peer u's i_{th} transaction, and $CF(u)$ denote the adaptive community context factor for peer u. The trust value of peer u denoted by $T(u)$, is defined in equation (3).

$$T(u) = \alpha \cdot \sum_{i=1}^{I(u)} S(u,i) \cdot Cr(p(u,i)) \cdot TF(u,i) + \beta \cdot CF(u) \tag{3}$$

where α and β denote the normalized weight factors for the collective evaluation and the community context factor, respectively.

PeerTrust proposes two different credibility measures. The first one is to use a function of the trust value of a peer as its credibility factor so feedback from trustworthy peers are considered more credible and, thus, weighted more than those from untrustworthy peers. This credibility measurement is defined in equation (4).

$$Cr(p(u,i)) = \frac{T(p(u,i))}{\sum_{j=1}^{I(u)} T(p(u,j))} \tag{4}$$

The second credibility measure is for a peer w to use a personalized similarity measure between itself and another peer v to weight the feedback by v on any other peers. Let $IS(v)$ denote the set of peers that have interacted with peer v, the common set of peers that have interacted with both peer v and w, denoted by $IJS(v,w)$, is $IS(v) \cap IS(w)$. This measure is defined in equation (5).

$$Cr(p(u,i)) = \frac{Sim(p(u,i),w)}{\sum_{j=1}^{I(u)} Sim(p(u,j),w)} \tag{5}$$

where

$$Sim(v,w) = 1 - \sqrt{\frac{\sum_{x \in IJS(v,w)} \left(\frac{\sum_{i=1}^{I(x,v)} S(x,i)}{I(x,v)} - \frac{\sum_{i=1}^{I(x,w)} S(x,i)}{I(x,w)} \right)^2}{|IJS(v,w)|}}$$

About the transaction context factor $TF(u,i)$, several transaction contexts such as the size, the category, or time stamp of the transaction, can be incorporated in the metric so that the feedback for larger, more important, and more recent transactions can be assigned a higher weight than those for other transactions.

The incentive problem of reputation systems such as those where users will not receive rating information without paying or providing ratings, can be addressed in PeerTrust by building incentives or rewards into the trust metric through community context factor for peers who provide feedback to others. For example, let $F(u)$ denote the total number of feedback peer u gives to others. Then, a community context factor measure could be defined as follows:

$$CF(u) = \frac{F(u)}{I(u)}$$

PATROL-F

In the case of *PATROL-F* [17] the model incorporates many important concepts in order to compute a peer reputation, such as: direct experiences and reputation values, the node credibility to give recommendations, the decay of information with time based on a decay factor, first impressions and a node system hierarchy. It uses three fuzzy subsystems.

First one is used to set the importance factor of an interaction and related decisions. To decide and choose which data is critical or indispensable, or which data is needed more quickly, is a concept close to humans that fuzzy logic can model.

Moreover, there is the region of uncertainty where an entity is not sure whether to trust or not (when the reputation of a host is greater than the absolute mistrust level ϕ, but less than the absolute trust level θ). It is in this region where the fuzzy techniques are effectively applied.

Finally, for the result of interaction (RI) value, fuzzy logic can be used to capture the subjective and humanistic concept of "good" or "better" and "bad" or "worse" interaction. RI becomes the result of several concepts effectively combined to produce a more representative value. The decay factor τ is calculated based on the difference of a host's values of RIs between successive interactions.

Trust Evolution

Authors of [21] present a trust evolution model for P2P networks where trust relationships among peers are automatically built, in which two critical dimensions, experience and context, are taken into account.

The model applies the real numbers within the interval $[0,1]$ in order to quantify two kinds of trust: direct trust and recommendation trust.

Direct trust (DT) between two peers is computed using the last n interactions between those entities. An experience vector is used, assigning a weight $W_i > 0$ to each interaction, where $\forall\, i,j \quad i < j \Rightarrow W_i > W_j$, that is, the effect of an experience on DT is regressive with the time.

The content of each cell of the experience vector, d_i is computed as follows. On the one hand, a quality tuple is used in order to represent the quality standards of the interactions, $< q_1, q_2, \ldots, q_n >$. On the other hand we have the results of the interaction got by the truster, $< r_1, r_2, \ldots, r_n >$. Thus, the following rules are applied in order to compare the results and the standards.

If q_i is quantitative and better with the bigger value, then

$$d_i = \begin{cases} \frac{r_i}{q_i} & \text{if } r_i \le q_i \\ 1 & \text{if } r_i > q_i \end{cases}$$

If q_i is quantitative and better with the smaller value, then

$$d_i = \begin{cases} \frac{r_i}{q_i} \text{ if } r_i \geq q_i \\ 1 \text{ if } r_i < q_i \end{cases}$$

If q_i is boolean, then $d_i = 1 - (r_i \oplus q_i)$, where $\oplus$ denotes the exclusive-OR relationship.

If q_i is an application-dependable type, the application should provide a method to map the comparison of r_i and q_i to the value in $[0,1]$.

Therefore, the direct trust of the truster A towards the trustee B can be calculated as follows:

$$DT_B^A = \frac{\sum_{i=1}^n W_i \cdot d_i}{\sum_{i=1}^n W_i}$$

On the other hand, for a particular recommendation from peer B toward C, peer A evaluates it as follows. It first calculates Q:

$$Q = \frac{M \cdot |DT_C^A - DT_C^B|}{\sqrt{M \cdot DT_C^B \cdot (1 - DT_C^B)}}$$

where M is the number of A's experiences about C. Then it calculates k_1 and k_2 according to $k_i = \int_{-\infty}^{1-\frac{\alpha_i}{2}} \frac{e^{-x^2/2}}{\sqrt{2\pi}}, \quad \alpha_0 > \alpha_1 > 0$. Next, it compares Q with k_i to get the result e, where $e = satisfied$ means that A's experience is consistent with the recommendation from B.

$$e = \begin{cases} satisfied \text{ or } + & \text{if } Q \leq k_0 \\ uncertain \text{ or } / & \text{if } k_0 < Q < k_1 \\ unsatisfied \text{ or } - & \text{if } Q \geq k_1 \end{cases}$$

Each peer holds a recommendation vector for a particular recommender to record the recommenders' behaviors. Each cell of that vector, C_i, has a weight w_i, equal to the weight used in the experience vector. Finally, the recommendation trust of B from A's point of view, RT_B^A is computed as follows:

$$RT_B^A = \begin{cases} RT_B^A + \frac{-\ln(RT_B^A)}{\ln(W) - \ln(RT_B^A)} \cdot \left(\frac{\sum_{C_i=+} w_i}{\sum_{i=1}^n w_i}\right) \cdot (1 - RT_B^A) & \text{if } e = satisfied \\ RT_B^A & \text{if } e = uncertain \\ RT_B^A - \frac{-\ln(1-RT_B^A)}{\ln(W) - \ln(1-RT_B^A)} \cdot \left(\frac{\sum_{C_i=-} w_i}{\sum_{i=1}^n w_i}\right) \cdot RT_B^A & \text{if } e = unsatisfied \end{cases}$$

where $W = \alpha_0 \cdot \left(\sum_{C_i \neq /} w_i\right) + \alpha_1 \cdot \left(\sum_{C_i = /} w_i\right)$.

Finally, the combination trust of peer A toward peer C is computed as:

$$CT_C^A = \frac{DT_C^A + \sum_{i=1}^n RT_{R_i}^A \cdot RT_{R_i}^A \cdot DT_C^{R_i}}{1 + \sum_{i=1}^n RT_{R_i}^A}$$

TDTM

In TDTM [25] the ant colony system is also applied. On the one hand it identifies the pheromone and the trust, and on the other hand it identifies the heuristic and the distance between two nodes. In this model the level of trust increases when an interaction takes place and it slows down when there are no interactions for a long time.

Trust-pheromone between node i and j at time $t+1$ is defined as follows:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \sigma\tau_{ij}(t)$$

Where ρ is the trust dilution factor and $\sigma\tau_{ij}(t)$ is the additional intensity at each inter-operation between entities, defined as:

$$\sigma\tau_{ij}(t)\begin{cases} \frac{1}{\frac{1}{1-\tau_{ij}(t)}+1} & \text{if } i \text{ and } j \text{ interact at time } t \\ 0 & \text{otherwise} \end{cases}$$

Given a certain threshold R, and being $p_{ij}(t)$ the trust-degree between i and j at time t, if $p_{ij}(t) > R$, then entities i and j have enough trust-degree, so they can validate their certificate each other, not otherwise. Thus, in this model every entity has a local trust vector $p_i(t) = (p_{i1}(t), p_{i2}(t), \ldots, p_{in}(t))$ (reflecting the trust degree of peer i with every other one in the network) and the existence of a PKI is assumed.

TACS

TACS [5, 6] (Trust Ant Colony System) is a Trust model for P2P networks based on the bio-inspired algorithm of Ant Colony System (ACS), where pheromone traces are identified with the amount of trust a peer has on its neighbors when supplying a specific service.

It has the particularity that it not only gets the most trustworthy node to interact with, but it also obtains the most trustworthy path leading to the most reputable peer. In summary, the steps that compose this model are the next ones:

1. Client C executes TACS in order to find the “optimum” server S offering the service s
2. TACS launches the ACS algorithm and ants modify the pheromone traces of the network
3. TACS finishes, having selected the “optimum” path to server S'
4. TACS informs the client C that the “optimum” server found is S'
5. Client C requests service s to the server S'
6. Server S' provides service s' to the client C
7. Client C evaluates his satisfaction with the received service s'

8. If client C is not satisfied with the received service s', she punishes the server S' evaporating the pheromone of the links leading from C to S'

Each peer has its own pheromone traces for every link in the whole network, so it needs to know at any time the current topology of that network. When a client launches a set of ants and these ants are travelling and building the most trustworthy path leading to the most reputable server they have to decide at each peer whether to stop or keep looking for that path.

They will stop if they find a node offering the service requested by the client and whose pheromone traces belonging to the current path leading to it are high enough (over a certain threshold). Otherwise ants will select one of current node's neighbors who has not been visited yet. Which of those not visited neighbors to move towards is decided using the probability

$$p_k(c,s) = \frac{\tau_{cs}}{\sum_{u \in J_k(c)} \tau_{cu}} \tag{6}$$

where τ_{cs} is the pheromone trace of the link connecting nodes c and s and $J_k(c)$ is the set of neighbors of node c not visited yet by ant k. Actually, if $q \leq q_0$ the neighbor with the maximum pheromone trace is selected, otherwise equation (6) is used, where q is a random number within the interval $[0,1]$, and q_0 is a constant within the same interval.

Every time an ant crosses a link it modifies its pheromone trace in the following way:

$$\tau_{cs} = \tau_{cs} + (1-\varphi)\cdot\varphi\cdot(1-\tau_{cs})\cdot\tau_{cs}$$

where $\varphi \in [0,1]$ is a constant used to control the pheromone local updating.

Once every ant returns to the client having found a path, the client has to decide which of those paths is better. In order to measure the quality of each path, the next expression is used:

$$Q(S_k) = \frac{\%A_k}{\sqrt{Length(S_k)}} \cdot \overline{\tau}^k$$

where S_k is the solution found by ant k, $\%A_k$ is the percentage of ants that have selected the same path as ant k, $Length(S_k)$ is the length of the path S_k, and $\overline{\tau}^k$ is the average pheromone of that solution.

Finally, an additional pheromone updating is carried out over the links belonging to the best path found by all ants in the following way:

$$\tau_{cs} = \tau_{cs} + \rho \cdot \tau_{cs}^2 \cdot Q(S_{Best})$$

where $\rho \in [0,1]$ is a constant used to control the pheromone global updating.

This process of launching ants, modifying pheromone traces and selecting the most trustworthy path is repeated a number of times (number of iterations) which depends on the size of the network.

Once the algorithm has finished, the client applies for the requested service to the node selected by TACS and assesses its satisfaction with the received service by measuring its similarity with the originally requested one. If the client is not fully satisfied, a punishment in terms of pheromone evaporation is carried out all along the path connecting the client and the server.

Therefore, if the satisfaction (which is a number within the interval $[0,1]$) is greater than or equal to 0.5, then the punishment is:

$$\tau_{cs} \leftarrow \tau_{cs} - \varphi(1 - Sat) \cdot 2 \cdot df_{cs}$$

Otherwise, if satisfaction is less than 0.5 the punishment we have is

$$\tau_{cs} \leftarrow \left(\frac{\tau_{cs}}{df_{cs}} - \varphi \right) \cdot Sat$$

where df_{cs} is a function that assigns values in $[0,1]$ to every link of the path and it is defined as follows:

$$df_{cs} = \sqrt{\frac{d_{cs}}{L \cdot (L - d_{cs} + 1)}}, \quad d_{cs} \in \{1, 2, \ldots, L\}$$

being L the actual length of the whole path and d_{cs} the distance of link e_{cs} from the client (number of hops). Therefore, the last link (the one reaching the server) takes a value of 1, and the rest take smaller values. The nearer to the client a link is, the closer to 0 is given.

Finally, all the links falling into the malicious server are also punished, but only if the satisfaction is under 0.5.

3 Discussion

3.1 Trust and Reputation Management – What for?

P2P networks have provided us with many solutions and advantages in human relationships, communications and e-businesses, among other fields. However, in most of the cases, a user belonging to a P2P network will have to interact with many other unknown or stranger participants, in a human-to-machine (instead of a human-to-human) manner. This intrinsic feature of P2P networks can lead to the possibility of being easily defrauded or disappointed.

In the same way we ask our friends or acquaintances their opinion about a good or a service (a film, a book, etc.) before buying that good or asking for that service, it would be equally efficient and beneficial to be able to perform the same survey in virtual communities such as P2P networks.

Having such a useful mechanism in order to assess the global trust or reputation of a peer in the system, by collecting the recommendations of other users in the network who have had some previous interactions with that peer, the probability of being cheated by a malicious entity would be significantly decreased.

In the following subsections we will describe the general steps followed by most of the trust and reputation models and present some threats directly related with these specific approaches. Finally, we will analyze the strong and weak points of every trust and reputation model presented in Section 2, and conclude with some real systems applying a trust and reputation scheme.

3.2 Trust and Reputation Models Steps

We have seen that the main target followed by every trust and reputation model is, in summary, to identify those peers who are most reliable supplying a certain service or more trustworthy carrying out a certain task.

How those peers are selected differs from one model to other but, for instance, in most of them we can observe more or less the same generic steps [12], as depicted in Fig. 1. First of all an entity checks its previous experiences with a given peer in order to form what is usually called direct trust.

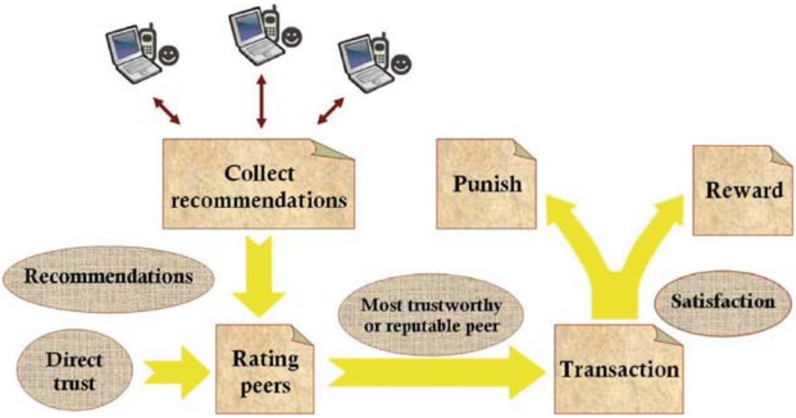


Fig. 1 Trust and reputation models steps

This direct trust can be assessed using complex expressions which usually take into account the number of previous transactions, the importance given to each transaction, the satisfaction obtained in each one, the time when it was performed, etc. Or it could even be computed as the difference between the number of satisfactory transactions and unsatisfactory ones, like in Eigentrust [9].

Additionally the indirect experiences (or experiences of other peers) are taken into account as well, obtaining what is commonly known as the reputation of a peer. At this point, some models (like [21, 23, 24]) even distinguish between the trust given to a peer as a service provider, and as a recommender, filtering out this way unfair ratings coming from malicious users, since their recommendations will be discarded.

How this reputation value is obtained is also very specific for each model, but the main idea is to collect information about the behavior of the target peer from other peers who have had previous interactions with it. This information or recommendations are influenced in some models by the reliability of the recom mender, as we mentioned. Otherwise, a collusion could be established where a set of malicious peers rated each other with the maximum value.

Therefore, an aggregation between the direct trust or direct experiences and the reputation or indirect experiences, weighted by the reliability of each recommender is performed in order to obtain a unique global trust value for a certain peer. Most of the models do not specify which peer is finally selected. It could be just the one with highest score, but not necessarily. And only TACS [5], from the studied models, provides not only the most reputable peer, but the most trustworthy path leading to it as well.

Once the peer to interact with has been selected, the transaction is effectively carried out. Then, the user who applied for a service or a task assesses her satisfaction with the received service or performed task. According to this satisfaction, a last step of punishing or rewarding the entity the transaction was done with, is performed.

However, not many models apply a specific and independent step of punish and reward, but they rather implicitly incorporate it in the rating step. Only TACS [5] and TDTM [25], from the analyzed ones, do it.

3.3 Common Challenges and Solutions in Trust and Reputation Management Over P2P Networks

As we have already seen, trust and reputation management in P2P networks provides several benefits to electronic interactions between users, like a minimum guarantee of benevolent behavior of another interacting peer.

Nevertheless, this kind of systems also have several common issues and challenges that need to be addressed when developing such mechanisms. Next we are going to discuss some of them.

3.3.1 Modeling Trust and Reputation

One of the first things to face when developing a new trust and reputation model, or when analyzing an existing one is the way of modeling (and consequently managing) precisely that: trust and reputation.

Thus, some models use bayesian networks (like BNBTM [20]), while others use fuzzy logic (like PATROL-F [17]), or even bio-inspired algorithms (as is the case of TACS [5], TDTM [25] and AntRep [19]). Other models, however, just give an analytic expression to compute trust (for example, GroupRep [18], and other models such as [23, 24] or [21]).

Each way of modeling trust and reputation has its own advantages and drawbacks. For instance, fuzzy logic allows us to model concepts like trust, reputation or recommendations in a manner closer to the way humans understand them. However, fuzzy logic will be difficult to scale to larger problems because there exist important limitations with conditional possibility, the fuzzy set theory equivalent of conditional probability [7].

Bio-inspired mechanisms have demonstrated a high adaptability and scalability in dynamic scenarios such as P2P networks. However, in some cases, their indeterminism and approximation technics can lead to choose a malicious peer as the most trustworthy one, discarding another clearly benevolent who could be selected.

Analytic expressions are most of the times easy to read and understand, but they may not take into consideration all the possible factors involved in the evaluation of trust and reputation for a certain participant in a P2P network since they need to manage those factors explicitly, while other approaches effectively deal with them in an implicit way.

Finally, Bayesian networks provide a flexible mechanism to represent multifaceted trust in many contexts of each others' capabilities (providing different services or carrying out several tasks, for instance). It also allows to efficiently combine different aspects of trust. One drawback is, however, that the approach can be computationally intensive, especially when the variables being studied are not conditionally independent of one another.

3.3.2 Contextualized Trust and Reputation

Another important concept managed in many trust and reputation models is what is commonly known as the context. Since a peer can be very trustworthy and benevolent when supplying a service or performing a task but, at the same time, very fraudulent or malicious when dealing with another service or task, it is not fair to identify it as fully trustworthy or untrustworthy.

That is why several models like CuboidTrust [1], BNBTM [20], PeerTrust [22], PATROL-F [17] or TACS [5] include a context factor or distinguish in one or another way the trust placed on a peer depending on the task or service it is requested to supply or perform.

3.3.3 P2P Networks Dynamism

Not many models take into account the intrinsic dynamic nature of P2P networks (i.e. nodes entering and leaving the community whenever they want) when modeling trust relationships.

Furthermore, only a few ones among the studied models (like CuboidTrust [1], EigenTrust [9], TACS [6] and PeerTrust [11]) present experiments dealing with this issue as well as with the fact that also the behavior of peers may be dynamic, i.e., peers are not always benevolent or malicious. And how fast and accurate a model can react against this behavioral changes is an important feature that every trust and reputation model should consider.

Therefore, in our opinion every trust and or reputation model for P2P networks should consider three basic scenarios:

1. A static one, where the topology of the network does not change along the time. This is the simplest scenario where trust and reputation models should work efficiently. As we said, many authors just consider a situation like this when developing their models. It is a good starting point but, however, it is not realistic.
2. A dynamic one, where the topology changes along the time, with nodes joining and leaving the network.
   This scenario could be used in order to test the reaction of a trust and reputation model against changes in the size and topology of the network, and the specific nodes composing it. For instance, it could be checked the reaction of the model if a very reputable (or, equally, a very fraudulent) node enters or leaves the system.
   This kind of experiment also allows to test how the model faces the topic of newcomers and deals with some threats like the Sybil attack [4].
3. An oscillating one, where the behavior of the nodes changes along the time, so they can be benevolent and become malicious and vice versa.
   Finally, this scenario would show if the model has a quick and accurate response or not against sudden behavioral changes of nodes trying to cheat. A good trust and reputation model should identify immediately these fluctuations and react consequently.

3.3.4 Collusion

There are also some security threats related to trust and reputation systems which are not completely considered in every model. For instance, only PeerTrust [22], CuboidTrust [1] and EigenTrust [9] explicitly treat the problem of collusion among malicious nodes.

A collusion consists of several malicious nodes (providing a bad service or performing tasks inadequately) joining in order to increase their reputation values by fake rating themselves and, on the other hand, decrease the reputation of current benevolent peers by giving negative recommendations about the latter, as it can be observed in Fig. 2.

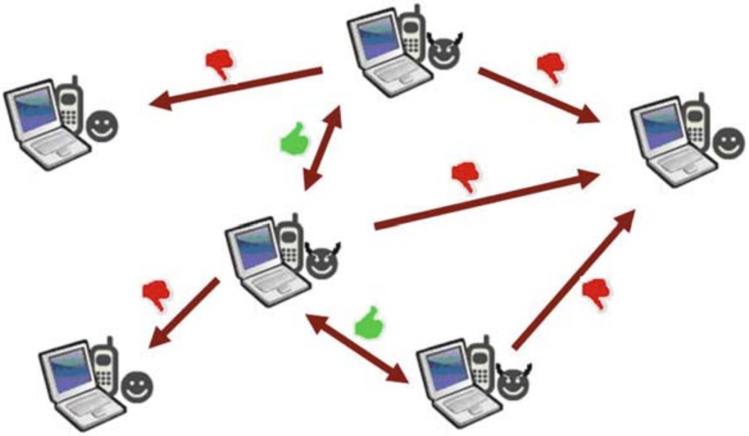


Fig. 2 Malicious peers forming a collusion

It is not easy to overcome this problem, specially when the percentage of nodes forming the collusion is quite high. Actually, any of the studied models is completely resilient against this kind of attack. What authors do is to try to minimize its global impact by punishing every node in the network which is known to belong to the collusion.

There are even variants of this attack, like a set of nodes providing good services but rating positively other malicious peers and negatively other benevolent ones. Those models which do not distinguish between the reliability of a user when providing a service or carrying out a task, and when supplying recommendations cannot effectively tackle this attack.

3.3.5 Identity Management – Sybil Attack

The last challenge we will discuss has to do with the identity management in virtual communities, specifically in P2P networks. It is a fact that cannot be obviated when designing and developing a new trust and reputation model since many deficiencies and weaknesses can emerge from an inaccurate management of this topic.

One of the most common problems related to identity management in trust and reputation schemes is what is known as Sybil attack. In a Sybil attack the reputation system of a P2P network is subverted by creating a large number of pseudonymous entities, using them to gain a disproportionately large influence. A reputation system's vulnerability to a Sybil attack depends on the cost of generating new identities, the degree to which the reputation system accepts inputs from entities that do not have a chain of trust linking them to a trusted entity, and whether the reputation system treats all entities identically.

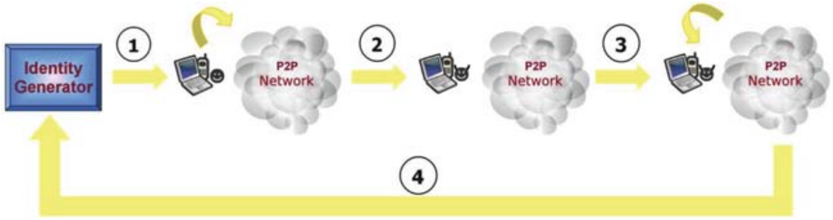


Fig. 3 Sybil attack

Figure 3 shows the steps followed in a Sybil attack and some other attacks related to identity management in trust and reputation models. These steps are:

1. An entity joins the community with a new identity, looking like being a trustworthy peer.
2. At a certain moment (probably after gaining some reputation in the system), this entity swaps her goodness and becomes malicious, obtaining thus a greater self-profit.
3. Once the system has detected her behavioral change and has identified her as a malicious participant, she leaves the network.
4. Finally, she generates a new identity and enters again the system, repeating the process indefinitely.

None of the studied models (except for EigenTrust [9]) considers explicitly the problem of the Sybil attack.

3.4 Strengths and Weaknesses of the Described Models

Once we have shown and described a number of trust and reputation models over P2P networks it is time to analyze in depth several important characteristics and features that define those models as well as some detected weaknesses and deficiencies.

Thus for instance, CuboidTrust [1] implements the testing scenarios described before, showing good outcomes and it also considers the problem of collusion. On the other hand, direct trust or direct experiences are not given a differentiated treatment, and the score takes discrete values in the set $\{-1,1\}$ instead of continuous ones in the interval $[-1,1]$.

EigenTrust [9] also implements the three mentioned scenarios as well as the collusion one. Moreover, it also takes into account the Sybil attack. It introduces the concept of pre-trusted peers, which is very useful in the model, but it is no applicable in all the cases, since there is not always a set of peers that can be trusted by default, prior to the establishment of the community.

One positive point of BNBTM [20] is its management of trust in different contexts, which can be combined to form an overall opinion of the trustworthiness of a peer. A weak point, in our opinion, is that it only deals with three discrete valuations for a transaction.

The distinction among trust between groups of peers, between groups and peers, and only between peers is the strong point of GroupRep [18]. However, it is missing a global trust value for a peer as a result of the combination of the three previous ones.

AntRep [19] has the ability to easily adapt to the dynamic topologies of P2P networks thanks to the use of ant colony systems. As a disadvantage, it just provides a mechanism to distribute reputation evidences, not to assess those evidences.

Searching all the paths connecting two agents may lead to some scalability problems in [23, 24]. On the other hand, both models clearly distinguish between direct trust and indirect trust or reputation, and also between the trust given to a peer as a service provider and as a recommender.

PeerTrust's [22] strengths are its good outcomes for the three previously proposed scenarios, as well as for a collusion one. It also introduces a context factor to distinguish the trust given to a peer for different transactions. However, the way it measures the credibility of a peer in equation (4) does not distinguish between the confidence placed on a peer when supplying a service or carrying out a task, and when giving recommendations about other peers.

This distinction is effectively done, however, in PATROL-F [17] and Trust Evolution [21]. Moreover, PATROL-F allows to model concepts like the result of an interaction in a similar way humans do thanks to the use of fuzzy logic.

The assumption of the existence of a PKI in a P2P network can be seen as a drawback of TDTM [25]. The use of an ant colony system allows it to adapt accurately to sudden changes in the topology of the network, as it is also the case of TACS [5].

TACS implements the three testing scenarios too, and it has a good performance in all of them as well. However, it needs every node to know the topology of the network at every moment, and that is not always feasible in a P2P community.

3.5 Real Scenarios

Finally, let us show some real trust and reputation systems [8] where most of the concepts explained here are employed. For instance, eBay auction market has a feedback scheme where every buyer and seller rates each other after a transaction between them is carried out. These feedbacks are centrally aggregated in order to get a reputation value for each role. Some studies [14, 16] reveal that buyers provide ratings about sellers 51.7% of the time, and sellers provide ratings about buyers 60.6% of the time. Of all ratings provided, less than 1% is negative, less than 0.5% is neutral and about 99% is positive. It was also found that there is a high correlation between buyer and seller ratings, suggesting that there is a degree of reciprocation of positive ratings and retaliation of negative ratings.

Another distributed system modeling reputation is PageRank [13], the algorithm which the search engine of Google is based on. It represents a way of ranking the search results based on a page's reputation, which is mainly obtained by the number of links pointing to it, since the higher is the number of incoming links, the better content that page is supposed to have. PageRank applies the principle of trust transitivity to the extreme since rank values can flow through looped or arbitrarily long hyperlink chains. Amazon, BizRate or Advogato are other examples of systems where a trust and or reputation scheme is applied in many different environments.

4 Conclusions and Future Work

P2P networks have been rapidly spread in the last few years. Nevertheless, together with its fast development, many security threats have also appeared, compromising sensitive information and promoting frauds in electronic transactions.

Trust and reputation management has arisen as one of the most innovative and accurate solutions to most of these threats. By using a trust and reputation system a peer who wants to interact with another peer in the community has more information and, therefore, more opportunities to select the right partner to have a transaction with, rather than with a fraudulent one.

In this work we have presented a description of some of the most representative trust and reputation models for P2P networks. We have explained how each of them works, how they manage the concepts of trust and reputation or how they gather information about other peers in the network.

But we have also discussed the main common characteristics of all the exposed models, as well as the most important differences among them. We have analyzed their weaknesses and deficiencies and shown some real examples of systems using trust and or reputation schemes.

However, there is still some work to do, for instance in the proposal of some standard patterns to lead a proper and accurate development of trust and reputation models for P2P networks, as well as validation tools allowing to homogeneously test every designed trust and reputation model.

Acknowledgements

This work has been supported by a Séneca Foundation grant within the Human Resources Researching Training Program 2007. Thanks also to the Funding Program for Research Groups of Excellence granted as well by the Séneca Foundation with code 04552/GERM/06.

References

1. Chen, R., Chao, X., Tang, L., Hu, J., Chen, Z.: CuboidTrust: A Global Reputation-Based Trust Model in Peer-to-Peer Networks. In: Autonomic and Trusted Computing, no. 4610 in LNCS, pp. 203–215. 4th International Conference, ATC 2007, Springer, Hong Kong, China (2007)
2. Cordón, O., Herrera, F., Stützle, T.: A review on the ant colony optimization metaheuristic: Basis, models and new trends. Mathware and Soft Computing **9**(2–3), 141–175 (2002)
3. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics **26**(1), 29–41 (1996)

4. Douceur, J.R., Donath, J.S.: The sybil attack. In: Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS 02), pp. 251–260 (2002)
5. Gómez, F., Martínez, G., Skarmeta, A.F.: TACS, a Trust model for P2P networks. Wireless Personal Communications, in press (2008)
6. Gómez Mármol, Félix: TACS – Trust Ant Colony System. URL `http://ants.dif.um.es/~felixgm/research/tacs`
7. Halpern, J.Y.: Reasoning about Uncertainty. The MIT Press (2003)
8. Josang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision Support Systems **43**(2), 618–644 (2007)
9. Kamvar, S., Schlosser, M., Garcia-Molina, H.: The EigenTrust Algorithm for Reputation Management in P2P Networks. In: Proc. of the International World Wide Web Conference (WWW). Budapest, Hungary (2003)
10. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
11. L3S Research Center: PEERTRUST – L3S Research Center. URL `http://www.l3s.de/peertrust`
12. Marti, S., Garcia-Molina, H.: Taxonomy of trust: Categorizing P2P reputation systems. Computer Networks **50**(4), 472–484 (2006)
13. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web (1998)
14. Resnick, P., Zeckhauser, R.: Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. In: The Economics of the Internet and E-Commerce, vol. 11, pp. 127–157. Elsevier Science (2002)
15. Stallings, W.: Data and Computer Communications (7th ed.). Prentice-Hall, Inc., Upper Saddle River, NJ, USA (2004)
16. Sundaresan, N.: Online trust and reputation systems. In: Proceedings of the 8th ACM conference on Electronic Commerce, pp. 366–367 (2007)
17. Tajeddine, A., Kayssi, A., Chehab, A., Artail, H.: PATROL-F – a comprehensive reputation-based trust model with fuzzy subsystems. In: Autonomic and Trusted Computing, no. 4158 in LNCS, pp. 205–217. Third International Conference, ATC 2006, Springer, Wuhan, China (2006)
18. Tian, H., Zou, S., Wang, W., Cheng, S.: A group based reputation system for P2P networks. In: Autonomic and Trusted Computing, no. 4158 in LNCS, pp. 342–351. Third International Conference, ATC 2006, Springer, Wuhan, China (2006)
19. Wang, W., Zeng, G., Yuan, L.: Ant-based reputation evidence distribution in P2P networks. In: GCC, pp. 129–132. Fifth International Conference on Grid and Cooperative Computing, IEEE Computer Society, Changsha, Hunan, China (2006)
20. Wang, Y., Cahill, V., Gray, E., Harris, C., Liao, L.: Bayesian network based trust management. In: Autonomic and Trusted Computing, no. 4158 in LNCS, pp. 246–257. Third International Conference, ATC 2006, Springer, Wuhan, China (2006)
21. Wang, Y., Tao, Y., Yu, P., Xu, F., Lu, J.: A Trust Evolution Model for P2P Networks. In: Autonomic and Trusted Computing, no. 4610 in LNCS, pp. 216–225. 4th International Conference, ATC 2007, Springer, Hong Kong, China (2007)
22. Xiong, L., Liu, L.: PeerTrust: Supporting Reputation-Based Trust in Peer-to-Peer Communities. IEEE Transactions on Knowledge and Data Engineering **16**(7), 843–857 (2004)
23. Yu, F., Zhang, H., Yan, F., Gao, S.: An improved global trust value computing method in P2P system. In: Autonomic and Trusted Computing, no. 4158 in LNCS, pp. 258–267. Third International Conference, ATC 2006, Springer, Wuhan, China (2006)
24. Zhang, Y., Chen, H., Wu, Z.: A social network-based trust model for the semantic web. In: Autonomic and Trusted Computing, no. 4158 in LNCS, pp. 183–192. Third International Conference, ATC 2006, Springer, Wuhan, China (2006)
25. Zhuo, T., Zhengding, L., Kai, L.: Time-based dynamic trust model using ant colony algorithm. Wuhan University Journal of Natural Sciences **11**(6), 1462–1466 (2006)

Anonymity in P2P Systems

Pilar Manzanares-Lopez, Juan Pedro Muñoz-Gea, Josemaria Malgosa-Sanahuja, and Juan Carlos Sanchez-Aarnoutse

Abstract In the last years, the use of peer-to-peer (P2P) applications to share and exchange knowledge among people around the world has experienced an exponential growth. Therefore, it is understandable that, like in any successful communication mechanism used by a lot of humans being, the anonymity can be a desirable characteristic in this scenario. Anonymity in P2P networks can be obtained by means of different methods, although the most significant ones are broadcast protocols, dining-cryptographer (DC) nets and multiple-hop paths. Each of these methods can be tunable in order to build a real anonymity P2P application. In addition, there is a mathematical tool called entropy that can be used in some scenarios to quantify anonymity in communication networks. In some cases, it can be calculated analytically but in others it is necessary to use simulation to obtain the network entropy.

Pilar Manzanares-Lopez
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain,
e-mail: pilar.manzanares@upct.es

Juan Pedro Muñoz-Gea
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain, e-mail: juanp.gea@upct.es

Josemaria Malgosa-Sanahuja
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain, e-mail: josem.malgosa@upct.es

Juan Carlos Sanchez-Aarnoutse
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain, e-mail: juanc.sanchez@upct.es

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_27, © Springer Science+Business Media, LLC 2010

1 Introduction

Anonymity can be a very desirable characteristic in peer-to-peer (P2P) networks. This property refers to the ability to do something without revealing one's identity. Anonymity can protect whistle-blowers within an organization. In this way, the organization cannot know who the whistle-blower is and as such can not persecute that person for their actions. Anonymous P2P systems can allow the development of applications in which anonymity is required. One example is electronic voting. In such an application anonymity is needed to prevent the vote being tied to the voter. In file sharing P2P applications, the anonymity can offer an comfortable environment to people who want to obtain (or to share) personal information about their experiences suffering a illness but without breaking their privacy.

Anonymity in P2P networks can be obtained by means of different mechanisms: source-rewriting mechanisms, broadcast protocols and Dining-Cryptographer (DC)-nets.

Source-rewriting systems provide anonymity via packet reshuffling. Examples of these systems in IP networks are Mixes, Onion Routing and Crowds. In these systems, messages are sent through the network using random paths different than the shortest path. Each node that forwards the message rewrites the source field of the packet with its own identifier. This mechanism ensures that when a particular node A receives a message from a particular node B, node A cannot be certain that B is the originator of the message. In Onion Routing, messages are routed through Onion Routers which are a dedicated set of servers. A message is encrypted into a layered data structure which contains cryptographic information for every hop on the path. The path through the network is established by the sender, who must have a complete knowledge of the routers. The last router in the path receives the message, decrypts the last layer of the message, and forwards it to the final destination. In Crowds, the paths are determined locally and on-the-fly by the forwarders instead of a priori by the senders, and encryption is done via pair-wise key exchange and symmetric encryption between forwarders. When a Crowds member request a message from another, it makes a random choice to either forward the request to another Crowds member or submit it to the server the request is intended for. In both of these systems, returned packets follow the reverse route from the sender.

Anonymous systems based on the broadcast mechanism, such as P^5, provide sender and receiver anonymity by transmitting encrypted packets at a constant rate to all participants. When a node has no data packets to send, it sends noise packets, which is then propagated throughout the network in the same manner as data packets. This approach provides strong anonymity, as a passive eavesdropper cannot tell which packets contain data and which packets are noise.

Finally, anonymous systems based on DC-nets, such as Herbivore, use the dining-cryptographer problem introduced in 1988 to obtain an anonymous communication. Herbivore organizes nodes into subgroups and requires shared secrets for nodes across subgroups via either a PKI or offline key exchanges.

Since there are several anonymity systems, it is necessary to define a tool, procedure or metric to measure or quantify the anonymity of each one, which makes possible to compare different anonymity implementations. A theoretical anonymity metric was introduced in 2002 [6] and revised by Diaz in 2006 [5]. This proposal defines the degree of anonymity as the normalized entropy of the analyzed anonymous system. Finally, the last section will present some case studies, where the entropy, and therefore, the degree of anonymity of some anonymous P2P systems is calculated.

2 Source-Rewriting Systems

Source-rewriting systems provide anonymity using indirection transmission: messages are routed from sender to receiver through intermediate nodes building a multiple-hop path. Examples of this type of protocols designed to offer anonymous IP communication are Mixes [2], Onion Routing [19] and Crowds [15].

Mixes [2] work by removing the timing correlation between arriving packets and outgoing packets. A mix is a forwarder node that queues incoming packets for a period of time $0 \leq t \leq T$ in such a way that, when finally sent, the packets appear as if they may have originated from any one of the nodes that contacted that mix in the last T seconds. In other source-rewriting protocols, including Onion Routing [19] and Crowds [15], messages are sent through the network via random paths that deter packet traces. Each node that forwards the message rewrites the source field of the packet with its own *id*. Consequently, attackers with limited wiretapping abilities cannot easily track packets back to their originators. In Onion Routing, the sender picks a full path through the network and encrypts the packets in layers. Each node along the path reorders the packets, strips off a layer of encryption and forwards them. In Crowds, the paths are determined locally and on-the-fly by the forwarders instead of a priori by the senders, and encryption is done via pair-wise key exchange and symmetric encryption between forwarders. In both of these systems, returned packets follow the reverse route from the sender. Source rewriting is also used in P2P content distribution networks such as Freenet [4] to obfuscate the identity of request originators.

2.1 Mixes

David Chaum introduced the concept of Mix-net [2] for anonymous communication. Anonymity is achieved by using mixes, which are proxies that help in hiding the sender from the receiver, and also provide sender and receiver with unlinkability against a global eavesdropper.

This system forwards encrypted messages through a chain of nodes, where each one can only know its predecessor and successor in the chain. The basic principle followed by this method is that every node in the chain waits until it receives a group of messages and then it forwards the mix-up of messages. That is, each node collects a group of encrypted messages from the senders, decrypts them, batches, reorders them, removes the senders' name and identifying information, and forwards them to next node (see Fig. 1). This makes it difficult for an eavesdropper to correlate the output messages with the corresponding input messages. The main drawback of this system is that each node in the chain has to wait until it gets sufficient number of messages for a proper mix-up. Some anonymous networks, such as Mixmaster and GNUnet, are based on this idea.

2.1.1 Mixmaster

Mixmaster [14] is an anonymous network based on the Mix-net concept. In this system, messages are encrypted using hybrid RSA and EDE 3DES encryption, while their sizes are kept constant by appending random noise at the end of the message. In version two, the integrity of the RSA encrypted header is protected by a hash, making tagging attacks on the header impossible. In version three the noise to be appended is generated using a secret shared between the network and the sender of the message.

Mixmaster allows large messages to be divided in smaller chunks and sent independently through the network. If all the parts end up at a common mix, then reconstruction happens transparently in the network, and so large emails can be sent to users without requiring a special software. It also allows messages to be sent multiple times, using different paths.

2.1.2 GNUnet

GNUnet is a searchable, decentralized, P2P network, whose main goal is to provide an anonymous file-sharing application. This network includes a technique for efficient content encoding (ECRS - the encoding for censor-ship resistant sharing) and a new protocol for anonymous routing, GAP (GNUnet's Anonymity Protocol) [1], which is based on Mix-net.

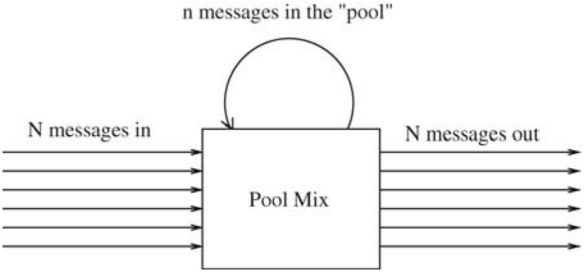


Fig. 1 General architecture of a Mix node

The content encoding technique divides a file into small blocks of 1kb which are hashed invididually. This blocks are called Data Blocks (*DBlocks*). The *DBlocks* are grouped into Indirection Blocks (*IBlocks*), and on top of these lies the Root Block (*RBlock*), which contains the description of the file and the query-hash to retrieve the file. Splitting the file into small chunks makes content distribution much easier. Further, the query for RBlock is sent encrypted so that the intermediaries cannot assess the information for which the user is looking for.

On the other hand, the GAP protocol has two types of messages: queries and replies. Furthermore, there exist two types of queries: search queries and download queries. A node requests for the RBlock of a particular file sending a search query. Once it gets the RBlock, it can download the entire file by sending several download queries.After receiving a download query a particular node checks its resources such as bandwidth, storage space or CPU to determine whether it can process the query or not. It drops the query if it is too busy. Then, it checks for the availability of the requested content. Finally, the different nodes in this network achieve anonymity by acting as intermediaries of messages, as in Mix-net.

2.2 Onion Routing

Onion routing [19] is a technique developed by David Goldschlag, Michael Reed and Paul Syverson for anonymous communication over a network. The main goal of this protocol is to protect the identity of an initiator and responder of a message, and also to protect the content of the message while it is traversing over the network.

The concept of routing onions is the core concept used by this protocol for anonymous communication. When an initiator wants to establish anonymous communication with a responder, he sends the message to an *application proxy* (see Fig. 2). Then, it forwards the message to an *onion proxy* which selects some routers and establishes a route, constructing an *onion*. The onion is a recursively layered data structure that contains the information about the route to be followed over a network. The routers in the onion are termed as *core onion routers*.

The next step in this process is to forward the onion to the *entry funnel*, which is an entrance for routing in the network. The entry funnel will decrypt the first layer to see the information about the next hop in the route. This process continues until the onion reaches the *exit funnel*. The exit funnel is responsible for traversing the packet to its destination.

Let us take an example for a better understanding of this protocol: User A wishes to send a message m to user B. Suppose the routing path is through onion routers S_1, S_2, ..., S_n (in that order), where S_i has public key P_i. It is assumed that all the parties involved use a common encryption algorithm E. Then, A sends the message $m' = E_{P_1}(E_{P_2}(...(E_{P_n}(m,B),S_n)...,S_3),S_2)$ to S_1. Each onion router on the path decrypts the message to find the identity of the next onion router and then forwards the encrypted payload to the next router. This process is repeated until the last router sends the message to its final destination, which is able to decrypt the original message m. If the destination responds to the message, the response message will be routed along the same path in reverse.

2.2.1 Tor

Tor [7] is an advanced version of Onion Routing. Instead of using a single onion, Tor implements an incremental path-construction in which the initiator extends the path hop by hop and negotiates session keys with each intermediate node on the path. As a benefit, the anonymous transmission is more reliable since the intermediate nodes on the path are online after the path construction. Therefore, this system is more convenient than onion routing in supporting TCP-based applications.

The algorithm to select the onion routers consists of two parts: the first part is used to select the entrance router, and the second part is used to select subsequent routers. The entrance router selection algorithm works by automatically selecting an onion router that is marked by the trusted directory servers as being "fast" and "stable". An onion router is marked as "fast" when its bandwidth is above the median of all bandwidth advertisements. On the other hand, a stable router is defined as one that advertises an uptime that is greater than the median uptime of all other routers.

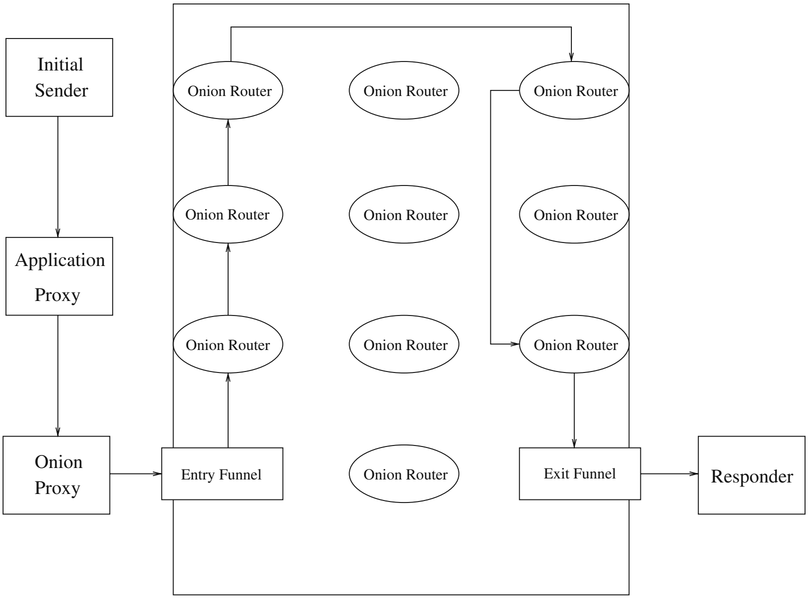


Fig. 2 Architecture of onion routing

The non-entrance router selection algorithm is intended to optimize onion router selection for bandwidth and uptime, while not always choosing the very best nodes every time. Tor has a set of TCP ports that are designated as “long-lived”. If the traffic transiting a path uses one of these long-lived ports, Tor will optimize the path for stability by pruning the list of available routers to only those that are marked as stable. The next part of the algorithm optimizes the path for bandwidth. Briefly, this algorithm works as follows: Let b_i be the bandwidth advertised by the i-th router, and let us assume that there are N routers. Then the probability that the i-th router is chosen is approximately $b_i/(\sum_{j=1}^{N} b_j)$. The most significant feature of the bandwidth optimization algorithm is that the more bandwidth a particular router advertises, the greater the probability that the router is chosen.

2.2.2 Tarzan

Tarzan [8] is a P2P anonymous network based on Onion Routing. All participant nodes run a software that: (1) discovers other participant nodes, (2) manages tunnels through chains of other participants to anonymize the packets, (3) forwards packets to implement other nodes’ tunnels, and (4) operates a NAT to forward other participants’ packets onto the ordinary Internet.

Tarzan routes packets through tunnels involving a randomly chosen sequence of Tarzan peers using mix-style layered encryption. Typical use of Tarzan proceeds in three stages. First, a node running an application that desired anonymity selects a set of nodes to form a path (also called tunnel) through the overlay network. Tarzan uses a simple gossip-based protocol similar to the Name-Dropper resource discovery protocol [9] for peer discovery. The tunnel initiator clears each IP packet’s source address, performs a nested encryption, and encapsulates the result in an UDP packet. Next, this source-routing node establishes a tunnel using these nodes. Finally, it routes data packets through this tunnel. That is, if the tunnel consists of a sequence of nodes $h_1, h_2, ..., h_l, h_{pnat}$ and the public key for each node is k_{hi}, the tunnel initiator produces the block $\{\{...\{\{p\}_{k_{hpnat}}\}_{k_{hl}}...\}_{k_{h2}}\}_{k_{h_1}}$ from the input packet p, and forwards it to h_1. h_1 will decrypt one layer of encryption and forward it on to the next hop. This process continues until the packet reaches the last node in the tunnel, which strips off the innermost layer of encryption. The exit point of the tunnel is a NAT, which forwards the anonymized packets to servers that are not aware of Tarzan.

In addition, Tarzan introduces the concept of traffic mimics to implement cover traffic mechanisms. Upon joining the network, a node a asks k other nodes to exchange mimic traffic with it. The node establishes a bidirectional, time-invariant packet stream witch each of the mimic nodes. Into this packet stream real data can be inserted. Tarzan uses a structured P2P lookup algorithm to discover the mimic nodes. Node a chooses its ith mimic as the peer returned by $lookup(K_i)$, where $K_i = hash_i(a.IPaddress, date)$ and $hash_i$ is a cryptographic hash function recursively applied i times. The uses of mimics slightly modifies the construction of a tunnel. Nodes only construct tunnels over links protected by cover traffic. To initiate a tunnel, node a chooses a mimic M_i^a as its first tunnel hop. This node M_i^a hast its own mimics, and returns this list to a. Node a selects the next hop from the resulting set and continues by tunneling an establish request to the second hop via the first hop. It repeats this process for l mimics. Finally, a selects a random node for h_{pnap} by $lookup(random)$.

2.2.3 I2P

I2P [10], also known as Invisible Internet Project, was started in 2003. I2P uses a technique called *garlic routing* for anonymous communication, in which the data is wrapped with several layers of encryption.

It is important to know the three critical concepts used by this network for a better understanding of it. First, I2P makes a distinction between the nodes participating in the network as "routers" and the endpoints, known as "destinations". Secondly, I2P makes use of "tunnels" to traverse messages in the network. A tunnel is a path selected from the set of routers. These tunnels can be categorized as either inbound tunnels (where everything given to it goes towards the creator of the tunnel) and outbound tunnels (where the tunnel creator shoves messages away from them). When a sender wants to send a message to a certain destination, he will select one of his outbound tunnels with instructions to forward the message to one of the inbound tunnels of the destination, thereby reaching the destination. Finally, the third concept is "network database" (or "netDb"), which maintains a pair of algorithms that are used to share network metadata. The two types of existing metadata are "routerInfo" and "leaseSets". The routerInfo gives the information that is required to contact a particular router and the leaseSet gives the necessary information required for routers to contact a destination.

As already stated, the message is wrapped with several layers of encryption so that the message looks entirely different in each hop of the tunnel. One more feature of I2P is that every participant in the network chooses the length of the tunnels and, in doing so, makes a tradeoff between anonymity, latency, and throughput according to its own needs.

2.3 Crowds

Crowds [15] is an anonymous protocol for web-transactions proposed by Reiter and Rubin. This protocol involves a group of users called *crowd*, each of whom wants to communicate with a corresponding web server but without revealing his identity. The idea is to randomly route each message through the crowd until one member of the crowd decides to pass it to the server.

A crowd can be thought of as a collection of user processes (also called *jondos*). When a user starts the jondo on the user's computer, it contacts a server called *blender* to request admittance to the crowd. If admitted, the *blender* reports to this *jondo* the current membership of the crowd and information that enables this *jondo* to participate in the crowd.

The user selects this jondo as her web proxy by specifying its host name and port number in her web browser as the proxy for all services. Thus, any request coming from the browser is sent directly to the jondo. Upon receiving the first user request from the browser, the jondo initiates the establishment of a random path of jondos that carries its users' transactions to and from their intended web servers. More precisely, the jondo picks a jondo from the crowd at random, and forwards the request to it. When this jondo receives the request, it flips a biased coin to determine whether or not to forward the request to another jondo; the coin indicates to forward with probability p_f . If the result is to forward, then the jondo selects a random jondo and forwards the request to it. Otherwise the jondo submits the request to the end server for which the request was destined (see Fig. 3).

2.3.1 AP3

AP3 [13] is an anonymous protocol built on top of Pastry [16] and based on Crowds. It relies on a network of peers to forward messages attempting to hide the originator identity.

When a node wishes to anonymously send a message, it first creates an anonymous request object comprised of the message itself and the address of the intended recipient. The request (that does not contain any information that can reveal the originator's identity) is forwarded to a node in the overlay selected by drawing a random key. Upon receiving a request, an AP3 node performs a weighted coin toss to decide whether to fulfill the request and send a message to the intended recipient, or to forward the message to another randomly selected peer. The decision to forward is made with probability p_f. This procedure provides a random path through the P2P network built from a variable number of random hops.

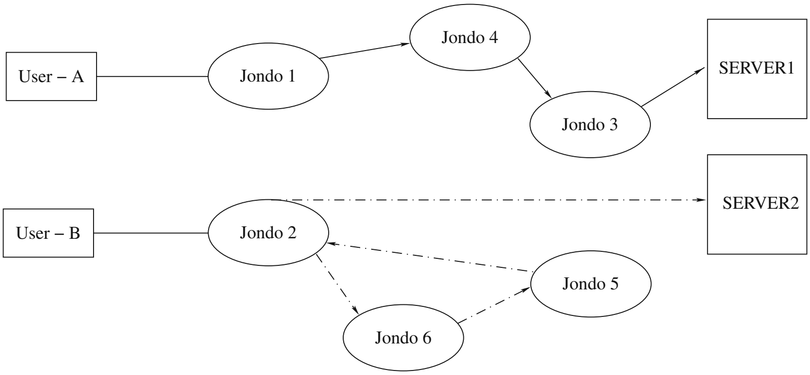


Fig. 3 Architecture of crowds. It is possible that random paths have loops

While the above mechanism allows nodes to send message without divulging their identity, it alone is insufficient to support a request-response communication in which the requester does not wish to divulge his identity. Since destination of a request doesn't know the identity of the sender, it is unable to reply. In order to offer this functionality, AP3 provides Anonymous Channels that allow a node to specify a return location for a message without divulging their identity.

When a node wishes to construct an anonymous channel, it first picks a channel identifier (*id*). It looks up the node closest to this *id* using Pastry, and then sends an anonymous message to that node using previously described anonymous sending mechanism. This message travels along a random path and each node along it remembers the previous node in a local table called the forwarding table.

Thus, if a node wishes to anonymously send a request and receive a response, it first creates an anonymous channel and then includes the address of the channel in the anonymously routed request. The response message is transmitted to the responsible node of the channel *id* included in the packet, and then it is forwarded to the sender through the anonymous channel. The nodes in the anonymous channel reconstruct the path back to the source node by using their forwarding tables. Using this mechanism, anonymity is preserved as no node along the channel knows if the previous node is the originator or just another intermediate node.

Additionally, when a path is established, the receiver specifies an expiration time that defines the period during which entries remain in the forwarding tables. Thus, forwarding table entries naturally expire over time. If a given channel has expired, the source node can simply create a new and different anonymous path to serve the anonymous channel.

2.4 Freenet

Freenet [4] is a P2P anonymous publishing system. In Freenet, a network of nodes is collectively responsible for storing and serving a set of documents. Each document is stored on a subset of nodes in the network and it is indexed by the hash of its contents, called the content-hash key. This key is used to locate the document in the network. Freenet also has a separate directory service to map names from a certain namespace to content-hash keys.

Each node maintains a routing table with references to several other nodes. The routing table includes a list of keys that each neighbor node is likely to serve. To find a document, a node will send a query to the neighbor with the closest matching key. If that node does not have the document, it will recursively forward the query to the node in its own routing table with the closest key, and so on. If a routing loop is detected, the process backtracks and the query is forwarded to the node with the next closest key. A time-to-live field in the query is used to limit the extent of the search.

A successful query causes the routing tables of the nodes along the path to be updated with a reference to the node that provided the data. Since the local storage on each node is limited, the routing table is limited in size; if it becomes full, the least recently used entry is removed. The document is sent back using the path of the query, and is usually cached along the return path. Caches are also updated by discarding less recently used items, resulting in popular documents being widely replicated in the network.

Because of the highly dynamic nature of a Freenet network, it is hard to say anything certain about the kind of topology it might have. The design intuition is that each node will specialize in storing and locating similarly valued keys. The larger structure of Freenet is expected to follow a power-law graph, with several high-degree nodes providing connectivity between nodes with smaller degree. Such a structure should support efficient routing even for large networks. Indeed, simulations show that Freenet networks can scale to many thousands of nodes and still maintain short median routes.

3 Broadcast Systems

In this category, researchers implement broadcast and multicast to achieve anonymity. An example might illustrate it well. Suppose two partners want to communicate anonymously with each other in an open environment. They can speak loudly in such a jargon that only they can understand the meaning of the sentences. Thus, even though other people within the sound scope can hear the talking, they cannot understand the content of the conversation. This kind of confidenciality can be used to provide anonymity in a broadcast scenario.

To hide the content of the message, users usually perform encryption to the messages using the receiver's public key. Thus, only desired receivers can recover the original data. In such a procedure, we find that the messages are propagated into replicate copies, and usually delivered by broadcasting or multicasting.

3.1 P^5

P^5 (*Peer-to-Peer Personal Privacy Protocol*)[17] can be used for scalable anonymous communication over Internet. This solution is based on a basic broadcast channel principle. All participants in the anonymous communication system are connected to a broadcast channel. All of them send fixed length packets onto this channel at a fixed rate to the entire group. Such a packet would be a noise packet or a signal packet, that is, a packet destined for a particular receiver. All the packets are encrypted using the receiver's public key, such that only the recipient of the message may decrypt it. The proposal assumes that there is a mechanism to hide the sender addresses, e.g., by implementing the broadcast using an application-layer P2P system.

This solution provides receiver anonymity, since the sender does not know where the receiver is or which host or address the receiver is using, and also provides sender anonymity, since all messages to a receiver comes from an upstream node, but the receiver cannot determine if that node is the original sender of the message or just a forwarder in the broadcast transmission. However, this basic solution does not scale due to its broadcast nature. As the number of people in the channel increases, the available bandwidth for any useful communication decreases linearly.

P^5 scales the basic broadcast solution by creating a broadcast hierarchy. Different levels of the hierarchy provide different levels of anonymity, at the cost of communication bandwidth. In a P^5 system with N users, each one have the public keys of every member $K_0, ..., K_{N-1}$. These N public keys will be used to create a logical broadcast hierarchy.

P^5 constructs a binary tree, called L, where each node of the tree consists of a bitstring and a bitmask. The bitmask indicates how many of the most significant bits on the bitstring are valid. The root of L consists of the null bistring and a zero length bitmask. The root is represented with the label $(\cdot/0)$. The left child of the root consists of the bistring starting with 0 and a 1-bit length bitmask (corresponding to label $(0/1)$) and the right child of the root consists of the bitstring starting with 1 and a 1-bit length bitmask (corresponding to label $(1/1)$). The rest of children of the L tree are defined in the same way as shown in Fig. 4.

A tree element (b/m) in L, also called group (b/m), corresponds to a broadcast channel in P^5. If a user sends a message to group (b/m), it will be forwarded to:

- Users in a local region. The message will be forwarded to all the members of the (b/m) group.
- Users in a path to root. For each $m' < m$, this message is also broadcast to all members of the group (b, m').
- Users in the subtree. For all $m'' > m$, the message is also sent to all groups (b/m'').

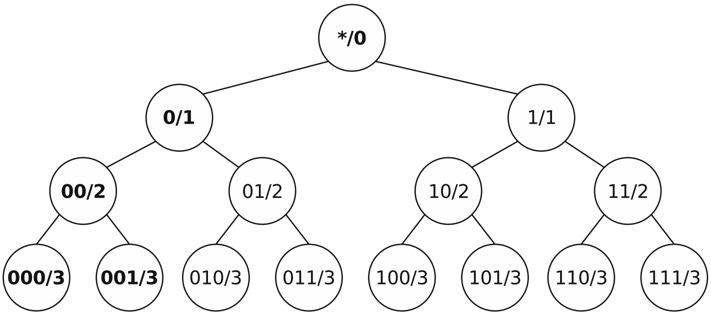


Fig. 4 Form of the P^5 logical broadcast tree (L). The effective broadcast channels of a user in group (00/2) is shown in boldface

The depth of the L tree is defined at run-time and depends on the number of members in the system (N). However, P^5 fix a maximum depth L_d, which is obtained considering that a system can accommodate approximately $2^{L_d} \cdot k$ users, where k is the least number of members in any channel.

Each user in the system joins a set of such broadcast groups. P^5 uses a public hash function $H(\cdot)$ to map users to some groups. Consider user A, with public-key K_A. Assume $b_A = H(K_A)\ mod\ 2^{L_d}$. User A will join some group of the form b_A/m_A, where m_A is chosen independently and randomly by user A. The choice of parameter m should be secret. Thus, giving a public key, it is public knowledge which set of groups a user may be in, but it is difficult to determine which specific group in this set the user has chosen.

Suppose users A and B join groups (b_A/m_A) and (b_B/m_B) respectively, and assume both know each other's public key. Since A knows K_B, it can determine b_B; however A does not know the value of m_B. Without any knowledge of the mask, A and B can communicate using the $(\cdot/0)$ channel. The communication efficiency can be improved as follows. Instead of sending messages through $(\cdot/0)$, A could try to send messages to some group (b_B, m), $m > 0$. B would receive these messages, and may replay back to A. It is supposed that the public key of the sender K_A is included in the message. However, in doing so, B sacrifices some anonymity since A can now map B to a small set of users. In general, B can choose to communicate back to A using any length mask; however, longer masks trade-off anonymity for communication efficiency.

A simple and logical anonymity analysis of this broadcast system is presented in the original paper. This is the main conclusion: Suppose a node A has joined the binary tree L at group (b/m), and let $E(b/m)$ be the number of members who receives a broadcast message sent to channel (b/m). Then, the anonymity of a node communicating using channel (b/m) is strongly related to the set of members who are part of $E(b/m)$.

In order to proof this property authors consider the sender and receiver anonymity cases. Sender anonymity is the size of the set of nodes that could have sent a particular packet to a given host. In P^5 a user connected to (b/m) sends packet at a constant rate (signal or noise), and these packets are received by all users in $E(b/m)$. If a malicious receiver tries to expose a sender, it can at best relate packets to its own broadcast group. Further, if the receiver is able to determine and compromise the router node, then the sender anonymity becomes the effective broadcast group of the sender. Respect to the receive anonymity, when user A sends a packet to user B at channel (b_B/m_B), every member of $E(b_B/m_B)$ receives the packet. From the perspective of an external observer, the behavior of the system is exactly the same whether B receives the packet or not. Thus B's receiver anonymity is exactly equivalent to the set of all users who receive the packet, namely $E(b_B/m_B)$.

3.2 Hordes

Hordes [11] provides sender anonymity by adopting the Crowds probabilistic forwarding mechanism, and achieves receiver anonymity by performing a multicast transmission. Figure 5 presents the main infrastructure of Hordes.

The purpose of using a group multicast is that the participants of the group cannot be easily determined. Since multicast addresses do not work with TCP connections, the transfers that involve a multicast address require the use of a UDP. The forward connection can use TCP, and within the data of the TCP payload it encapsulates the multicast address and a random number, used to identify the return message. Each intermediate jondo forwards this information to the responding jondo. That jondo will take the response payload and send it via the multicast address. Each jondo listens to one multicast address along with other jondos. However, not all jondos listen to the same address. One reason for this is so that a few jondos on the forward path will receive the actual reply from the server. It is very important that at least two jondos listen to an address, so that there is some unlinkability between the multicast address and the actual client.

The differences between Hordes and Crowds are the following. Crowds use TCP for all communication, where Hordes use a combination of UDP and TCP. The reverse path is much more interesting. Since packets travel directly back to the client without going through the jondos on the forward path, they are not subject to the same attacks of request/response matching that Crowds is subject to. In Crowds the forward and reverse data paths follow the same set of jondos, so a timing attack between a request and response may reveal information between the two. A problem with performing an encryption/decryption operation between each jondo is that it is possible for a local eavesdropper to determine the content of all messages originating from that jondo. However, this affects both Crowds and Hordes, since their forward routing schemes are very similar.

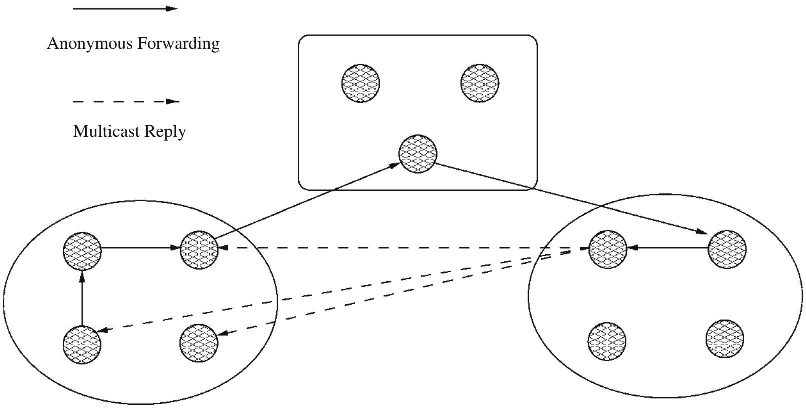


Fig. 5 Hordes infrastructure

4 DC-Net Systems

DC-nets [3] or Dining-Cryptographer networks were introduced by David Chaum in 1988. This anonymity solution is based on the broadcasting of XOR combination of messages to the system in predefined time slots. As in other broadcast-based anonymity systems, the main problem with DC-nets is that bandwidth decreases quadratically with respect to the number of nodes. That is because they are not used often in practice.

DC-nets work as follows. Suppose there are three participant, Alice, Bob and Charlie, one of whom (Alice) wants to communicate a one-bit message to Charlie. Each pair of users tosses a coin in secret; call these AB, AC and BC. Bob and Charlie report the XOR of their two coin tosses (Bob reports $B = AB \oplus BC$, Charlie reports $C = AC \oplus BC$). Alice, on the other hand reports the XOR of her coin tosses along with her message; that is, she reports $A = AB \oplus AC \oplus m$. A eavesdropper or a participant can then take the XOR of all packets sent in the network, i.e., $A \oplus B \oplus C = m$, since the double-reported coin tosses cancel out to reveal m. Neither the eavesdropper nor a participant can tell that Alice originated the message since it is composed with the participation of all parties.

Herbivore, the evolution of CliqueNet, is an anonymous broadcasting protocol that uses DC-nets.

4.1 Herbivore

The original DC-Net system is not practical because it scales with the number of users. Herbivore [18] was proposed as an attempt to incorporate DC-nets into an anonymous broadcast protocol that can be used in the real world. This anonymous communication system scales by partitioning the global network into smaller anonymizing cliques. Small cliques enable the protocol to operate efficiently within a small group and decouple the performance from the total number of participants in the network.

Herbivore relies on Pastry for its global organization. Each Herbivore node has a unique ID that determines its virtual position in relation to other nodes. Once a node arrives on the Pastry overlay, it finds the closest preexisting clique in the virtual identifier space and initiates a join protocol. The join protocol authenticates the client to the clique, authenticates the clique members to the client, and then picks pairwise PRNG[1] seeds to be used for transmission. A node that has exchanged PRNG keys with all preexisting clique members notifies a randomly selected clique participant of its intent to join the anonymous communication. This request is broadcast to all clique members on the anonymous channel, and the node is included in the next round of communication.

[1] Herbivore nodes eliminate the coin tosses and use a computationally secure PRNG (*PseudoRandomNumberGenerator*)-generated bit stream instead.

Within a clique, the DC-net protocol is used to communicate bits anonymously. As in traditional DC-nets, communication in Herbivore takes place in rounds, with a reservation and transmission phase. The reservation phase enables nodes to anonymously sign up for a slot in which to transmit their data. During the transmission phase, nodes transmit their data in the slots they reserved, or else simply broadcast the XOR of the keys they derived from their keys in order to assist other nodes' transmissions.

To enable inter-clique operation and also to contact traditional network services anonymously, Herbivore relies on proxies to bridge packets between cliques or between the anonymous and the public network. Each Herbivore node wishing to communicate with another clique or with such a service, randomly selects one of the nodes in its clique, other than itself, and anonymously asks it to serve as a proxy.

5 Analysis of Anonymity

The need for a metric to measure the performance of anonymity implementations appeared with the development of applications that enable anonymous electronic transactions, such as untraceable email, electronic voting, anonymous e-coins or privacy-enhanced web browsing.

As it is presented in [5], there has been different attempts to quantify anonymity in communication networks. Some authors define the degree of anonymity as a probability $1-p$, where p is the probability assigned by an attacker to potential senders. This metric considers users separately, and therefore does not capture anonymity properties very well. Berthold et al. define the degree of anonymity as a function of $log_2(N)$, where N is the number of users of the system. This metric only depends on the number of users of the system, and therefore does not express the anonymity properties of different systems.

In 2002, information theoretic anonymity metrics were independently proposed in two papers published at the same time [6]. Serjantov and Danezis uses entropy as measure of the effective anonymity set size. Diaz et al. goes one step further, normalizing the entropy to obtain a degree of anonymity in the scale $(0,1)$.

Entropy provides a measure of the uncertainty of a random variable. Let X be the discrete random variable with probability mass function $p_i = Pr(X = i)$, where i represents each possible value that X may take with probability $p_i > 0$. In anonymous communications, each i corresponds to a node of the anonymity set and the p_i denotes the probability of being the originator of the messages. We denote by $H(X)$ the entropy of a random variable, and by N the number of nodes in the anonymity set. $H(X)$ can be calculated as:

$$H(X) = -\sum_{i=1}^{N} p_i log_2(p_i) \tag{1}$$

The degree of anonymity is a normalized version of the entropy, which tells how good the system is performing on 0–1 scale. Therefore, the maximum entropy for N subjects is reached when all subjects are linked to the item of interest with equal probability ($p_i = 1/N$). In this case, all subjects are indistinguishable towards the adversary. The maximum entropy is $H_M = log_2(N)$.

If we assume that the adversary has no a priori information on the system, the amount of information gained by the adversary with an attack is the difference in the entropy before and after the attack: $H_M - H(X)$. The degree of anonymity is defined as the normalized value of this difference in knowledge of the adversary:

$$d = 1 - \frac{H_M - H(X)}{H_M} = \frac{H(X)}{H_M} \tag{2}$$

If a message goes only through honest nodes the degree of anonymity will be $d|_{\text{honest nodes}} = d_h = 1$. If we assume that the message goes through at least one corrupt node with probability p_c, and it crosses only through honest nodes with probability $p_h = 1 - p_c$, the mean degree of anonymity of the system is

$$\overline{d} = p_c \cdot d|_{\text{corrupt nodes}} + p_h \cdot d_h = p_c \cdot d_c + p_h \tag{3}$$

Next, it is presented the analysis of anonymity of different anonymous systems. The analysis of some of the them is extracted from [5].

5.1 Mix

It is considered a system with 10 potential senders, a mix network and a recipient. The attacker wants to find out which of the senders sent an email to the recipient. By means of timing attacks and traffic analysis, the attacker assigns a certain probability to each user as being the sender. First, it is considered an active internal attacker who is able to control eight of the senders (that means that these eight users have to be excluded from the anonymity set). Let p be the probability assigned to *user 1* and $1 - p$ the probability assigned to *user 2*. The distribution of probabilities is: $p_1 = p$; $p_2 = 1 - p$, and the maximum entropy for two honest users is: $H_M = log_2(2) = 1$. In Fig. 6 it is showed the variation of the degree of anonymity with respect to p. d reaches the maximum value ($d = 1$) when both users are equiprobable ($d = 1/2$). The minimum level ($d = 0$) is reached when the attacker can assign probability one to one of the users ($p = 0$ or $p = 1$).

Next, it is considered a passive global external attacker who is able to analyze the traffic in the whole system, but who does not control any of the entities (the anonymity set is, therefore, composed by 10 users). The maximum entropy for this system is: $H_M = log_2(10)$. The attacker comes to the following distribution:

$$p_i = \frac{p}{3},\ 1 \le i \le 3; \qquad p_i = \frac{1-p}{7},\ 4 \le i \le 10 \tag{4}$$

In this case there are two groups of users, one with three users and the other one with seven. Users belonging to the same group are seen by the attacker as having the same probability. The maximum degree $d = 1$ is achieved for the equiprobable distribution ($p = 0.3$). In this case d does not drop to zero because in the worst case, the attacker sees three users as possible senders with probability $p = 1/3$ (see Fig. 6), and therefore he cannot identify a single user as the sender of the message.

5.2 Onion Routing

Let N be the size of the *anonymity set*; the maximum entropy for N users is: $H_M = log_2(N)$. The attacker is able to obtain a subset of the *anonymity set* that contains the possible senders. The size of the subset is S $(1 \leq S \leq N)$. It is assumed that the attacker cannot assign different probabilities to the users that belong to this subset:

$$p_i = \frac{1}{S},\ 1 \leq i \leq S; \qquad p_i = 0,\ S+1 \leq i \leq N \tag{5}$$

Therefore, the entropy after the attack has taken place, and the degree of anonymity are:

$$H(X) = log_2(S); \qquad d = \frac{H(X)}{H_M} = \frac{log_2(S)}{log_2(N)} \tag{6}$$

Figure 8 shows the degree of anonymity with respect to S for $N = 5$, $N = 20$ and $N = 100$. Obviously, d increases with S, i.e., when the number of users that the attacker is able to exclude from the *anonymity set* decreases.

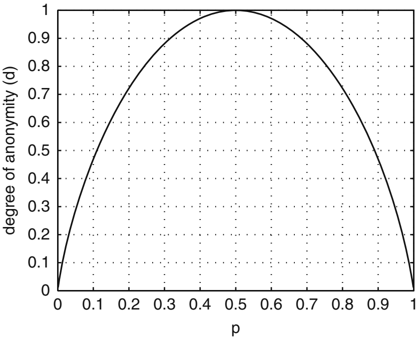


Fig. 6 Degree of anonymity for Mixes

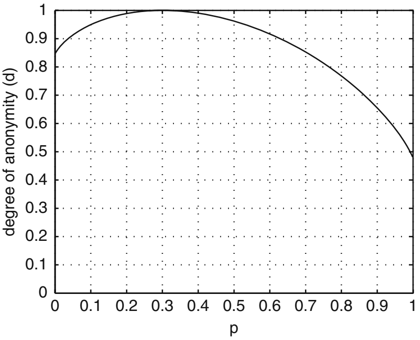


Fig.7 Degree of anonymity for Mixes

5.3 Crowds

First of all, it is necessary to establish some fundamental definitions. The event I denotes that the first collaborator on the path is immediately preceded by the initiator. H_k, $k \geq 1$ denotes the event that the first collaborator occupies the kth position on the path, where the initiator occupies the 0th position, and finally $H_{k+} = H_k \vee H_{k+1} \vee H_{k+2} \vee \ldots$. Therefore, $P(I|H_{1+})$ describes, given that a corrupt node is on the path, the probability assigned to the predecessor of the first collaborative node in the path.[2] Authors in Crowds paper [15] obtains an expression to determine $P(I|H_{1+})$

$$P(I|H_{1+}) = \frac{N - p_f(N-C-1)}{N} = 1 - p_f \frac{N-C-1}{N} \tag{7}$$

The probability assigned to the rest of collaborative nodes is zero. Assuming that the collaborative node does not have any extra information about honest nodes, the probabilities assigned to the rest of honest nodes are:

$$p_i = \frac{1 - P(I|H_{1+})}{N-C-1} = \frac{p_f}{N} \tag{8}$$

Using equations (7) and (8) in equation (1), the entropy of the system can be computed by:

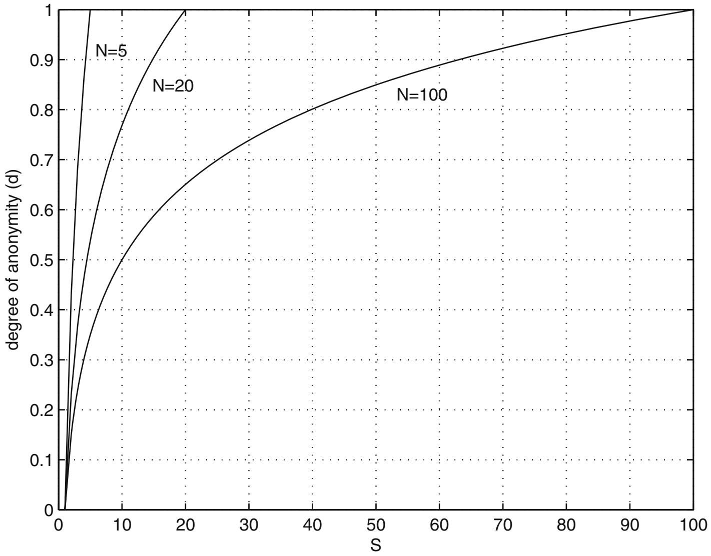


Fig. 8 Degree of anonymity for onion routing

[2] Note that, in Crowds, each node in the path is selected randomly. Therefore, $I \Rightarrow H_{1+}$ but the opposite is not true.

$$H(X) = -\frac{N - p_f(N-C-1)}{N} log_2 \left[\frac{N - p_f(N-C-1)}{N}\right] - \sum_{1}^{N-C-1} \frac{p_f}{N} log_2 \left(\frac{p_f}{N}\right) =$$

$$= \frac{N - p_f(N-C-1)}{N} log_2 \left(\frac{N}{N - p_f(N-C-1)}\right) + \frac{N-C-1}{N} p_f log_2 \left(\frac{N}{p_f}\right) \quad (9)$$

The degree of anonymity (d) is obtained normalizing the entropy, as it is indicated in equation (2). Taking into account that the size of the anonymity set is $N - C$ (because the C corrupt nodes are not considered part of the anonymity set), the maximum entropy is

$$H_M = log_2(N - C) \quad (10)$$

Therefore, knowing the probability that at least one corrupt node being in the path (p_c), $\overline{d}$ can be computed substituting equations (9) and (10) in equation (3).

The degree of anonymity (d) provided by this system is a function of C, N and p_f. In order to show the variation of d with respect to these three parameters authors in [5] chose $p_f = 0.5$ and $p_f = 0.75$, and $N = 5$, $N = 20$ and $N = 100$. The degree d is represented in Figs. 9–11 as a function of the number of collaborating *jondos* C. The minimum value of C is 1 (if $C = 0$ there is no attacker), and the maximum value of C is $N - 1$ (if $C = N$ there is no user to attack). For the case $C = N - 1$ it is obtained $d = 0$ because the collaborating *jondos* know that the real sender is the remaining non-collaborating *jondo*. We can deduce from the figures that d decreases with the number of collaborating *jondos* and increases with p_f. The variation of d is very similar for systems with different number of users.

5.4 Crowds with Limited Path Lengths

In the above section, it is calculated the entropy of Crowds. In [12], the authors compute the entropy of a system with fixed length multiple hop-paths. Based on these works, we introduce an analytical model to calculate entropy when the path length is L with probability $P(L = i)$, but limited to a maximum of M.

We define $q = (N - C)/N$ as the probability of selecting a honest node, and we know that

$$p_c = P(H_{1+})$$

$$P(H_1) = 1 - q$$

$$P(I|H_1) = 1$$

and

$$P(I|H_{2+}) = P(I|H_2) + P(I|H_{3+}) = P(I|H_{3+}) = 1/(N - C)$$

because a node never selects itself to be the next node ($P(I|H_2) = 0$) and if the first corrupt node on the path occupies the third or higher position, then it is immediately preceded on the path by any host peer (including the initiator) with equal likelihood.

In order to calculate the mean degree of anonymity of the system, it is only necessary to calculate the p_i associated with every node in the network. It is clear that $p_i = 0$ for corrupt nodes. The probability assigned to the predecessor node of the first corrupt node is $P(I|H_{1+})$. Finally, the rest of honest nodes $(N-C-1)$ share uniformly the rest of the probability of being the origin of the message. Therefore:

$$p_i = \begin{cases} 0 & \text{i} \in \text{C}, \\ P(I|H_{1+}) & \text{predecessor}, \\ \frac{1-P(I|H_{1+})}{N-C-1} & \text{rest} \end{cases} \tag{11}$$

Thus, it is enough to calculate $P(I|H_{1+})$. Since $I \rightarrow H_{1+}$ the value of this probability can be simplified as

$$P(I|H_{1+}) = \frac{P(I \wedge H_{1+})}{P(H_{1+})} = \frac{P(I)}{P(H_{1+})} \tag{12}$$

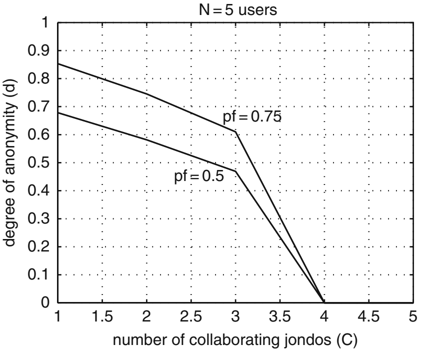


Fig. 9 Degree of anonymity for crowds with $N = 5$

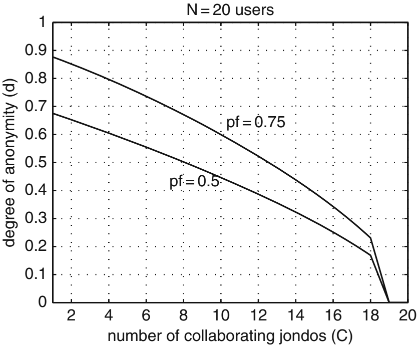


Fig.10 Degree of anonymity for crowds with $N = 20$

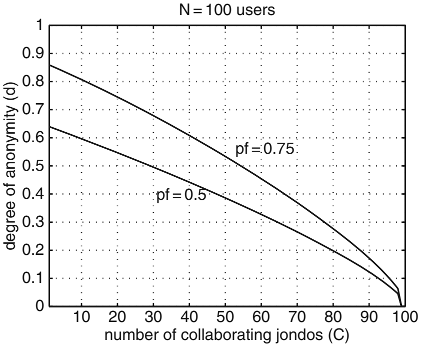


Fig. 11 Degree of anonymity for crowds with $N = 100$

Therefore, to calculate $P(I|H_{1+})$ it is necessary to previously know $P(I)$ and $P(H_{1+})$. First, $P(H_{1+})$ is calculated. It is assumed that a path of length j is composed by $j-1$ intermediate nodes and a destination node, and any node in the path can be corrupt. Therefore,

$$\begin{aligned} P(H_1) &= (1-q) \\ P(H_2) &= q(1-q) \\ &\cdots \\ P(H_M) &= q^{M-1}(1-q) \end{aligned}$$

The previous expressions can be simplified as

$$P(H_i) = q^{i-1}(1-q) \tag{13}$$

At this point, $P(H_{1+})$ can be resolved as

$$P(H_{1+}) = \sum_{j=1}^{M}\left(P(l=j)\cdot\sum_{i=1}^{j}P(H_i)\right) = \sum_{j=1}^{M}\left(P(l=j)\cdot\sum_{i=1}^{j}q^{i-1}(1-q)\right) \tag{14}$$

where $P(l=j)$ is obtained in each particular anonymous system.

On the other hand, in order to simplify the following operations, $P(H_{1+})$ can be expressed as

$$P(H_{1+}) = \frac{1-q}{T} \tag{15}$$

and inspection of equation (15), the value of T must be

$$T = \frac{1}{\sum_{j=1}^{M}\left(P(l=j)\cdot\sum_{i=1}^{j}q^{i-1}\right)} \tag{16}$$

Now, $P(I)$ can be derived as

$$P(I) = P(H_1)P(I|H_1) + P(H_{2+})P(I|H_{2+}) =$$

$$= (1-q) + [P(H_{1+}) - P(H_1)]\frac{1}{N-C} = \frac{(1-q)(1+T(N-C-1))}{T(N-C)} \tag{17}$$

Then,

$$P(I|H_{1+}) = \frac{P(I)}{P(H_{1+})} = \frac{1+T(N-C-1)}{N-C} \tag{18}$$

and therefore

$$p_i = \begin{cases} 0 & \text{i} \in \text{C}, \\ \frac{1+T(N-C-1)}{N-C} & \text{predecessor}, \\ \frac{1-T}{N-C} & \text{rest} \end{cases} \tag{19}$$

Now, the entropy $H(X)$ of the system can be computed as follows

$$H(X) = -\sum_{i=1}^{M} p_i log_2(p_i) = \frac{(N-C-1)(1-T)}{N-C} log_2\left(\frac{N-C}{1-T}\right) +$$

$$+\frac{1+(N-C-1)T}{N-C} log_2\left(\frac{N-C}{1+(N-C-1)T}\right) \tag{20}$$

The maximum entropy (H_{max}), considering that the size of the anonymity set is $N-C$ and that all these nodes have the same probability of being the origin $(1/(N-C))$ is equal to:

$$H_{max} = -\sum_{i=1}^{N-C} \frac{1}{N-C} log_2\left(\frac{1}{N-C}\right) = log_2(N-C) \tag{21}$$

Therefore, the degree of anonymity of the system will be

$$d_c = \frac{H(X)}{H_{max}} = \frac{(N-C-1)(1-T)}{(N-C)log_2(N-C)} log_2\left(\frac{N-C}{1-T}\right) +$$

$$+\frac{1+(N-C-1)T}{(N-C)log_2(N-C)} log_2\left(\frac{N-C}{1+(N-C-1)T}\right) \tag{22}$$

where $T = (1-q)/P(H_{1+})$ and $P(H_{1+})$ is obtained from equation (14).

Finally, the mean degree of anonymity of the system is

$$\overline{d} = p_c \cdot d_c + p_h = P(H_{1+}) \cdot d_c + (1 - P(H_{1+})) = 1 + P(H_1+)(d_c - 1) \tag{23}$$

6 Experimental Analysis of Anonymity

If it is not possible to model the entropy of a certain anonymous system analytically, it can be obtained by simulation or experimentation as follows. Each node, after selecting a random destination, builds a multi-hop path towards it. After n iterations, the anonymity metrics can be obtained using the following methodology.

Let us introduce two random variables: X, modeling the senders ($X = 1,\ldots,N-C$), and Y, modeling the predecessor observed by the first corrupt node in the path ($Y = 1,\ldots,N-C,\oslash$; where $\oslash$ represents that there is not an attacker in the path).

The probability distribution computing in the entropy expression (equation (1)) is really the distribution of X conditioned on a particular observation y. Therefore, the entropy metric evaluates $H(X|Y=y)$ for some y, an it can be calculated as

$$H(X|Y=y) = -\sum_{x} q_{x,y} log_2(q_{x,y}) \tag{24}$$

The distribution $q_{x,y}$ is an estimate of the true distribution p_i. The way to calculate it is as follows: We keep a counter $C_{x,y}$ for each pair of x and y. We pick a random value x for X and simulate a request. The request is forwarded according to the algorithm until it either is sent to the destination or is intercepted by a corrupt node. In the former case, we write down the observation $y = \oslash$, while in the latter case we set y to the identity of the predecessor node. In both cases, we increment $C_{x,y}$. Next, $q_{x,y}$ can be computed as follows

$$q_{x,y} = \frac{C_{x,y}}{K_y} \tag{25}$$

where $K_y = \sum_x C_{x,y}$.

On the other hand, the average entropy of the system, taking into account that a message can also go through honest nodes, can be expressed by

$$\overline{H} = \sum_{y\geq 1} Pr[Y=y] H(X|Y=y) + Pr[Y=\oslash] H(X|Y=\oslash) =$$

$$= \sum_{y} Pr[Y=y] H(X|Y=y) \tag{26}$$

With these definitions, expression (2) can be redefined as

$$\overline{d} = \frac{1}{H_{Max}} \sum_{y} Pr[Y=y] H(X|Y=y) \tag{27}$$

This methodology has been implemented in a Crowds simulator written in C language. First, in order to obtain an appropriate value for the number of iterations we simulate an scenario with $\overline{l} = 4$ for different number of iterations. Figure 12 presents the results with the 95% confidence intervals. We can see that from 10,000 iterations, the accuracy of the simulation results is within the 0.1% with respect to the stable value. Therefore, in our simulations the number of iterations is set to 10,000.

Figures 13 and 14 compares the simulation and the analytical results for Crowds when $N = 100$. Figure 13 represents $\overline{d}$ in function of p_f when $C = 10$, and Fig. 14

represents $\overline{d}$ in function of C when $p_f = 0.75$. In both cases, the simulation and the analytical results fit in perfectly.

7 Summary

In this work, the most important methods and applications to provide anonymity in P2P scenarios have been explained. Although all of them can be configured to have a good degree of anonymity without wasting communication efficiency, multiple hop path method seems to be the easiest way to obtain anonymity in a connectionless (datagram) IP network.

Entropy can be used as a mathematical tool to measure the overlay's degree of anonymity. A mathematical study in the case of multiple-hop path with limited and unlimited path length has been presented. In other (more complex) cases, entropy can be computed by simulation.

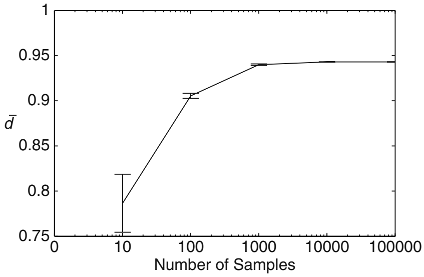
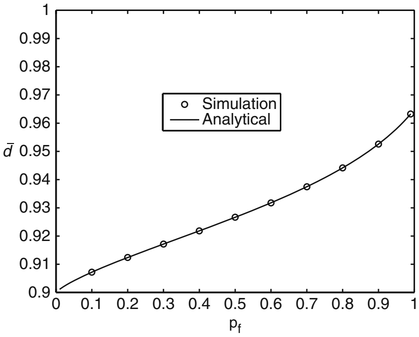


Fig. 12 $\overline{d}$ with confidence intervals as a function of the number of iterations



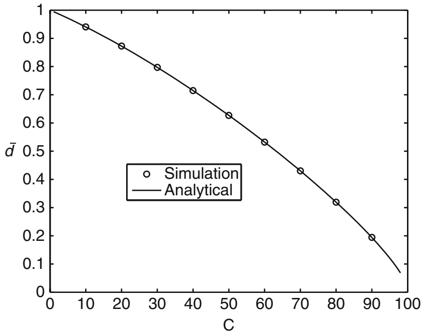


Fig. 13 Analytical and experimental evaluation of $\overline{d}$ for crowds as a function of p_f

Fig. 14 Analytical and experimental evaluation of $\overline{d}$ for crowds as a function of C

8 Future Research Directions

In the following we overview the most heatedly discussed problems to sketch the main challenges in anonymous P2P approaches

Tradeoff Between Performance and Anonymity. Current anonymity approaches incur extra overhead to both the system and the participants. The overhead is caused by encryptions and decryptions, anonymous transmissions, and mimic traffics. Most approaches in unstructured P2P systems suffer the inefficiency problem. On the other hand, a lot of researches have been proposed to improve the performance of the non-anonymous P2P systems by refining overlay topology, resource localization, and data delivery. However, most of these studies need the user identities in their optimization, which would compromise the anonymity. For anonymous P2P applications, without the help of identities, such as IP addresses, the performance may be severely degraded. Thus, any development of anonymous applications has a challenging effect on the performance of P2P systems. When providing anonymity, new efficient anonymous solutions will be urgently required in future P2P research.

Trust and Trust Management. Currently, the conflict is irreconcilable between anonymity and trust. Most prior approaches of trust and trust management are identity-based, which means real user identities are needed to make authentication and verification. However, this mechanism does not work when considering user's anonymity. Even though many anonymous schemes correlate a real ID with a pseudonym, the trust problem becomes more difficult in the proof of the correlation between these two entities. Therefore, trust management schemes need to be further explored in anonymous P2P environments. Combined with anonymity, trust management systems should deal with issues including authentication, verification, reputation or credit record storage, auditing, and misbehavior reporting in anonymous environments.

Incentive Versus Free-Riding. A free-rider always consumes resources from others while contributing a little or nothing to the system. Because of the open nature of P2P models, the free-riding phenomenon is popular and degrades the system performance. Anonymity may exacerbate this problem since the free-riders cannot be located, and since selfish behaviors might be prevalent without any punishment. Most existing incentive schemes are implemented in non-anonymous environments. In those approaches, the awards and punishments are related to users' real identities. Therefore, it is difficult to build an incentive mechanism in anonymous environments. Researchers are encouraged to design incentive schemes that can effectively correlate the awards and punishments to certain users without destroying their anonymity.

Acknowledgement

This research has been supported by project grant TEC2007-67966-C03-01/TCM (CON-PARTE-1) and it is also developed in the framework of "Programa de Ayudas a Grupos de Excelencia de la Region de Murcia, de la Fundacion Seneca, Agencia de Ciencia y Tecnologia de la RM (Plan Regional de Ciencia y Tecnologia 2007/2010)". Juan Pedro Muñoz-Gea also thanks the Spanish MEC for a FPU (AP2006-01567) pre-doctoral fellowship.

References

1. Bennett, K., Grothoff, C.: GAP – practical anonymous networking. In: Proceedings of Privacy Enhancing Technologies workshop (PET 2003), pp. 141–160 (2003)
2. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM **4**(2), 84–90 (1981)
3. Chaum, D.: The dining cryptographers problem: unconditional sender and recipient untraceability. Journal of Cryptology **1**(1), 65–75 (1988)
4. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A distributed anonymous information storage and retrieval system. In: Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, pp. 46–66 (2000)
5. Diaz, C.: Anonymity metrics revisited. In: Anonymous Communication and its Applications (2006)
6. Diaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Proceedings of Privacy Enhancing Technologies Workshop (PET 2002), pp. 54–68 (2002)
7. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium, pp. 21–21 (2004)
8. Freedman, M.J., Morris, R.: Tarzan: A peer-to-peer anonymizing network layer. In: Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), pp. 193–206 (2002)
9. Harchol-Balter, M., Leighton, T., Lewin, D.: Resource discovery in distributed networks. In: PODC '99: Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing, pp. 229–237 (1999)
10. I2p anonymous network. http://www.i2p2.de/ (2003)
11. Levine, B.N., Shields, C.: Hordes – a multicast based protocol for anonymity. Journal of Computer Security **10**(3), 213–240 (2002)
12. Lu, T., Fang, B., Sun, Y., Cheng, X.: Performance analysis of wongoo system. In: Proceedings of Fifth International Conference on Computer and Information Technology, 2005. CIT 2005., pp. 716–722 (2005)
13. Mislove, A., Oberoi, G., Post, A., Reis, C., Druschel, P., Wallach, D.S.: Ap3: cooperative, decentralized anonymous communication. In: EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop, p. 30 (2004)
14. Möller, U., Cottrell, L., Palfrader, P., Sassaman, L.: Mixmaster Protocol – Version 2. IETF Internet Draft (2003)
15. Reiter, M., Rubin, A.: Crowds: Anonymity for web transactions. ACM Transactions on Information and System Security **1**(1), 66–92 (1998)
16. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pp. 329–350 (2001)
17. Sherwood, R., Bhattacharjee, B., Srinivasan, A.: P5: A protocol for scalable anonymous communication. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 58–70 (2002)

18. Sirer, E.G., Goel, S., Robson, M., Engin, D.: Eluding carnivores: file sharing with strong anonymity. In: EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop, p. 19 (2004)
19. Syverson, P.F., Goldschlag, D.M., Reed, M.G.: Anonymous connections and onion routing. In: SP '97: Proceedings of the 1997 IEEE Symposium on Security and Privacy, pp. 44–54 (1997)

Private Peer-to-Peer Networks

Michael Rogers and Saleem Bhatti

Abstract This chapter offers a survey of the emerging field of private peer-to-peer networks, which can be defined as internet overlays in which the resources and infrastructure are provided by the users, and which new users may only join by personal invitation. The last few years have seen rapid developments in this field. We describe deployed systems, classify them architecturally, and identify some technical and social tradeoffs in the design of private peer-to-peer networks.

1 Introduction

Most peer-to-peer networks are designed to be open to the public: anyone can join a Kademlia overlay or share files in Gnutella simply by obtaining the addresses of other participants, which are often available from public servers [46, 61]. Even systems designed to protect the privacy of their users often have open membership policies [14, 29, 40, 58]. This openness enables wide participation, ensuring that a large amount of content is available in file sharing networks, for example, but it also allows attackers to monitor, join, and disrupt peer-to-peer networks [5, 11, 44, 71, 73].

In recent years, pervasive surveillance and censorship of the internet and highly publicised lawsuits against users of file sharing networks have led to increasing interest in private, authenticated communication between friends [19, 24, 25]. In a parallel development, creators of collaborative software have sought to combine the flexibility and autonomy of peer-to-peer networks with the confidentiality and authentication provided by traditional groupware.

Michael Rogers
University College London, London WC1E 6BT, UK, e-mail: m.rogers@cs.ucl.ac.uk

Saleem Bhatti
University of St Andrews, Fife KY16 9SS, UK, e-mail: saleem@cs.st-andrews.ac.uk

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_28, © Springer Science+Business Media, LLC 2010

This area has so far received relatively little attention from researchers; as with the first wave of peer-to-peer networks in 1999–2001, the developer community has often moved ahead of the academic community, opening up new areas of potential research. Consequently, many of the references in this chapter are to websites rather than to peer-reviewed papers.

The next section defines the scope of this chapter and describes some of the technical challenges faced by private peer-to-peer networks. Section 3 provides a survey of deployed systems, and in Section 4 we classify the systems architecturally and discuss design tradeoffs. Related research is discussed in Sections 5 and 6 concludes the chapter.

2 Background

2.1 Definitions

We define a *private peer-to-peer network* as an internet overlay in which the resources and infrastructure are provided by the users, and new users may only join the network by personal invitation. This definition excludes systems that rely on public servers, such as many online social networks and media sharing websites, but it does not necessarily imply decentralisation – some private peer-to-peer networks use central servers, but access to those servers is restricted to invited users, and the servers are owned and operated by users of the network.

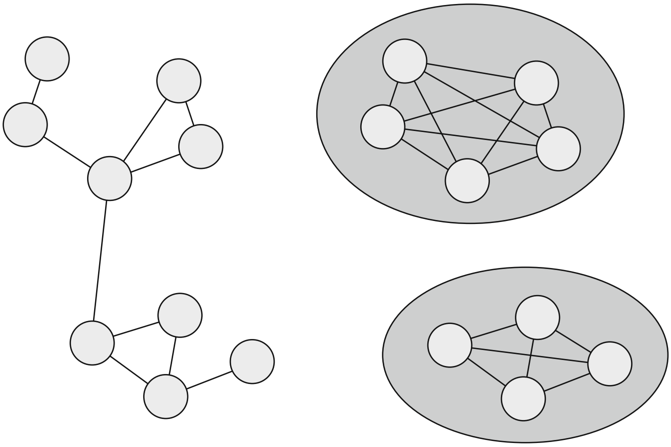


Fig. 1 Friend-to-friend and group-based networks. In a friend-to-friend network (*left*), users only connect directly to people they know. In a group-based network (*right*), any member of a group may connect directly to any other

Some private peer-to-peer networks allow direct connections between any pair of users, while others only allow direct connections between users who know one another. We will refer to the former as *group-based networks* and the latter as *friend-to-friend networks* [8, 34]. The distinction is illustrated in Fig. 1.

2.2 Technical Challenges

Firewalls and network address translators create significant problems for peer-to-peer networks, whether public or private. As the number of internet-connected devices increases, a growing proportion of users are behind "middleboxes" of one kind or another, and many also have dynamic network addresses that change daily or with every session.

Studies of peer-to-peer networks have shown rapid turnover or "churn" in the node population: many nodes are offline at any given time, there are significant daily and weekly cycles in the number of online nodes, and session durations are highly variable, with a few nodes staying online for long periods but most spending only a few hours online in each session [9, 37, 65, 70].

Middleboxes, dynamic network addresses and churn are especially problematic for private networks because of their restricted topologies. In a group-based network, at least one member of the group must run a stable node that is reachable from the public internet; the current addresses of other members can then be learned from that member. In a friend-to-friend network, on the other hand, every user needs at least one friend with a stable, reachable node.

Some systems cope with this problem by using internal or external lookup services for middlebox traversal and address discovery; others require users to exchange updated addresses out-of-band if they cannot reconnect to the network. By monitoring or participating in a lookup service, an attacker may be able to discover who connects to whom, so the decision to use such services involves a tradeoff between privacy and ease of use.

Data integrity, confidentiality and authentication are areas where private peer-to-peer networks may have an advantage over public networks. Because the users know one another, it is feasible for them to exchange cryptographic keys out-of-band; private peer-to-peer networks could even be bootstrapped using existing keys and trust relationships, such as those recorded in the PGP web of trust [10].

"Free riding" is a well-known problem in public peer-to-peer networks, where users often consume the resources provided by others without contributing anything in return [1]. Li and Dabek [43] argue that users of private networks will be more willing to contribute resources than users of public networks, since their contributions will benefit people they know. However, in some private peer-to-peer systems, such as those that support anonymous communication, resources can be consumed by people the provider does not know; free riding could be a problem for such systems.

3 Survey of Deployed Systems

In this section we review a number of private peer-to-peer systems that have been deployed in recent years.

3.1 Group-Based Networks

Groove [35, 72] is a groupware application for creating "shared spaces" that can span organisational boundaries. Each member of the group maintains a copy of the shared space's state, and encrypted updates are transmitted to other members when the state changes. It is not necessary to maintain connections between every pair of members, and indeed firewalls may make this impossible; members who are unable to communicate directly can exchange messages through dedicated relays. Changes to the shared space can be made while members are offline and synchronised when they reconnect.

Two kinds of shared space can be created. In a mutually trusting space, all changes to the state are authenticated using a single key. This makes it possible for members to spoof updates from other members. In a mutually suspicious space, an authentication key is generated for each pair of members, preventing spoofing but increasing the size of update messages.

Members can only join groups by invitation. The inviter is responsible for communicating the new member's encryption and authentication keys to the group, so messages from new members can be spoofed by their inviters, even in mutually suspicious spaces. Any member can evict any other member from a group, which is done by creating a new group key and transmitting it to all members except the evicted member.

N2N [20] is a group-based virtual private network (VPN) that uses relay nodes for address discovery and middlebox traversal.

Shinkuro [66] and PowerFolder [56] support group-based file sharing on local area networks; wide area connections are also possible if at least one member of the group acts as a relay. Each group is associated with a shared directory, and changes are synchronised automatically.

Direct Connect [18, 50] requires one member of each group to run a server, which is used for address discovery, keyword searches and chat.

Octopod [51] avoids the need for central servers by using OpenDHT [60], a public distributed hash table, for address discovery. Each group is associated with a shared directory, and the group owner can grant other users read-only or read-write access by sending them the appropriate keys.

WASTE [23, 74] is a group-based network created by Justin Frankel, the author of Gnutella. Like Gnutella it supports flooded queries and reverse path forwarded replies; these are used to implement keyword searches, file sharing and chat. Nodes can relay one another's messages if all-to-all connectivity is not possible. Links are encrypted and optionally padded to a constant traffic level, but there is no end-to-end encryption or authentication, so users can eavesdrop on one another and spoof messages. There are no group keys, so it is hard to evict users from groups.

Phex [52, 53] is a Gnutella implementation that supports the creation of private networks.

3.2 Friend-to-Friend Networks

Turtle [48, 54] is a friend-to-friend file sharing network designed for censorship resistance. Searches are flooded through the network, search results are forwarded back along the reverse path, and virtual circuits can be established for anonymous file transfer. The virtual circuit architecture is also capable of supporting other applications, including real-time communication.

Turtle uses a novel key agreement protocol in which friends exchange personal questions, the answers to which are assumed to be known to both users but not to eavesdroppers. This avoids the need for out-of-band key exchange, but the strength of the resulting keys will depend on the extent of the eavesdropper's knowledge about the users.

Freenet is a distributed cache where files can be published and retrieved anonymously. Early versions of the software depended on nodes learning addresses from successful queries [14], but Freenet 0.7 uses a new routing algorithm that does not require connections between untrusted nodes [13].

The new algorithm implements a distributed hash table where efficient routing is achieved by changing the nodes' locations in the key space rather than by changing the topology: each node starts at a random location in the key space and uses a stochastic algorithm to swap locations with other nodes until the network converges on a suitable arrangement for efficient routing [63]. Files are stored in the distributed hash table, allowing publishers and readers to remain anonymous.

Freenet stores two kinds of data: content hash keys (CHKs), which are blocks of data identified by their cryptographic hashes, and signed subspace keys (SSKs), which are blocks of data signed with a private key and identified by the hash of the corresponding public key. SSKs can be updated by anyone who knows the private key and retrieved by anyone who knows the public key, which makes it possible to implement a wide range of services over Freenet, including web browsing, message boards and email. An SSK keypair can be derived from a keyword, in which case anyone who knows the keyword can retrieve and update the SSK.

Like Freenet, GNUnet [6] can be configured to connect only to trusted nodes. GNUnet provides content-based and updatable keys and supports keyword searches [36]. Queries are routed using randomised flooding, which might seem to reveal less information about users than Freenet's social network-based routing. However, Kugler [41] describes a statistical attack that makes it possible, given a long series of related requests, to determine whether the requests are likely to have originated at a neighbouring node or to have been forwarded on behalf of another node. Similar attacks might be possible against Freenet even though its routing algorithm is deterministic, because related requests are routed independently [75].

SockeToome [68] enables friend-to-friend file transfers between users with dynamic network addresses, but it is arguably not a peer-to-peer network since no overlay is constructed. Gazzera [32] and Hybrid Share [39] use manually configured connections for friend-to-friend file sharing.

In anoNet [4], virtual private network (VPN) tunnels between friends are connected to form an encrypted overlay. The overlay uses standard internet protocols such as BGP, and even has an internal DNS hierarchy. A reserved network prefix is used to avoid accidentally routing packets between the overlay and the public internet.

Easter [22] uses email as a substrate for friend-to-friend file sharing. This makes it possible to circumvent many firewalls, but the protocol requires frequent polling of email accounts, which might attract the attention of system administrators.

CSpace [16] is a general-purpose friend-to-friend connection service based on a distributed hash table. The connections established by CSpace can be used for any application: file sharing, screen sharing and chat have been implemented so far. Participants in the distributed hash table can observe who connects to whom, which may have implications for privacy.

The Retroshare [59] instant messaging and file sharing network also uses a DHT for address discovery. Users can communicate indirectly through mutual friends, which may allow them to build up trust in one another before requesting direct connections. Galet [31] and Cryptic6 [15] allow friends-of-friends to communicate in a similar way, but they do not use distributed hash tables or relay servers for address discovery, so addresses and encryption keys must be exchanged out-of-band or through mutually trusted friends. Alliance [3] optionally creates direct connections between friends-of-friends, which may help with address discovery, churn and middlebox traversal. Users lose some privacy by revealing their network addresses to friends-of-friends, but this may be preferable to using a third-party lookup service operated by strangers.

Tsnecv [28] is a file sharing system for local area networks in which users may assign different access levels to friends, friends-of-friends and strangers.

3.3 Other Networks

Aimster [2] was the first peer-to-peer system to enable private file sharing between friends. Because of its reliance on centralised public servers, it was quickly shut down [24].

GigaTribe [33] uses a public server to coordinate group-based file sharing and chat, with an optional commercial relay service for users who are unable to connect directly.

Sneakernet [67] uses small data-carrying devices such as mobile phones and memory sticks to pass information between friends. A gossip-based protocol allows encrypted messages to travel over multiple hops between a trusted public server and anonymous users.

Many other systems use websites or other public servers to coordinate peer-to-peer communication between friends or in private groups, and recently some centralised instant messaging networks have also begun to support peer-to-peer voice and video connections. We consider such systems to be outside the scope of this chapter, however, because of their reliance on public servers.

4 Architecture

The private peer-to-peer networks described in the previous section can be classified architecturally along four axes: scale, visibility, centralisation, and application support. Table 1 summarizes existing research works based on these axes.

1. Scale – does the system consist of isolated local networks or a single global network?

The issue of scale raises difficult technical and social tradeoffs. Large private networks are likely to face many of the same connectivity challenges as public peer-to-peer networks, including heterogeneity and churn. Because of their size, they are also more likely to attract the attention of eavesdroppers and attackers. Users may feel less of an obligation to contribute to strangers than friends, so free riding may also be an issue for large networks. On the other hand, the wider range of people and resources can make large networks more attractive to potential users [45].

Deployed systems deal with these tradeoffs in a variety of ways. At one end of the axis is WASTE, which is designed for small groups of mutually trusting users; users can belong to more than one network, but traffic does not pass between networks. The group size is limited in practice by the protocol's use of flooding and the difficulty of evicting misbehaving users, which encourages users to be cautious about giving invitations.

At the other end of the axis is Freenet 0.7, which is designed to be a "globally scalable darknet" [12]. Freenet's routing algorithm is based on the assumption that all users belong to a single small-world social network, and it may not be possible to merge mature networks without seriously disrupting routing. It is difficult to grow a global network from a single seed, however: not all potential users know a user of the main network, so recent versions of the software optionally create connections between strangers in addition to manually configured friend-to-friend connections [64].

GNUnet and Turtle take an intermediate approach: messages can be forwarded anonymously across the friend-to-friend overlay, and local networks created by small groups of users can be merged by establishing friend-to-friend connections between them. However, GNUnet and Turtle both rely on flooding, which does not perform well in large networks; thus even in a merged network, communication may effectively be confined to local regions of the overlay. Indirect communication in Galet, Cryptic6 and Alliance is explicitly local: friends-of-friends can communicate pseudonymously, which may allow users to build up trust in new friends before connecting to them directly.

The ability to merge mature networks could be important for the growth of private peer-to-peer systems, because it may be easier for a potential user to find friends who are interested in setting up a local network than to contact and befriend a member of an existing network.

2. Visibility – can users connect to everyone in the network, or only to their friends? Who else can see that they are participating?

The issue of visibility separates group-based networks from friend-to-friend networks. This distinction becomes more important as networks grow, because any user may invite a friend who does not know all the other users. In a group-based network, the newly invited user will be able to connect to any existing user; thus group-based networks become less private as they grow, whereas friend-to-friend networks can (at least in theory) remain private at any scale. Indirect pseudonymous communication through mutual friends represents an intermediate position between group-based and strictly friend-to-friend visibility.

Group-based networks could be vulnerable to Sybil attacks [21], where an attacker uses multiple identities simultaneously, and whitewashing [30], where an attacker changes identities to escape the consequences of past misbehaviour. For example, it is easy to imagine an attacker automatically "inviting" new identities into a group more quickly than the other users can manually evict them. Such attacks can be prevented by requiring a certain fraction of existing members to approve each invitation [57], but this approach does not scale to large groups where not all users know one another.

Friend-to-friend connections are not a panacea for identity-related attacks, but they guarantee that every identity within a local scope belongs to a different individual, which prevents whitewashing; it might also be possible to use the structure of social networks to limit the impact of Sybil attacks (see Section 5).

Regardless of whether the network is group-based or friend-to-friend, users may need to make additional connections to discover one another's addresses. Some networks use public distributed hash tables for address discovery, while others use external servers for NAT and firewall traversal [27, 38, 62]. Unfortunately, involving third parties in communication can have implications for autonomy and privacy: if participants in a public DHT or the operators of a public server can identify the users of a private network, and perhaps even observe which users connect to which others, many of the benefits of using a private network will have been lost.

It may be possible to avoid relying on external lookup services if some members of the network have reliable nodes with stable network addresses. In a group-based network, only one member of the group needs a stable address, but in friend-to-friend network, every user needs at least one friend with a stable address.

Table 1 The architecture of deployed private peer-to-peer networks

Name	Scale	Visibility	Centralisation	Application support
Direct Connect	Local	Group	Central server	File sharing, chat
N2N	Local	Group	Dedicated relays	VPN
Groove	Local	Group	Dedicated relays	Shared workspace
PowerFolder	Local	Group	Members may run relays	Shared workspace
Shinkuro	Local	Group	Members may run relays	Shared workspace
WASTE	Local	Group	Members may run relays	File sharing, chat
Phex	Local	Group	Members may run relays	File sharing
Easter	Local	Friends	Email servers	File sharing
Gazzera	Local	Friends	Decentralised	File sharing
Hybrid Share	Local	Friends	Decentralised	File sharing
Tsnecv	Local	Configurable	Decentralised	File sharing
Alliance	Flexible	Friends-of-friends	Decentralised	File sharing, chat
Cryptic6	Flexible	Friends-of-friends	Decentralised	File sharing, chat
Galet	Flexible	Friends-of-friends	Decentralised	File sharing, chat
Turtle	Flexible	Friends	Decentralised	Virtual circuits
GNUnet	Flexible	Friends	Decentralised	Distributed cache
Freenet 0.7	Global	Friends	Decentralised	Distributed cache
anoNet	Global	Friends	Decentralised	VPN
CSpace	Global	Friends, DHT	Decentralised (DHT)	Virtual circuits
Retroshare	Global	Friends-of-friends, DHT	Decentralised (DHT)	File sharing, chat
Octopod	Global	Group, DHT	Decentralised (DHT)	File sharing

Connections between friends-of-friends may help to mitigate this problem by giving each node a larger set of neighbours.

If a user cannot contact any previously known nodes when reconnecting to the network, it may be necessary to exchange updated addresses with a friend out-of-band, so the use of external address discovery services involves a tradeoff between privacy and ease of use.

3. Centralisation – Does the Network Rely on a Central Server?

A number of public peer-to-peer networks have been shut down by attacking their central servers [24], leading to the perception that centralised designs are fragile and should be avoided. However, the risks may be different in private networks, where servers can be more or less hidden from untrusted parties. Centralisation can make it easier to manage identities, exchange cryptographic keys, and learn the current addresses of other users, provided all users trust the operator of the server.

Direct Connect requires a central server or "hub" for every group. Many other group-based networks can operate without servers if all users are on the same local area network, but require at least one member to act as a relay for wide area communication. Groove and N2N use dedicated relays that can see who is communicating but cannot decrypt the messages they forward. Easter relies on email servers, which are decentralised but may not be controlled by users of the network. Most other friend-to-friend systems rely on manual port forwarding or hole punching to traverse NATs and firewalls [27, 38, 62]. Friend-to-friend connections can be lost if both friends change their addresses at the same time, and it may be necessary to exchange updated addresses through an alternative channel such as email.

Octopod, Retroshare and CSpace use distributed hash tables for address discovery, which may allow untrusted parties to observe which users connect to which others. Freenet avoids this problem because its DHT implementation only uses existing friend-to-friend connections; users can publish their encrypted contact details under updatable keys for their friends to retrieve anonymously.

4. Application support – what functionality does the network provide?

The systems described in Section 3 are designed for a wide range of purposes, from business collaboration to resisting censorship. Some focus on supporting a specific use case, while others aim to provide a general-purpose communication layer.

N2N and anoNet are the most flexible systems, creating virtual private networks that can be used by a wide range of existing software. Turtle and CSpace provide general-purpose virtual circuits, but existing applications cannot use them without modification. Similarly, Freenet and GNUnet provide general-purpose anonymous storage layers, but they cannot be used transparently by standard software.

Groove, PowerFolder and Shinkuro create shared workspaces with automatic synchronisation, which is useful for collaborative projects but may not be ideal for sharing large collections of files. Groove is also integrated with the Microsoft Office suite, enabling other applications to make use of its private connections.

Most private peer-to-peer networks simply provide a graphical user interface for file sharing and chat and do not attempt to integrate with other applications.

The disadvantage of restricting a private network to a single application is that users must go to the trouble of constructing a new overlay whenever they wish to use a new application; general-purpose networks avoid this duplication of effort. On the other hand, by blurring the line between local and wide area networks, and by supporting existing applications that may not have been designed with security in mind, general-purpose networks may inadvertently undermine their users' security or privacy.

5 Related Research

In a seminal paper that predates most of the systems described in this chapter, Biddle et al. [7] predict that as public peer-to-peer networks come under attack from copyright holders, users will turn to sharing content with their friends through "the darknet" – not a single global network, but rather a patchwork of local networks, technologically isolated but interconnected through their users. “In light of strong cryptography,” Biddle *et al.* argue, “it is hard to imagine how sharing could be observed and prosecuted as long as users do not share with strangers.”

Pouwelse et al. [55] identify five research challenges for peer-to-peer systems: decentralising functionality; maintaining availability in the face of churn; ensuring the integrity of data and metadata; creating incentives to contribute resources; and achieving network transparency across middleboxes. They suggest that all five challenges can be addressed by encouraging users to form cooperative social groups.

A number of researchers have proposed using explicit or implicit information about the social connections between users to improve the robustness of peer-to-peer networks [17, 42, 47]. SybilGuard [77] and SybilLimit [76] use the structure of social networks to limit the impact of Sybil attacks on open systems.

Figueiredo et al. [26] describe how to establish VPN connections between users of social networking websites to create “social VPNs”; the websites act as trusted third parties for key exchange. Nagaraja [49] proposes an anonymous communication system that uses a social networking website for key distribution.

Strufe and Reschke [69] describe a peer-to-peer overlay that stores group information as well as file information, so users can create authenticated groups for file sharing and instant messaging. Each group has a single owner who is responsible for adding and removing members.

These systems are not private peer-to-peer networks according to the definition used in this chapter, but they demonstrate some of the advantages of incorporating the social relationships between users into the design of communication networks.

6 Conclusions

This chapter has provided a brief survey of the emerging field of private peer-to-peer networks, which attempt to combine the flexibility and autonomy of peer-to-peer architectures with the confidentiality and authentication of traditional groupware. Deployed systems can be classified along four architectural axes: scale, visibility, centralisation, and application support. Each of these axes involves tradeoffs affecting the robustness, scalability, privacy, and ease of use that the resulting systems can provide.

Private peer-to-peer networks are already being used in fields as diverse as business collaboration, file sharing, social networking, grassroots political activity and censorship-resistant communication. Considering the varied and sometimes conflicting requirements raised by these applications, we do not expect that any single network will be able to meet the needs of all users; instead we will continue to see a range of architectures that are adapted to particular uses.

References

1. Adar, E., Huberman, B.: Free riding on Gnutella. First Monday **5**(10) (2000). URL `http://firstmonday.org/issues/issue5_10/adar/`
2. URL `http://web.archive.org/web/20010801151157/aimster.com/`. Aimster website, archived August 2001, available from `http://web.archive.org/web/20010801151157/aimster.com/`
3. URL `http://www.alliancep2p.com/`. Alliance website, `http://www.alliancep2p.com/`
4. URL `http://anonet.org/`. AnoNet website, `http://anonet.org/`
5. Banerjee, A., Faloutsos, M., Bhuyan, L.: P2P: Is Big Brother watching you? Tech. Rep. UCR-CS-2006-06201, Department of Computer Science and Engineering, University of California, Riverside (2006). URL `http://www1.cs.ucr.edu/store/techreports/UCR-CS-2006-06201.pdf`
6. Bennett, K., Grothoff, C.: GAP - practical anonymous networking. In: Proceedings of the 3rd International Workshop on Privacy Enhancing Technologies (PET 2003), Dresden, Germany, *Lecture Notes in Computer Science*, vol. 2760, pp. 141–160 (2003). URL `http://gnunet.org/download/aff.ps`
7. Biddle, P., England, P., Peinado, M., Willman, B.: The darknet and the future of content protection. In: Proceedings of the 2nd International Workshop on Digital Rights Management (DRM 2002), Washington, DC, USA, *Lecture Notes in Computer Science*, vol. 2696, pp. 155–176 (2003). URL `http://msl1.mit.edu/ESD10/docs/darknet5.pdf`
8. Bricklin, D.: Friend-to-friend networks (2000). URL `http://www.bricklin.com/f2f.htm`. Available from `http://www.bricklin.com/f2f.htm`
9. Bustamante, F., Qiao, Y.: Friendships that last: Peer lifespan and its role in P2P protocols. In: 8th International Workshop on Web Content Caching and Distribution, Hawthorne, NY, USA (2003). URL `http://2003.iwcw.org/papers/bustamante.pdf`
10. Cederlöf, J.: Web of trust statistics and pathfinder. URL `http://www.lysator.liu.se/~jc/wotsap/`. Available from `http://www.lysator.liu.se/~jc/wotsap/`
11. Christin, N., Weigend, A., Chuang, J.: Content availability, pollution and poisoning in file sharing peer-to-peer networks. In: ACM Conference on Electronic Commerce, Vancouver, Canada (2005). URL `http://www.weigend.com/ChristinWeigendChuang2005.pdf`
12. Clarke, I.: Project status update, and request for your help (2005). URL `http://archives.freenetproject.org/message/20050914.103042.4b8aac35.en.html`. Available from `http://archives.freenetproject.org/message/20050914.103042.4b8aac35.en.html`
13. Clarke, I., Sandberg, O.: Routing in the dark: Scalable searches in dark P2P networks. In: DefCon 13, Las Vegas, NV, USA (2005). URL `http://www.math.chalmers.se/~ossa/defcon13/vegas1_print.pdf`
14. Clarke, I., Sandberg, O., Wiley, B., Hong, T.: Freenet: A distributed anonymous information storage and retrieval system. In: Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, *Lecture Notes in Computer Science*, vol. 2009, pp. 46–66 (2001). URL `http://www.cl.cam.ac.uk/~twh25/academic/papers/icsi-revised.pdf`
15. URL `http://cryptic6.sourceforge.net/`. Cryptic6 website, `http://cryptic6.sourceforge.net/`
16. URL `http://www.cspace.in/`. CSpace website, `http://www.cspace.in/`
17. Danezis, G., Lesniewski-Laas, C., Kaashoek, M., Anderson, R.: Sybil-resistant DHT routing. In: 10th European Symposium on Research in Computer Security (ESORICS 2005), Milan, Italy (2005). URL `http://www.cl.cam.ac.uk/users/gd216/sybildht.pdf`
18. URL `http://dcplusplus.sourceforge.net/`. DC++ website, `http://dcplusplus.sourceforge.net/`

19. Deibert, R., Palfrey, J., Rohozinski, R., Zittrain, J. (eds.): Access Denied: The Practice and Policy of Global Internet Filtering. MIT Press (2008)
20. Deri, L., Andrews, R.: N2N: A layer two peer-to-peer VPN. In: Proceedings of the 2nd International Conference on Autonomous Infrastructure, Management and Security (AIMS 2008), Bremen, Germany, *Lecture Notes in Computer Science*, vol. 5127, pp. 53–64 (2008). URL `http://luca.ntop.org/n2n.pdf`
21. Douceur, J.: The Sybil attack. In: Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA, *Lecture Notes in Computer Science*, vol. 2429, pp. 251–260 (2002). URL `http://www.cs.rice.edu/Conferences/IPTPS02/101.pdf`
22. URL `http://easta.sourceforge.net/`. Easter website, `http://easta.sourceforge.net/`
23. Ek, M., Hultin, F., Lindblom, J.: WASTE peer-to-peer protocol (2005). URL `http://prdownloads.sourceforge.net/j-waste/waste_documentation-1.1.pdf?download`. Reverse-engineered protocol documentation, available from `http://prdownloads.sourceforge.net/j-waste/waste_documentation-1.1.pdf?download`
24. Electronic Frontier Foundation: RIAA v. the people: Four years later (2007). URL `http://w2.eff.org/IP/P2P/riaa_at_four.pdf`. Available from `http://w2.eff.org/IP/P2P/riaa_at_four.pdf`
25. Electronic Privacy Information Center, Privacy International: Privacy and Human Rights 2006: An International Survey of Privacy Laws and Developments. Washington, DC: Electronic Privacy Information Center (2007). URL `http://www.privacyinternational.org/article.shtml?cmd[347]=x-347-559458`
26. Figueiredo, R., Boykin, O., St. Juste, P., Wolinsky, D.: Social VPNs: Integrating overlay and social networks for seamless P2P networking. In: Workshop on Collaborative Peer-to-Peer Systems (COPS), Rome, Italy (2008). URL `http://ast-deim.urv.cat/wwiki/images/1/18/Socialvpn-cops08.pdf`
27. Ford, B., Srisuresh, P., Kegel, D.: Peer-to-peer communication across network address translators. In: USENIX Annual Technical Conference, Anaheim, CA, USA (2005). URL `http://www.usenix.org/events/usenix05/tech/general/full_papers/ford/ford.pdf`
28. Fredriksen, L.: Securing private peer-to-peer networks. Master's thesis, University of Tromsø(2007). URL `http://www.ub.uit.no/munin/handle/10037/1197`
29. Freedman, M., Morris, R.: Tarzan: A peer-to-peer anonymizing network layer. In: 9th ACM Conference on Computer and Communications Security (CCS 2002), Washington, DC, USA (2002). URL `http://pdos.csail.mit.edu/tarzan/docs/tarzan-ccs02.pdf`
30. Friedman, E., Resnick, P.: The social cost of cheap pseudonyms. Journal of Economics and Management Strategy **10**(2), 173–199 (2001). URL `http://www.si.umich.edu/~presnick/papers/identifiers/081199.pdf`
31. URL `http://galet.sourceforge.net/`. Galet website, `http://galet.sourceforge.net/`
32. URL `http://www.lugato.net/gazzera/`. Gazzera website, `http://www.lugato.net/gazzera/`
33. URL `http://www.gigatribe.com/`. GigaTribe website, `http://www.gigatribe.com/`
34. Gonze, L.: Friendnet (2002). URL `http://www.oreillynet.com/pub/wlg/2428`. Available from `http://www.oreillynet.com/pub/wlg/2428`
35. URL `http://www.groove.net/`. Groove Networks website, `http://www.groove.net/`
36. Grothoff, C., Grothoff, K., Horozov, T., Lindgren, J.: An encoding for censorship-resistant sharing (2005). URL `http://gnunet.org/download/ecrs.ps`. GNUnet white paper, available from `http://gnunet.org/download/ecrs.ps`

37. Guha, S., Daswani, N., Jain, R.: An experimental study of the Skype peer-to-peer VoIP system. In: Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS '06), Santa Barbara, CA, USA, pp. 1–6 (2006). URL http://saikat.guha.cc/pub/iptps06-skype.pdf
38. Guha, S., Francis, P.: Characterization and measurement of TCP traversal through NATs and firewalls. In: Internet Measurement Conference (IMC 2005), Berkeley, CA, USA (2005)
39. URL http://hybrid-share.sourceforge.net/. Hybrid Share website, http://hybrid-share.sourceforge.net/
40. URL http://www.i2p2.de/. I2P website, http://www.i2p2.de/
41. Kugler, D.: An analysis of GNUnet and the implications for anonymous, censorship-resistant networks. In: Proceedings of the 3rd International Workshop on Privacy Enhancing Technologies (PET 2003), Dresden, Germany, *Lecture Notes in Computer Science*, vol. 2760, pp. 161–176 (2003). URL http://gnunet.org/papers/GNUnet_pet.pdf
42. Lesniewski-Laas, C.: A Sybil-proof one-hop DHT. In: 1st International Workshop on Social Network Systems (SocialNets 2008), Glasgow, Scotland (2008). URL http://pdos.csail.mit.edu/papers/sybil-dht-socialnets08.pdf
43. Li, J., Dabek, F.: F2F: Reliable storage in open networks. In: 5th International Workshop on Peer-to-Peer Systems (IPTPS '06), Santa Barbara, CA, USA (2006). URL http://pdos.csail.mit.edu/~jinyang/pub/iptps-f2f.pdf
44. Liang, J., Kumar, R., Xi, Y., Ross, K.: Pollution in P2P file sharing systems. In: IEEE Infocom, Miami, FL, USA (2005). URL http://cis.poly.edu/~ross/papers/pollution.pdf
45. Liebowitz, S., Margolis, S.: Network externalities (effects). In: New Palgrave Dictionary of Economics and the Law, MacMillan (1998). URL http://wwwpub.utdallas.edu/~liebowit/palgrave/network.html
46. Lua, E., Crowcroft, J., Pias, M., Sharma, R., Lim, S.: A survey and comparison of peer-to-peer overlay network schemes. IEEE Communications Surveys and Tutorials **7**(2) (2005). URL http://www.cl.cam.ac.uk/users/ek125/lua.pdf
47. Marti, S., Ganesan, P., Garcia-Molina, H.: SPROUT: P2P routing with social networks. In: Proceedings of the 9th International Conference on Extending Database Technology (EDBT 2004), Heraklion, Crete, Greece, *Lecture Notes in Computer Science*, vol. 3268, pp. 425–435 (2004). URL http://home.comcast.net/~sergiomarti/papers/2004-5.pdf
48. Matějka, P.: Security in peer-to-peer networks. Master's thesis, Department of Software Engineering, Charles University, Prague (2004). URL http://www.turtle4privacy.org/documents/masterThesis.pdf
49. Nagaraja, S.: Anonymity in the wild: Mixes on unstructured networks. In: Proceedings of the 7th Workshop on Privacy Enhancing Technologies (PET 2007), Ottawa, Canada, pp. 254–272 (2007). URL http://www.cl.cam.ac.uk/~sn275/papers/unstructured-mixes.pdf
50. URL http://web.archive.org/web/20050627012020/www.neo-modus.com/. NeoModus Direct Connect website, archived June 2005, available from http://web.archive.org/web/20050627012020/www.neo-modus.com/
51. URL http://sysnet.ucsd.edu/octopod/. Octopod website, http://sysnet.ucsd.edu/octopod/
52. URL http://www.phex.org/. Phex website, http://www.phex.org/
53. Phex wiki: Creating a private network. URL http://www.phex.org/wiki/index.php/Creating_a_private_Network. Available from http://www.phex.org/wiki/index.php/Creating_a_private_Network
54. Popescu, B., Crispo, B., Tanenbaum, A.: Safe and private data sharing with Turtle: Friends team-up and beat the system. In: 12th International Workshop on Security Protocols, Cambridge, UK (2004). URL http://www.cs.vu.nl/~bpopescu/papers/sec_prot04/sec_prot04.pdf

55. Pouwelse, J., Garbacki, P., Wang, J., Bakker, A., Yang, J., Iosup, A., Epema, D., Reinders, M., van Steen, M., Sips, H.: Tribler: A social-based peer-to-peer system. In: 5th International Workshop on Peer-to-Peer Systems (IPTPS '06), Santa Barbara, CA, USA (2006). URL http://iptps06.cs.ucsb.edu/papers/Pouw-Tribler06.pdf
56. URL http://www.powerfolder.com/. PowerFolder website, http://www.powerfolder.com/
57. Quercia, D., Hailes, S., Capra, L.: TATA: Towards anonymous trusted authentication. In: Proceedings of the 4th International Conference on Trust Management (iTrust 2006), Pisa, Italy, pp. 313–323 (2006). URL http://www.cs.ucl.ac.uk/staff/D.Quercia/publications/querciaTATA06.pdf
58. Rennhard, M., Plattner, B.: Practical anonymity for the masses with MorphMix. In: Proceedings of the 8th International Financial Cryptography Conference (FC 2004), Key West, FL, USA, *Lecture Notes in Computer Science*, vol. 3110, pp. 233–250 (2004). URL http://www.tik.ee.ethz.ch/~rennhard/publications/FC2004.pdf
59. URL http://retroshare.sourceforge.net/. Retroshare website, http://retroshare.sourceforge.net/
60. Rhea, S., Godfrey, B., Karp, B., Kubiatowicz, J., Ratnasamy, S., Shenker, S., Stoica, I., Yu, H.: OpenDHT: A public DHT service and its uses. In: SIGCOMM 2005, Philadelphia, PA, USA (2005). URL http://berkeley.intel-research.net/sylvia/f230-rhea.pdf
61. Risson, J., Moors, T.: Survey of research towards robust peer-to-peer networks: Search methods. Tech. Rep. UNSW-EE-P2P-1-1, University of New South Wales (2004). URL http://uluru.ee.unsw.edu.au/~john/tr-unsw-ee-p2p-1-1.pdf
62. Rosenberg, J., Weinberger, J., Huitema, C., Mahy, R.: RFC 3489: STUN - simple traversal of user datagram protocol (UDP) through network address translators (NATs) (2003). URL http://www.ietf.org/rfc/rfc3489.txt
63. Sandberg, O.: Distributed routing in small-world networks. In: 8th Workshop on Algorithm Engineering and Experiments (ALENEX06), Miami, FL, USA (2006). URL http://www.math.chalmers.se/~ossa/swroute.pdf
64. Sandberg, O., Clarke, I.: The evolution of navigable small-world networks. Tech. Rep. 2007:14, Department of Computer Science and Engineering, Chalmers University of Technology (2007). URL http://www.math.chalmers.se/~ossa/evolution.pdf
65. Saroiu, S., Gummadi, P.K., Gribble, S.: A measurement study of peer-to-peer file sharing systems. In: Multimedia Computing and Networking (MMCN '02) (2002). URL http://www.cs.washington.edu/homes/gribble/papers/mmcn.pdf
66. URL http://shinkuro.com/. Shinkuro website, http://shinkuro.com/
67. URL http://code.google.com/p/sneakernet/. Sneakernet website, http://code.google.com/p/sneakernet/
68. URL http://www.ziggy.speedhost.com/bdsock.html. SockeToome website, http://www.ziggy.speedhost.com/bdsock.html
69. Strufe, T., Reschke, D.: Efficient content distribution in semi-decentralized peer-to-peer networks. In: Proceedings of the 8th International Netties Conference, Ilmenau, Germany, pp. 33–38 (2002). URL http://eris.prakinf.tu-ilmenau.de/pub/papers/strufe02efficient.pdf
70. Stutzbach, D., Rejaie, R.: Towards a better understanding of churn in peer-to-peer networks. Tech. Rep. UO-CIS-TR-04-06, Department of Computer Science, University of Oregon (2004). URL http://www.barsoom.org/~agthorr/papers/tr04-06.pdf
71. Svensson, P.: Comcast blocks some internet traffic. Associated Press (2007). URL http://www.msnbc.msn.com/id/21376597/
72. Udell, J., Asthagiri, N., Tuvell, W.: Security. In: A. Oram (ed.) Peer-to-Peer: Harnessing the Power of Disruptive Technologies, chap. 18. O'Reilly (2001). This chapter describes Groove.
73. Veiga, A.: Music labels tap downloading networks. Associated Press (2003). URL http://www.usatoday.com/tech/webguide/music/2003-11-14-sharestats_x.htm

74. URL http://waste.sourceforge.net/. WASTE website, http://waste.sourceforge.net/
75. Wright, M., Adler, M., Levine, B., Shields, C.: An analysis of the degradation of anonymous protocols. In: ISOC Symposium on Network and Distributed System Security, San Diego, CA, USA (2002). URL http://www.freehaven.net/anonbib/cache/wright02.pdf
76. Yu, H., Gibbons, P., Kaminsky, M., Xiao, F.: SybilLimit: A near-optimal social network defense against Sybil attacks. In: IEEE Symposium on Security and Privacy, Oakland, CA, USA (2008). URL http://www.comp.nus.edu.sg/~yuhf/yuh-sybillimit.pdf
77. Yu, H., Kaminsky, M., Gibbons, P., Flaxman, A.: SybilGuard: Defending against Sybil attacks via social networks. In: SIGCOMM 2006, Pisa, Italy (2006). URL http://sigcomm06.stanford.edu/discussion/getpaper.php?paper_id=26

Part VII
Broadcast and Multicast Services

Gossip-Based Broadcast

João Leitão, José Pereira, and Luís Rodrigues

Abstract Gossip, or epidemic, protocols have emerged as a powerful strategy to implement highly scalable and resilient reliable broadcast primitives on large scale peer-to-peer networks. Epidemic protocols are scalable because they distribute the load among all nodes in the system and resilient because they have an intrinsic level of redundancy that masks node and network failures. This chapter provides an introduction to gossip-based broadcast on large-scale unstructured peer-to-peer overlay networks: it surveys the main results in the field, discusses techniques to build and maintain the overlays that support efficient dissemination strategies, and provides an in-depth discussion and experimental evaluation of two concrete protocols, named HyParView and Plumtree.

1 Introduction

Many distributed systems benefit from the availability of a reliable broadcast service. Informally, reliable broadcast ensures that all correct participants receive all broadcast messages, even in the presence of network omissions or node failures. Gossip protocols have emerged as a highly scalable and resilient approach to implement reliable broadcast [1, 10, 14, 17]. In a typical gossip protocol, when a node wants to broadcast a message, it selects t nodes from the system at random (t is a configuration parameter called *fanout*) and sends the message to them; upon receiving a message for the first time, each node repeats this procedure [17].

João Leitão
INESC-ID / IST, Lisbon, Portugal, e-mail: jleitao@gsd.inesc-id.pt

José Pereira
University of Minho, Braga, Portugal, e-mail: jop@di.uminho.pt

Luís Rodrigues
INESC-ID / IST, Lisbon, Portugal, e-mail: ler@ist.utl.pt

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_29, © Springer Science+Business Media, LLC 2010

Gossip protocols are interesting because they are highly resilient (they have an intrinsic level of redundancy that allows them to mask node and network failures), scalable (they distribute the load among all nodes in the system), and simple to implement. In fact, gossip protocols have been proposed as a building block to solve several other problems in distributed systems such as: consistency management in replicated databases [6], failure detection [25], system monitoring [24], and distributed publish-subscribe brokers [10].

Strictly speaking, in order to select a gossip peer at random among all system nodes, each node would be required to know the entire system membership. Unfortunately, this solution is not scalable, not only due to the large number of nodes that may constitute the membership view, but also due to the cost of maintaining the complete membership up-to-date in face of the typical dynamics of large-scale systems. To overcome this problem, several gossip protocols rely on *partial views* [8, 10, 20, 23] instead of full membership information. A partial view is a small subset of the entire system membership. In each gossip step[1] a node extracts a random sample of peers as targets for gossip messages. The aim of a *membership service* (also called a peer sampling service [15]) is to maintain these partial views in such a way that a random selection of gossip peers from partial views approximates the random selection from the full membership.

Unfortunately, if each node has only a partial view, the system becomes more vulnerable to the effect of node failures. In particular, if a large number of nodes fail, the partial view of a node may include only failed nodes, therefore the network may become disconnected and several nodes may become isolated. Even if the network remains connected, if failed nodes are select as targets for gossip, specially in early gossip steps, the reliability of the gossip broadcast is severely disrupted. Also, the membership service may take a long time to restore partial views, with a negative impact on the reliability of all messages disseminated meanwhile. Thus, the quality of partial views is of paramount importance for the operation of gossip protocols.

A disadvantage of gossip-based broadcast protocols is that they incur in an excessive message overhead in order to enforce high reliability. This is not the case with structured broadcast protocols, such as the ones that rely on tree-construction but, on the other hand, structured protocols are very fragile in the presence of failures, lacking the natural resilience of epidemic protocols. A promising approach is to develop new broadcast primitives which combine gossip-based and tree-based approaches; in such a way one can benefit from the scalability and resilience of pure gossip-based and approximate the efficiency of the broadcast mechanism to that of tree-based solutions.

In this chapter we start by describing gossip protocols and their relation with unstructured overlay networks. We also discuss several strategies used in the development of gossip protocols and the characteristics of the unstructured overlay networks that are used to support those strategies. We then describe two recently proposed

[1] We consider that a gossip step happens each time a node transmits (or retransmits) a broadcast message to a set of peers.

gossip-based protocols: a protocol to build an highly resilient unstructured overlay network, called HyParView, and a broadcast protocol that approaches the efficiency of structured tree-based protocols, called Plumtree. Experimental results show that when combined, these protocols allows one to build a broadcast primitive which is more resilient and more efficient than other gossip-based broadcast protocols.

2 Gossip-Based Broadcast

The basic idea behind gossip-based broadcast protocols is to have all participants in the protocol collaborating equally to disseminate information. To this end, when a node wishes to send a broadcast message, it selects t nodes at random – its *gossip targets* – and sends the message to them (t is a typical configuration parameter called *fanout*, which is detailed later in Section 2.1). Upon receiving a message for the first time, a node repeats this process: by selecting t gossip targets at random and forwarding the message to them.

If a node receives the same message twice – which is possible, as each node selects its gossip targets in an independent way (without being aware of gossip targets selected by other nodes) – it simply discards the message. To allow this, each node has to keep track of which messages it has already seen and delivered. The history of message identifiers may grow indefinitely during the execution of the protocol, unless some purging scheme is applied to garbage-collect obsolete entries. The challenge of selecting an adequate strategy to purge message histories is out of the scope of this chapter, and it has already been addressed by previous work, for instance in [18].

The simple operation model of gossip protocols not only provides high scalability but also a high level of fault tolerance, as its intrinsic redundancy is able to mask network omissions and also node failures.

2.1 Parameters

Gossip protocols can be parameterized in order to better control their operation. The two most relevant parameters associated with the configuration of gossip protocols can be described as follows.

Fanout: This is the number of nodes that are selected as gossip targets by a node at each gossip step in order to retransmit the message. There is a trade-off associated with this parameter between desired reliability level and redundancy level of the protocol. High fanout values ensure higher levels of fault tolerance level (increasing the probability of reaching all nodes in the system, or in other words, of atomic delivery as defined in [17]) but also generate more redundant network traffic.

Maximum rounds: This is the maximum number of times a given gossip message is retransmitted by nodes. Each message is transmitted with a *round value* – initially with a value of zero – which is increased each time a node retransmits the message. Nodes will only retransmit a message if its *round value* is smaller than the *maximum rounds* parameter. A gossip protocol can operate in one of the two following modes:

- *Unlimited mode*: In this mode of operation the parameter *maximum rounds* is undefined and there is no specific limit to the number of retransmissions executed to each gossip message.
- *Limited mode*: In this mode of operation the parameter *maximum rounds* is set to some integer value (higher than 0), effectively limiting the maximum hops executed by each message in the overlay.[2]

The reader should notice that by limiting the number of maximum gossip rounds one can also limit the maximum number of receivers. For instance, if the message is forwarded by flooding, in order to reach the entire system the maximum number of rounds must be set to a value equal or higher than the network diameter.

There is an inherent trade-off between reliability and redundancy level associated with the use of maximum number of rounds attribute. In unlimited mode (or configuring the *maximum rounds* parameter with high values) there is a higher probability of achieving atomic delivery but, on the other hand, more redundant messages are produced.

2.2 Strategies

We distinguish the following three basic approaches to implement a gossip protocol. These different strategies are distinguished by the way a message is disseminated to neighbors, and who initiates the gossip exchange: receiver or the sender.

Eager push approach: Nodes send the full payload to random selected peers as soon as they receive a message for the first time. This is an approach initiated by the sender.

Pull approach: Periodically, nodes query random selected peers for information about recently received or available messages. When they become aware of a message they did not received yet, they explicitly request to that neighbor the payload of that message. This is a strategy that works best as a complement to a best-effort broadcast mechanism (i.e., IP Multicast [5]).

Lazy push approach: When a node receives a message for the first time, it gossips only the message identifier (for instance, a hash of the message) and not the full payload. If peers receive an identifier for a message they have not yet received, they explicitly request the payload from the sender.

[2] Neighboring relations between nodes form an overlay network, as it will be discussed in Section 2.4.

The reader should notice that there is a trade-off when selecting eager push, pull, or lazy push strategies. Eager push strategies produce more redundant traffic but they also achieve lower latency than the remaining strategies (that require at least an extra round trip time to produce a delivery). From a latency perspective, lazy push gossip is very similar to pull gossip.

Another important practical aspect to retain is that, contrary to pull/lazy push gossip, eager push gossip is not required to maintain copies of delivered messages for later retransmission upon request. Hence, pull/lazy push gossip approaches are more demanding in terms of memory usage.

These basic strategies can also be combined to develop more complex gossip strategies, that usually, try to benefit from the strong aspects of each individual strategy. As the reader can guess, the number of possible combinations is very high, as one can use different algorithms to switch from one strategy to the other. In the following, we depict two of these hybrid strategies.

Eager push and pull approach: Gossip is executed in two distinct phases. A first phase uses push gossip to disseminate a message in a best-effort manner to a maximum of participants (in this phase, the configuration of the push gossip can be somewhat conservative). A second phase of pull gossip is used to recover from omissions that may occur in the first phase. The idea is to lower the amount of redundancy of the gossip process, without decreasing its efficiency. However, the use of pull gossip for recovery increases the global delivery latency.

Eager push and lazy push: Gossip is executed by applying eager push to a sub-set of the gossip targets of each node. The selection of the sub-set of peers can be made using several strategies (examples can be found in [2, 19]). Lazy push is used in the remaining gossip targets to recover from omissions and ensure the gossip properties.

2.3 Peer Sampling Service

A *peer sampling service* (as described in [15]) is a service that allows nodes to know a sub-set of the full group of nodes executing the protocol. The interface of this service is quite simple and may only export the following two methods:

INIT(): This method initializes the service, if it has not been initialized before. If node p returns from the INIT() call, there should be a non-zero probability of other participating nodes getting p as a return value when calling the GETPEER() method.

GETPEER(): This method returns the identifier of a participating node, as long as there exists more than one node executing the service. In most cases, the returned node should be selected at random across nodes that have called the `init()` method, although the specific distribution (i.e., correlation with returned identifiers from previous call of this method) is implementation dependent.

The GETPEER() method is enough to support the operation of any gossip protocol – as a node can call repeatedly this method if it requires more than one peer – however, in practice, this method can be redefined as:

GETPEER($n, veto$): Where n is an integer greater than zero and $veto$ is a node identifier. This method returns a list with, at most, n identifiers of participating nodes that does not contain the $veto$ identifier nor the identifier of the invoking node. This method should be called with n equal to the fanout used by the gossip protocol and $veto$ should be the identifier of the node who sent the message to the invoking node.[3]

The semantics of GETPEER($n, veto$) can be informally described as: Return a sample of at most n peer identifiers in the system which does not contain the identifier of either the invoking peer or the peer identified by $veto$ (if $veto$ is not *null*).

2.4 Partial View

A *partial view* is a set of node identifiers provided to each node. This set should be much smaller than the full system membership. The size constraint is related with scalability requirements. A viable strategy to promote scalability, if for the partial view size to grow logarithmic with relation to the total number of processes in the system. Typically, an identifier is a tuple $(ip : port)$ that allows a node to be reached. In some cases, such as in DHT's, the identifier is a random bit string which is used both to map the nodes in a specific position of the overlay and to support routing [26, 29].

A membership protocol is in charge of initializing and maintaining the partial views for each node in face of dynamic changes in the system membership. For instance, when a new node joins the system, its identifier should be added to the partial view of (some) other nodes. Furthermore, the new node has to build its own partial view, including identifiers of peers already in the system. Also, if a node fails or leaves the system, its identifier should be removed from all partial views as soon as possible.

Partial views establish *neighboring* associations among nodes. Therefore, partial views define an overlay network or, in other words, partial views establish an oriented graph that captures the neighbor relation between all the nodes executing the protocol. In this graph, nodes are represented by a vertex while a neighbor relation is represented by an arc originating from the node that contains the target node in his partial view.

The favored implementation of a peer sampling service is to use a membership service that maintains a partial view of participating nodes at each node. The selection of nodes to serve as gossip target is then performed locally using the partial view.

[3] When a node wishes to send a message by the first time, the $veto$ argument takes the *null* value.

2.5 Strategies to Maintain Partial Views

We identify two main strategies that can be employed to maintain partial views, namely:

Reactive strategy: In this type of approach, a partial view is only updated in response to some external event that affects the overlay (i.e., a node joining or leaving the system). In stable conditions, partial views remains unaltered. Scamp [12, 13] is an example of a reactive algorithm.[4]

Cyclic strategy: In this type of approach, a partial view is updated every ΔT time units, as a result of some periodic process that usually involves the exchange of information with one or more neighbors. Therefore, a partial view may be updated even if the global system membership is stable. Cyclon (which is further discussed in Section 2.8.1) is an example of a cyclic algorithm.

Reactive strategies usually rely on some failure detection mechanism to trigger the update of partial views when a node fails. If the failure detection mechanism is fast and accurate, reactive mechanisms can provide faster response to failures than cyclic approaches. This approach is also more efficient as it avoids the communication overhead required by the cyclic strategy to update views.

On the other hand, a cyclic strategy allows each node to select a wide range of distinct nodes as gossip targets for different messages even in stable conditions, as the elements of each partial view are continually changing, making this strategy more close, in nature, to the original intuition of gossip protocols.

2.6 Partial View Properties

In order to be useful, namely to support fast message dissemination and high level of fault tolerance to node failures, partial views must own a number of important properties. These properties, that can be used to measure the quality of partial views, are intrinsically related with graph properties of the overlay they define. Some relevant properties are:

Connectivity: The overlay defined by the partial views should be connected. To consider an overlay as connected, there should be at least one path from each node to all other nodes.[5] If this property is not met, isolated nodes will not be able to receive or send broadcast messages.

Degree Distribution: In an undirected graph, the degree of a node is simply the number of edges of the node. Given that partial views define a directed graph, it is importantto distinguish *in-degree* from *out-degree* of a node. The in-degree of

[4] To be precise, Scamp is not purely reactive as it includes a lease mechanism that forces nodes to rejoin periodically.

[5] Obviously, if the graph is directed, the path between nodes have to respect the direction of arcs.

a node n is the number of nodes that have n's identifier in their partial view; it provides a measure of the reachability of a node in the overlay and also affects the probability and the maximum amount of redundant messages that n can receive. The out-degree of a node n is the number of nodes in n's partial view; it is a measure of the node contribution to the membership protocol and consequently a measure of the importance of that node to maintain the overlay.

If the probability of failure is uniformly distributed in the node space, for improved fault-tolerance both the in-degree and out-degree should be evenly distributed across all nodes executing the membership protocol.

Average Path Length: A path between two nodes in the overlay is the set of edges that a message has to cross from one node to the other. The average path length is the average of all shortest paths between all pair of nodes in the overlay. This property is closely related to the overlay diameter. To ensure the efficiency of the overlay for information dissemination, it is essential to enforce low values of the average path length, as this value is related to the time (and number of hops in the overlay) a message will require to reach all nodes. As the reader could expect, the average path length is affected by the clustering coefficient. A high clustering coefficient will increase the average path length.[6]

Clustering Coefficient: The clustering coefficient of a node is the number of edges between that node's neighbors divided by the maximum possible number of edges across those neighbors. This metric indicates a density of neighbor relations across the neighbors of a given node, having it's value between 0 and 1. The clustering coefficient of a graph is the average of clustering coefficients across all nodes. This property has a high impact on the number of redundant messages received by nodes when disseminating data, where a high value to clustering coefficient will produce more redundant messages. It also has an impact in the fault-tolerant properties of the graph, given that areas of the graph that exhibit high values of clustering will more easily be isolated from the rest of the graph.

Accuracy: Accuracy of a node is defined as the number of neighbors of that node that have not failed divided by the total number of neighbors of that node. The accuracy of a graph is the average of the accuracy of all correct nodes. Accuracy has high impact in the overall reliability of any dissemination protocol using an underlying membership protocol to select its gossip targets. If the graph accuracy values are low, the number of failed nodes selected as gossip targets will be higher, which in turn, can disrupt the gossip process. To avoid this, higher fanout values must be used to mask the selection of failed nodes.

[6] The reader should notice that this property is only meaningful if the property of connectivity is met. If the overlay is not connected then at least one node in unreachable which translates into a infinite shortest path between all other nodes and that node.

2.7 Performance Metrics

We now introduce three metrics that we will use to evaluate the performance of gossip-based broadcast protocols.

Reliability: Reliability is defined as the percentage of active nodes in a system which delivers a given broadcast message. A reliability value of 100% is indicative that the broadcast protocol was successful in delivering a given message to all active nodes or, in other words, that the broadcast process resulted in an atomic broadcast as defined in [17].

As the reader should expect, the goal of a gossip-based broadcast protocol is to obtain a reliability of 100% despite network omissions or node failures.

Relative Message Redundancy (RMR): The relative message redundancy, or simply RMR, is a metric that captures the message overhead in a gossip-based broadcast protocol [19]. It is defined as:

$$\left(\frac{m}{n-1}\right)-1$$

where m is the total number of payload messages exchanged during the broadcast process and n is the total number of nodes that delivers the broadcasted message. This metric is only applicable when at least 2 nodes are able to deliver the message.

A RMR value of zero means that there is exactly one payload message exchanged for each node in the system, which is clearly the optimal value. By opposition, high values of RMR are indicative of a broadcast strategy that shows a poor network usage. Note that it is possible to achieve a very low RMR by failing to be reliable. Therefore, a broadcast protocol should aim to combine low RMR values with high reliability values. Furthermore, RMR values are only comparable for protocols that exhibit similar reliability. Finally, note that in pure gossip approaches, RMR is closely related with the protocol fanout, as the value of this tends to $fanout-1$.

Control messages are not considered by this metric. The reason behind this decision is twofold: first, control messages are typically much smaller than payload messages and thus, they are not the main source of contribution to the exhaustion of network resources; secondly, control messages can usually be sent using delay and piggyback strategies, providing a better usage of the available network resources.

Last Delivery Hop (LDH): The last delivery hop, or simply LDH, is a metric which measures the round number of the last message which is delivered by a gossip-based broadcast protocol or, in other words, it measures the maximum number of hops performed by a message successfully delivered. Naturally, a gossip-based broadcast protocol should aim at minimizing this metric.

This metric in intuitively related with the diameter of the overlay network used to disseminate messages. Moreover, it can also provide some comparative measure relatively to the latency of a gossip-based broadcast protocol. When all links exhibit the same latency, the latency of a gossip broadcast is simply the last deliver hop multiplied by the *per hop* latency.

2.8 An Overview of Existing Protocols

Several gossip-based protocols have been proposed in the literature. In this section we describe some of these works, addressing three different types of protocols. We start with gossip based membership protocols, that build and maintain unstructured overlay network, which serve as substrate for the operation of gossip-based broadcast protocols. We then present a few gossip-based broadcast protocols.

We conclude by surveying application level multicast protocols that rely in some sort of tree-like structure to support data dissemination. Some of these protocols are not gossip-based broadcast schemes. However, they rely in peer-to-peer structured multicast solutions, being a complementary approach to gossip-based broadcast protocols.

2.8.1 Gossip-Based Membership Protocols

Scamp

Scamp [12, 13], is a reactive membership protocol that maintains two separate views, a *PartialView* from which nodes select their gossip targets, and a *InView* with nodes from which they receive gossip messages. One interesting aspect of this protocol is that the *PartialView* does not have a fixed size; it grows to values that are distributed around $\log n$, where n is the total number of nodes executing the protocol, without n being known by any node.

When a new node wishes to join the overlay, it has to know a node that already belongs to the overlay, to which it sends a *new subscription request*. Upon receiving this request, a node forwards it to all neighbors that belong to its *PartialView* in the form of a *forwarded subscription request*; it also creates c additional copies of this *forwarded subscription request* that are forwarded to c random neighbors from the *PartialView*; c is a configuration parameter that is related with the level of fault tolerance supported by this protocol, as it will affect the global distribution of degree (in-degree and out-degree) values across the overlay. Higher values of c will produce overlays in which nodes have, on average, higher degrees. In turn, this will also impact network usage, as well as other graph properties.

Upon receiving a *forwarded subscription request* a node integrates the new member in its local *PartialView* (if the node is not already present) with a probability p, where p is equal to $1/(1+\mathit{sizeof}(\mathit{PartialView}))$. If the node does not integrate the new member, itforwards the request to a random neighbor in his own *PartialView*.

To avoid these messages to be forwarded an infinite number of times, which is more probable when the number of nodes in the overlay is small, there is an upper limit to the number of times a node can forward the same message. When this limit is reached, the message is simply dropped.

The *InView* is used when a node wishes to leave the overlay. An unsubscribing node, say n_u, will send to some of its peers[7] in the *InView* a *replace request* containing an element from its *PartialView*, say n_p. The node that receives this request will replace in its *Partial View* the identifier of n_u with the received identifier n_p. To the remaining nodes in its *InView*, n_u will simply send a request asking them to remove its own identifier from their *PartialView*.

In order to recover from node isolation, this algorithm uses a mechanism in which nodes periodically send heartbeat messages to all members of their *PartialView*. If a node does not receive a heartbeat for a long time, it assumes that it has become isolated, and it sends a *new subscription request* to a random node in his own *PartialView* in order to rejoin the overlay.

When a node fails (i.e., leaves the system without executing the un-subscription procedure), its identifier will remain in the *PartialViews* of some correct nodes, which means that it can still be selected by those nodes as a gossip target. In order to (eventually) purge these identifiers from the *PartialViews* of correct nodes, Scamp relies in a *lease mechanism*. When a node joins the overlay, its subscription has a finite lifetime which is called its *lease time*. When the lease of a node subscription expires, all peers containing that node identifier in their *PartialView* should delete it. Each node is responsible to rejoin the overlay through a *new subscription request* sent to a random peer in its *PartialView* before the expiration of the *lease time* of its last subscription. The *lease time* of each subscription might be set individually by each node (sending information relative to it in the *new subscription* request), or be enforced through a global configuration parameter that affects all nodes.

Cyclon

Cyclon, is a cyclic membership protocol where nodes maintain a fixed length *partial view*. The size of the *partial view* is a protocol parameter: it takes into account the maximum number of nodes that are expected to participate in the protocol and the desired level of fault-tolerance.

This protocol relies in a *shuffle* operation which is executed every ΔT time units by every node. Basically, to execute a shuffle operation, a node selects the "oldest" node in its partial view and performs an exchange with that node. In the exchange, the node provides to its peer a sample of its partial view and, symmetrically, collects a sample of its peer's partial view.

[7] The number of peers who receive a *replace request* is $sizeof(InView) - c$ this is related with the overlay desired average degree.

2.8.2 Gossip-Based Broadcast Protocols

Bimodal Multicast

Bimodal multicast [1] was one of the pioneer works to combine tree-based best-effort multicast and gossip-based primitives to build reliable multicast in large scale distributed systems. The approach works as follows: in a first phase, broadcast messages are disseminated using the unreliable IP-Multicast [5] primitive; in a second phase, participants engage in gossip exchanges in order to mask omissions that may occur during the first phase. In the second phase, nodes periodically select a peer with whom they exchange a summary of (recently) received messages unique identifiers. After this exchange, nodes explicitly request messages that they miss.

This approach has two major drawbacks. One is that it depends on the availability of IP-multicast, which is not widely deployed in large-scale [9]. To overcome this limitation one could envision to replace IP-multicast by some application-level multicast protocol. Still, a second drawback persists: the approach requires the use of two distinct protocols and therefore, it may present undesired complexity. Moreover, the existence of two distinct protocols may limit the application of optimizations that are only possible if both phases are integrated in a single protocol. Due to these drawbacks, more recent work has favored the use of pure gossip approaches [12, 28].

NeEM

NeEM, or Network Friendly Epidemic Multicast [22], is a gossip protocol that relies on the use of TCP to disseminate information across a gossip-based overlay. In NeEM, the use of TCP aims at eliminating correlated message losses due to network congestion. The authors show that improved gossip reliability can be achieved by leveraging on the flow control mechanisms of TCP.

NeEM controls buffer management directly, by disabling TCP buffers, and applies several purging strategies to discard messages when overflow occurs. This enables the gossip protocol to preserve throughput stability, even when the network became congested and also avoids inter-blocking of nodes, due to exhaustion of TCP reception buffers.

NeEM uses its own (partial view) membership service, which is gossip-based. The membership construction and maintenance is based on random walks in the overlay, with a probabilistically length dependent on a protocol configuration parameter p whose value is fixed a priori. Random walks are used both when a node joins the overlay and in a cyclic manner, to advertise neighbors to random nodes. Nodes which receive these advertisements, will integrate received peers identifiers in their own partial views, replacing random peers if their views are full.

CREW

CREW [8] is a gossip protocol for flash dissemination, i.e., fast simultaneous download of files by a large number of destinations using a combination of pull and push gossip. It uses TCP connections to implicitly estimate available bandwidth thus optimizing the fanout of the epidemic dissemination process. The emphasis of CREW is on optimizing latency, mainly by improving concurrent pulling from multiple sources. A key feature is to maintain a cache of open connections to peers discovered using a random walk protocol, to avoid the latency of opening a TCP connection when a new peer is required by the protocol operation.

CREW uses an underlying membership service, also based on partial views, called Bounce. Bounce is briefly presented in [7] where the authors claim, based on experimental results, that the use of the overlay produced by Bounce is equivalent to the selection of nodes uniformly at random, from all nodes in the system. Bounce relies in random walks to establish neighbor relations between nodes. Random walks are probabilistically terminated according to a certain probability p that depends on the degree of the receiving node, a random factor and finally, to avoid infinite sizes random walks in the overlay, the length of the actual random walk.

2.8.3 Structured Application-Level Multicast

Narada

Narada [4] is an efficient application-level multicast protocol based on dissemination trees that are constructed in two distinct steps. In the first step, the protocol creates and maintains a random and rich connected overlay (that the authors name *mesh*) which attempts to ensure that: (i) the quality of paths between any two nodes in the overlay is comparable to the quality of the unicast path between the same pair of nodes[8] and; (ii) each node has a limited number of neighbors. The overlay is self-organizing and self-improving, adapting itself to changing network conditions to preserve efficiency, by using a set of heuristics that add or remove links among nodes.

In a second step, the overlay is used to create several multicast trees rooted at each source. To this end, a distance vector algorithm is run on top of the overlay. Nodes that wish to join a multicast group explicitly select their parents among their neighbors, using information from the routing algorithm.

Narada is targeted toward medium sized groups; all nodes maintain full membership lists and some additional control information for all other nodes, and consequently it does not scale to very large (and dynamic) systems. Also, the normal dynamics of the algorithm may partition the overlay affecting the global reliability of the system (until the protocol is able to repair the overlay).

[8] In this context, quality refers to application dependent metrics such as latency or bandwidth.

Scribe

Scribe [3, 27] is a scalable application-level multicast infrastructure built on top of a structured overlay network, named Pastry [26]. Scribe supports multicast groups with multiple senders. It constructs a distribution tree for each group, by using a receiver-based strategy and leveraging in Pastry as follows:

Each multicast group has a node that serves both as *rendez-vous point* and as a root for the multicast tree. This node is selected, and can be found by other nodes, taking advantage of Pastry key-based resource location mechanism (using the multicast group name). The keys identifying rendez-vous points can be obtained locally at every node by applying a hash function to the multicast group name. When a node wishes to join a multicast group it uses Pastry to route a JOIN message to the rendez-vous point. This message is used to set-up state in the intermediate nodes along the route, concerning the specific multicast group, thus constructing a distribution tree.

Tree repair is performed as follows: Intermediate nodes periodically send HEARTBEAT messages to nodes which they have registered as being their children. A node will suspect that its parent node has failed when it stops receiving HEARTBEAT messages from it. In this case, the node uses Pastry to route another JOIN message, that is used to set-up another route to the rendez-vous node, effectively healing the tree structure.

All state concerning multicast trees is maintained using a soft state approach. Therefore, nodes have to periodically refresh their interest in belonging to a multicast group by re-sending JOIN messages.

Although Scribe is fault-tolerant and provides a mechanism to handle root failures, it only provides best-effort guarantees. The authors argue that strong reliability and order guarantees (among others) are only required by some applications, and that those properties can be easily provided on top of Scribe.

MON

MON [21], which stands for Management Overlay Network, is a system designed to facilitate the management of large distributed applications and is currently deployed in the PlanetLab testbed.[9] MON builds on-demand overlay structures that are used by users to issue a set of *instant management commands* or distribute software across a large set of nodes. To that end it uses a random overlay network based in *partial views* that is maintained by a *cyclic* approach. It supports the construction of both tree structures and directed acyclic graphs structures.

A tree is always rooted at an external entity (named the MON client). To build the tree, the MON client sends a SESSION message to a nearby MON node. A node that receives a SESSION message for the first time reply with a SESSIONOK and becomes a child node of the SESSION sender. It then sends *k* SESSION messages to random nodes from its partial view. A node that receives a SESSION message for

[9] http://planet-lab.org/

a second time simply sends a PRUNE message to its originator. Hence the tree is constructed using the combination of a sender-based strategy with a gossip strategy, where *k* is the gossip fanout value.

To build a directed acyclic graph, where a node can have more than one parent, MON employs the same algorithm with the following modifications: In order to avoid cycles, each node has a *level* value, where the level at the root is 1 and the level of other nodes is 1 plus the level of their (first) parent. When a node receives a second SESSION message, that also carries the level value of the node who sent it, a node can accept the message, and reply with a SESSIONOK message, if the level value in the SESSION is smaller than its own (therefore, gaining one more parent node).

Because MON is aimed at supporting short-lived interactions, it is not required to maintain these structures for prolonged time. Therefore, it does not owns any repair mechanism to cope with failures. Also, it only gives probabilistic coverage of all nodes, as the gossip strategy used to disseminate the SESSION message only ensures probabilistic atomic broadcast guarantees.

3 The HyParView Protocol

In this section we describe the *Hy*brid *Par*tial *View* membership protocol, or simply *HyParView*[20]. The motivation behind the design of HyParView is based on the following two observations:

- The fanout of a gossip protocol is constrained by the desired reliability and fault-tolerance levels of the protocol. When partial views are used, the quality of these views has an impact on the fanout required to achieve high reliability.
- High failure rates may have a strong impact on the quality of partial views and disrupt the broadcast protocol. Even if the membership protocol has healing properties, the reliability of broadcasted messages after heavy failures may be seriously affected. Therefore, gossip-based broadcast protocols would strongly benefit from membership protocols with fast healing properties.

Leveraging on the observations above, HyParView is based on the two following complementary techniques:

Symmetric partial views: which allow a node to have some control over its in-degree. Notice that, unlike other membership protocols based on partial views, this technique offers two interesting properties: (i) it establishes an upper bound to the in-degree of every node, and (ii) if every node in the system has a full partial view i.e., if the out-degree of all nodes is equal to the size of partial views), then every node in the system is known by exactly the same amount of other nodes (i.e., every node has the same in-degree).

Partial views with a size equal to the fanout $+1$: which provide some determinism on the selection of gossip targets without changing the behavior of gossip protocols. This ensures that, as long as the overlay is connected, and every node has at least one correct node in its partial view, every node in the system will receive every disseminated message, as the dissemination process implicitly becomes a flooding in the overlay.

These techniques allow to use smaller fanout values while ensuring a high reliability. To reduce the time required to recover the accuracy of partial views, HyParView employs two additional techniques:

Unreliable low-cost failure detector: providing a timely detection of failed peers in the partial view. By immediately removing failed peers from the active view, one avoids to select such peers as gossip targets. Moreover, this allows a timely replacement of failed peers by correct ones, improving the overall connectivity of the overlay network.

Maintain a backup partial view: that is used to fill the main partial view (e.g., the partial view used to select gossip targets) when it contains free slots, for instance, after some peers had been detected as failed. This allows to improve the time required to recover the connectivity degree of the overlay network.

HyParView uses TCP as the unreliable low-cost failure detector. TCP is appropriate because, as described previously, we rely in symmetric partial views to establish deterministic communication patterns among peers. The use of TCP actually simplifies the implementation of these symmetric relations. Moreover, as initially noted and proposed in [22], TCP is a network friendly protocol which, unlike UDP, avoids the exhaustion of the network resources. Finally, TCP also allows to use more conservative fanout values, as the gossip protocol is no longer required to mask network omissions.

The use of a second and larger partial view, as a repository of backup nodes, also allows to address the fault-tolerance issues that arise due to the use of a small partial views. Although a smaller partial view increases the probability of a node becoming disconnected from the overlay due to faults, this limitation is circumvented by relying on the second partial view, which increases the probability of a node to contain information about correct nodes and quickly reconnect to the overlay.

Therefore, the HyParView protocol maintains two distinct views at each node. A small symmetric active view of size $fanout+1$ (one link from which a message is received and *fanout* links to which the message is relayed). A larger passive view, that ensures connectivity despite a large number of faults and that must be larger than $log(n)$. Note that the overhead of the passive view is small, as no TCP connections are kept open.

Active views of all nodes create an overlay that is used for message dissemination. Links in the overlay are symmetric: if node q is in the active view of node p then node p is also in the active view of node q. As we have stated before, the architecture of HyParView assumes that nodes use TCP to exchange messages in the overlay. This means that each node keeps an open TCP connection to every other node in its active view. This is feasible because the active view is very small. When a node receives a message for the first time, it forwards the message to all nodes of its active view (except, obviously, to the node that has sent the message). Therefore, the gossip target selection is deterministic in the overlay. However, the overlay itself is created at random, using the gossip membership protocol.

A reactive strategy is used to maintain the active view. Nodes can be added to the active view when they join the system. Also, nodes are removed from the active view when they fail (or leave). Notice that each node tests its entire active view every time it forwards a message. Therefore, the entire overlay is implicitly tested at every broadcast, which allows a very fast failure detection.

HyParView relies in a join mechanism based on fixed size random walks on the overlay. The join mechanism allows to add the identifier of a new node both to the active views and passive views of some random peers. This is achieved by adding the new node identifier: (i) to the passive view of nodes located in a middle point of random walks; and (ii) to the active view of nodes where random walks end.

Additionally, the passive view is maintained using a cyclic strategy. Periodically, each node performs a shuffle operation with one random node in the overlay network, found through a random walk, which will result in the update of both nodes passive views.

One interesting aspect of our shuffle mechanism is that the identifiers exchanged in the shuffle operation are not extracted only from the passive view: a node also sends its own identifier and some nodes collected from its active view to its neighbor. This increases the probability of having nodes that are *active* in the passive views and ensures that failed nodes are eventually expunged from all passive views.

4 Achieving Resilient Broadcast

The HyParView peer sampling service described in the previous section can be used to implement an efficient and resilient broadcast protocol just by relying on a eager push strategy to flood the messages in the overlay network defined by the active views. This algorithm ensures that all links in the overlay are used at least once by leveraging on the semantics of the `getPeer`() call of the HyParView protocol. As discussed before, this ensures that all nodes will receive every broadcast message as long as the membership protocol is able to maintain the overlay connected. The reader should notice that the algorithm can work in the “Infect and Die” model [11], as it only relays messages to other nodes upon the reception of each gossip message for the first time.

This strategy is only viable because HyParView maintains an active view that has a small degree.[10] The degree of the overlay will determine the relative message redundancy of the protocol in stable environment, for instance, if the overlay has a degree of 5, the fanout of the protocol will be 4 (because the overlay has symmetric links), hence the relative message redundancy expected by this gossip strategy in a stable environment will tend to a value of 3.

[10] In fact, as stated before, HyParView was originally designed to support this specific strategy.

The combination of flooding, generating some amount of message redundancy, with the mechanisms of HyParView, allows the protocol to: (i) contrary to other gossip-based approaches, maintain a constant reliability of 100% in the presence of simultaneous node failures as high as 20%; (ii) lower the number of hops required by the gossip-based broadcast protocol, to disseminate a message, to the lowest value allowed by the overlay diameter. This happens because when flooding the overlay, the protocol uses all available links between nodes, implicitly using all overlay shortest paths among peers to disseminate a message (independently of the sender).

Finally, another point that favors this strategy is it simplicity. It is based on a pure eager push gossip approach, hence it does not have to buffer messages it delivers and, as it is topology independent, it does not require the maintenance of complex state related with its neighbors.

5 Building Low Cost Spanning Trees

We now describe the Plumtree protocol[19], a gossip-based dissemination protocol that can exploit the special properties of the unstructured overlay network provided by HyParView, in order to efficiently build and maintain a highly resilient spanning tree, embedded in the original overlay, which allows to improve the cost of the gossip-based dissemination scheme.

The eager push strategy presented in Section 4 allows to obtain a high reliability while ensuring the smallest possible value of last delivery hop. Unfortunately in a stable environment it still produces a significant RMR (relative message redundancy) value.[11] An intuitive approach that could help to further reduce the RMR value is to use a structured overlay that establishes a multicast tree covering all nodes in the system. This is the motivation of this protocol, which was originally described in [19] and named *P*ush-*l*azy-p*u*sh *m*ulticast *tree*, or simply, Plumtree

Plumtree has two main components, each one answers a specific challenge of a fault-tolerance broadcast scheme that employs spanning trees. These can be defined as follows:

Tree construction This component is in charge of selecting which links of the random overlay network will be used to forward the message payload using an eager push strategy. The goal is to provide a tree construction mechanism that is as simple as possible, with minimal overhead in terms of control messages.

Tree repair This component is in charge of repairing the tree when failures occur. The process should ensure that, despite failures, all nodes remain covered by the spanning tree. Therefore,it should be able to detect and heal partitions of the

[11] The reader should notice that the RMR value is still lower than those obtained with other protocols that must use higher values of fanout to ensure a (probabilistic) high level of node coverage.

tree. The overhead imposed by this operation should also be as low as possible. Additionally, this process should recover lost messages, increasing the overall reliability of the broadcast process.

Consider that the eager push strategy presented in Section 4 is used to flood the HyParView overlay from a single source. In stable conditions,[12] it is likely that messages triggering deliveries to the application layer (i.e., messages that are received at a node for the first time) are usually received through the same incoming link (i.e., from the same neighbor). Together, these links form a spanning tree that connects all nodes to a the source node (or root). All other links in the overlay are redundant, and are only required to mask node failures. Therefore redundant links can be *pruned* (removed) from the overlay for as long as no failure exists.

The basic idea behind the Plumtree protocol derives from this simple concept. The operation of Plumtree combines the basic flooding process with a *prune* process. Initially all links in a random (connected) overlay are considered as being part of the broadcast tree. Payload messages are only sent to those links (neighbors). Whenever a redundant (payload) message is received a PRUNE message is used to remove the link, used to transmit that message, from the tree.

Although this explain how tree construction may be performed, it does not address tree repair. Without additional measures, a single node failure is able to partition the spanning tree, disconnecting a large set of nodes from the source. To solve the challenge of repairing the spanning tree, a technique based in a lazy push gossip strategy is employed. This technique enables nodes that do not receive some messages (due to a failure in the tree) to retrieve those messages from other neighbors and trigger the creation of new links to the spanning tree, reconnecting the tree. This procedure enables the whole spanning tree to be repaired. Thus, nodes are required to announce messages they receive through the links of the overlay that are not part of the spanning tree by sending IHAVE messages. Whenever a node requests a message from a neighbor, by sending a GRAFT message, the link between those nodes becomes a new branch of the spanning tree. The interested reader can find more detail on the properties of this mechanism in [19].

Plumtree leverages in the reactive nature of the underlying peer sampling service, which allows to have a stable spanning tree structure in steady state. Moreover, to speed up the recovery process of the spanning tree, whenever a new node is added to the partial view of the underlying peer sampling service, the new link is immediately added to tree. Redundant branches that are created due to this procedure, are removed by the tree construction mechanism upon the dissemination of a broadcast message.

The spanning tree is constructed with a node serving as root, hence it is optimized, at least in terms of last delivery hop (latency), for messages that are sent by that node. But given that the links of the overlay are symmetric, any node can use

[12] Stable conditions in this context concerns not only to no changes in the membership protocol, but also to a network where links have a low variance in behavior, such as latency or bandwidth.

the same (shared) tree structure to broadcast messages, although this may result in sub-optimal routing, in terms of last delivery hop.

In summary, the Plumtree design is based on the following principles:

- It constructs a spanning tree on top of HyParView[13] that is optimized for systems with a single source. Nevertheless, the resulting tree is shared and can be used by any node to broadcast messages.
- The construction of the tree is based on the combination of an eager push algorithm and a pruning process. Because paths in the tree are selected using messages sent by a root node, our tree construction can be classified as a source-based strategy.
- The repair of the tree is based on a lazy push gossip approach. In addition to forwarding the payload through the links that form the spanning tree, nodes also send IHAVE messages on the remaining links. Whenever a node requests a message it has missed from a neighbor, a new branch is added to the tree. Because the repair process is controlled by the receiver, it can be said that the tree repairing uses a receiver-based strategy.

The tree built by Plumtree is optimized for a specific sender: the source of the first broadcast that is used to prune the redundant links from the original overlay in order to form the spanning tree. As a result, the overlay links that form the spanning tree are those that were faster to propagate the message disseminated by that specific node. In a network with multiple senders, Plumtree can be used in two distinct manners:

- For optimal latency, a distinct instance of Plumtree may be used for each different sender. This however, requires an instance of the Plumtree state to be maintained for each sender-based tree, with the associated memory and signaling overhead.
- Alternatively, a single shared Plumtree instance may be used for multiple senders. Clearly, the last delivery hop value may be sub-optimal for all senders except the one whose original broadcast created the tree. On the other hand, a single instance of the Plumtree protocol needs to be executed.

6 Experimental Evaluation

This section starts by evaluating the peer sampling service properties that are relevant for data dissemination. Then, the performance of the eager push and the Plumtree protocols on top of HyParView is assessed.

[13] Although Plumtree was developed to leverage on the properties of HyParView, it is not limited to the use of this peer sampling service.

6.1 Experimental Setting

All simulations described in the section were conducted using the PeerSim Simulator [16]. In order to get comparative figures, both HyParView, Cyclon and Scamp were implemented in this simulator. In order to validate the implementations of Cyclon and Scamp, results obtained with the PeerSim Simulator were compared with published results for these protocols (those results are not presented here, as this is not the main focus of the chapter).

A modified version of Cyclon, named CyclonAcked, is also evaluated. This version adds a failure detection system to Cyclon, based on the exchange of explicitly acknowledgments during message dissemination. Thus, CyclonAcked is able, in less time, to detect failed nodes when it attempts to gossip to them and, therefore, to perform a faster removal of such elements from partial views, increasing the accuracy of the original Cyclon after failures. This benchmark is used to show that the benefits of the HyParView protocol are not only derived from the use of a reliable transport (and also, as an unreliable failure detector), but also from the clever use of two separate partial views.

An eager push gossip broadcast protocol for PeerSim was also implemented. This gossip protocol is able to use any of the membership protocols referred above as a *peer sampling service* (by means of the `GetPeer()` method). It operates in unlimited gossip mode, such that the global reliability is not affected by the configuration of the *maximum rounds* parameter (as shown in Section 2.1).

The eager gossip broadcast protocol was configured so that, when combined with HyParView, it implements the flood gossip strategy described previously, in Section 4. For fairness, the same configuration parameter was used in all membership protocols.

An implementation of the Plumtree protocol was also developed for the PeerSim simulator. This implementation relies in HyParView as the underlying membership protocol, as it was originally designed to leverage on its properties, such as the symmetry and stability of the active view.

In all simulations, the overlay was created by having nodes join the network one by one, without running any membership rounds in between. Cyclon was initiated by having a single node to serve as contact point for all join requests. Scamp was initiated by using a random node already in the overlay as the contact point. These are the configurations that provide the best results for each of these protocols. HyParView achieves similar results with either method. For simplicity, and similarly to the Cyclon protocol, a single node as contact was used to create the HyParView overlay.

All simulations conducted in the PeerSim simulator used its cycle based engine. Each simulation is composed of a sequence of cycles, which begin at cycle 0. Each simulation cycle is composed, at most, of the following steps:

Failure: In this step, some number (or a percentage) of nodes are marked as failed (e.g., their internal state is set to *Down*). This step might not be required for all simulations and it usually only executes at a predefined cycle (or cycles) of a given simulation.

Broadcast: In this step a number of nodes[14] send a broadcast message. The step is only terminated when there are no more messages in transit in the overlay (either gossip messages or any gossip strategy specific control messages).

Data retrieval: In this step, performance information is retrieved by inspecting the internal state of protocols and components of all correct nodes (e.g., nodes which have their internal state set to *Up* in the current simulation cycle).

Membership: In this step, the membership protocol executes any *cyclic* (e.g., periodic) step. It is also in this step that any membership protocol that uses TCP becomes aware of failures that might have happened in the last *failure* step.

Clean up: All data stored on nodes concerning the broadcast of messages or any information concerning the state of the overlay in the current cycle is erased (e.g., temporary state that is maintained at nodes or specific components of the simulation is deleted).

Although most simulations follow the structure above, some of them are conducted without performing all these steps. For instance, when testing the behavior of a protocol in a stable environment (i.e., without the presence of node failures) the failure step is not required. Also, when the goal of the simulation is to evaluate properties of the overlay network, the broadcast step is not required.

6.2 HyParView and Eager Push Strategy

As noted in Section 2.6, the overlays produced by membership protocols, or peer sampling services, should exhibit some relevant properties such as low clustering coefficient, small average shortest path, and balanced in-degree distribution, to contribute for a fast message dissemination and a high level of fault tolerance in gossip-based broadcast protocols. In this section it is shown how the simulated protocols perform regarding these metrics.

	Average clustering coefficient	Average shortest path	Last delivery hop
Cyclon	0.006836	2.60426	10.6
Scamp	0.022476	3.35398	14.1
HyParView	0.000920	6.38542	9.0

Table 1 Graph properties after stabilization

[14] The nodes that send broadcast messages can be selected at random or be predefined.

Table 1 shows values concerning average clustering coefficient, average shortest path, and last delivery hop for all protocols after a period of stabilization of 50 membership cycles.[15] It can be seen that in terms of average clustering coefficient, HyParView achieves significantly lower values than Scamp or Cyclon. This is not surprising given that HyParView's active view is much smaller than other protocols partial views. Nevertheless, this feature is an important factor, which contributes to the high resilience that HyParView exhibits to node failures.

In terms of average shortest path, it is clear that HyParView falls behind Scamp and Cyclon. This is no surprise, as HyParView maintain a much smaller active view, which limits the number of distinct paths that exist across all nodes. Fortunately, this has no impact on the latency of the gossip protocol. The small level of global clustering, and the fact that all existing paths between nodes are used to disseminate every message, makes a HyParView based gossip protocol to deliver messages with a smaller number of hops than the other protocols, as can also be seen in Table 1.

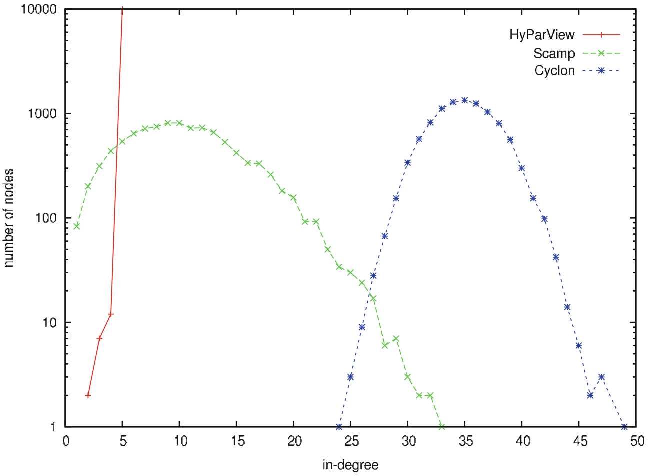


Fig. 1 In-degree distribution

Figure 1 shows the in-degree distribution of all nodes in the overlay after the same stabilization period. Cyclon and Scamp have an in-degree distribution across a wide range of values, which means that some nodes are extremely popular on the overlay, while other nodes are almost totally unknown. As stated before, because of this distribution, some nodes on the overlay have larger probability of receiving redundant messages, while other nodes have a very small probability of receiving messages even once. Naturally, nodes which are known by only a few other peers,

[15] In fact, this stabilization time is not required by Scamp, as it stabilizes immediately after the join period. HyParViews active view also stabilizes within 1 – 2 simulation cycles, but its passive view requires some rounds of membership in order to stabilize completely.

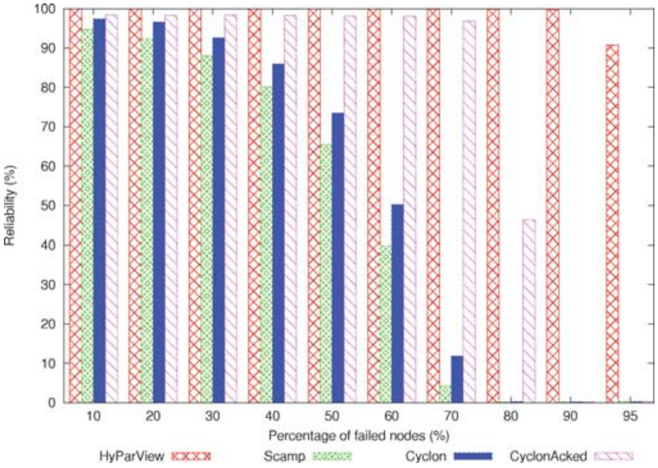


Fig. 2 Average reliability for 1000 messages

have a smaller probability to be selected as gossip targets. Also, such nodes have an increased probability to become disconnected from the overlay, as the number of nodes that are required to fail in order to disconnect the node is smaller. This is particularly obvious in Scamp, where some nodes are only known by one other node.

Due to HyParViews symmetric active view, almost all nodes in the overlay are known by the maximum amount of peers possible, which is the active view size (5). This means that all nodes, with high probability, will receive each message exactly the same amount of times. Also, there is little probability for any node not to receive a message at least once. Finally, notice that nodes who have the smallest in-degree have at least 2 neighbors,[16] and that the number of nodes in these conditions is marginal (with only 2 in 10.000 nodes).

We now evaluate the impact of massive failures in the reliability of the eager push gossip broadcast, when combined with different membership protocols.

In each experiment, all nodes join the overlay after which they execute 50 cycles of membership protocol to guarantee stabilization. After the stabilization period, failures at random are induced in a percentage of all nodes in the system, ranging from 10 to 95% of node failure. A measure of the reliability of 1.000 messages sent from random correct nodes was then extracted. All these messages were sent before the execution of any cycle of a membership protocols. However, the membership protocols still execute all reactive steps; in particular, they can exclude a node from their partial views if the node is detected to be failed. The rationale for this setting is that the interval of the cyclic behavior of the membership protocols is often long enough to allow thousands of messages to be exchanged; the goal here is to focus on the impact of failures in the reliability of these specific broadcasts.

[16] Obviously, this also means that such nodes have an out-degree of 2.

The average reliability for these runs is depicted in Fig. 2. As it can be seen, massive percentage of failures have almost no visible impact on HyParView below the threshold of 90%. Even for failure rates as high as 95%, HyParView still manages to maintain a reliability value in the order of 90%. Both Scamp and Cyclon exhibit a constant reliability[17] for failure percentages as low as 10%, and their performance is significantly hampered with failure percentages above 40% (with reliability values below 50%). On the other hand, CyclonAcked manages to offer a competitive performance. Although the reliability is not as high as with HyParView, it manages to keep high reliabilities for percentage of failures up to 70%. This behavior highlights the importance of fast failure detection in gossip protocols and shows the benefits that come from the use of TCP as an unreliable failure detector.

6.3 Plumtree

In this section, the Plumtree protocol is evaluated. In order to extract comparative figures, experimental results are shown together with those obtained by the eager push strategy and a pull based gossip strategy in the same scenarios. The Plumtree protocol performs better in scenarios with only one (fixed) sender. Nevertheless, the protocol was evaluated operating in two distinct modes:

Single sender (labeled as *s–s*) mode in which a single node is the source of all broadcast messages.

Multiple senders (labeled as *m–s*) mode in which each broadcast message is sent by a random (correct) node.

To evaluate the protocol in a stable environment, several simulations were conducted in the following manner: First all nodes join the overlay by using the HyParView join mechanism. As in the evaluation of the HyParView protocol, this is achieved by using one single node that serves as *contact* for all remaining nodes. No membership cycles are executed during the join process. Simulations are run for a total of 250 simulation cycles. The first 50 cycles of each run are used to ensure stabilization and hence, are not depicted in the results. In each simulation cycle, in the broadcast step, a single node sends a message. The reliability, last delivery hop, and relative message redundancy of the broadcast protocols are then evaluated for each message disseminated.

Simulations show that all the gossip-based broadcast protocols were able to achieve and maintain a reliability of 100% in stable conditions. This includes Plumtree operating in the two modes described above. This is expected, given the properties of the HyParView protocol, namely from the connectivity point of view: HyParView ensures the connectivity of the overlay, and all approaches (eager push and pull based strategies along with the tree based strategy) ensure that all nodes in the overlay receive each broadcast message as long as the overlay remains

[17] Although their reliability is unable to reach 100% with a fanout of 4.

connected. Still, measurement of reliability in stable conditions serves as a validation for the design of the Plumtree protocol.

The relative message redundancy (RMR) value (as defined in Section 2.7) is of paramount importance when evaluating the Plumtree protocol, as this is the main metric aimed for optimization by this approach. Table 2 shows aggregated values of relative message redundancy obtained in our experiments for all gossip protocols. All protocols, in stable conditions, were able to maintain a constant value for RMR in the last 200 cycles of every simulation. Notice that, as expected, the eager protocol has a relative message redundancy close to 3. This derives directly from the fanout value used (4), as explained earlier in the chapter.

	RMR
Eager	3.00
Pull	0.00
Plumtree (s-s)	0.00
Plumtree (m-s)	0.00

Table 2 Relative message redundancy in stable environment

In Plumtree, after the stabilization of the protocol, payload messages are only propagated through links that belong to the spanning tree. The Plumtree algorithm eliminates redundant branches from the tree. Therefore, Plumtree does not generate any redundant message. As a result, the RMR value for the Plumtree protocol presents a constant value of 0.

The pull approach is also able to maintain a constant value of 0 for RMR. This is expected, as in a pull approach payload is only transmitted when explicitly requested by the receiver. Therefore, in a stable environment, a single payload message has to be transfered for each receiver. However, such an approach as some drawbacks as we show in the following paragraphs.

Table 3 shows the number of messages received by all gossip protocols after 150 cycles of stabilization. The extra control messages received by Plumtree are essentially due to IHave messages. However, the reader should consider that: (i) usually IHave messages are smaller than payload message, hence these messages will contribute less to the exhaustion of network resources and (ii) IHave messages may be aggregated, by delaying the transmission of these messages and sending several broadcast message unique identifiers on a single IHave message. Notice that aggregation will not have a significant impact in reliability, as messages are still sent, only with a delay. Therefore, aggregation would only affect the time required to repair the spanning tree after failures and the overall latency of the dissemination process.

	Payload	Control	Total
Eager	39984.00	0.00	39984.00
Pull	9999.00	49994.33	59993.33
Plumtree (s-s)	9999.00	29987.33	39986.33
Plumtree (m-s)	9999.00	29990.00	39989.00

Table 3 Number of messages received

Notice that the pull approach requires much more control messages than Plumtree. This is expected as the transmission of each payload message requires two distinct control messages. Additionally, several control messages are sent in the dissemination process which are redundant, similar to the payload messages sent by the eager push strategy.

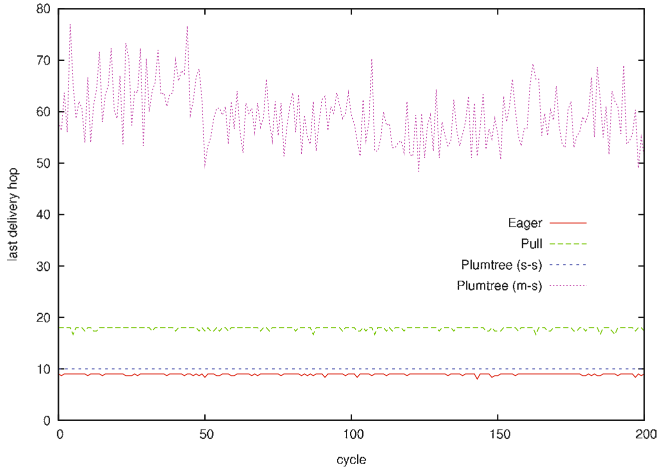


Fig. 3 Last delivery hop in stable environment

Figure 3 presents the values for last delivery hop (LDH). The eager push protocol and Plumtree with a single sender offer the best performance. The eager protocol uses all available links to disseminate messages, which ensures that all shortest paths of the overlay are used. This shows that with a single sender Plumtree is able to select the links that provide faster delivery.

The pull approach doubles the value of LDH when compared with Plumtree and the eager protocol.[18] This happens because this strategy imposes additional delay

[18] Notice that in these experiments we ignore the hop executed by the message explicitly requesting the transmission of payload.

to the dissemination of payload messages in the overlay. This will have a negative impact in the latency of the broadcast process.

With multiple senders the Plumtree LDH values are naturally higher. This happens because the spanning tree is optimized for the first source. Therefore, when messages are sent by nodes located at leaf positions of the tree, messages are required to perform additional hops in the overlay to reach all remaining nodes.

The interested reader may refer to [19], where an optimization to the Plumtree protocol described here is presented and evaluated. This optimization is able to lower the LDH value when the protocol operates in multiple senders mode. Moreover, the optimized protocol achieves a better distribution of the spanning tree branches across all nodes in the system.

7 Discussion and Future Directions

In this chapter we introduced some of the most relevant aspects of gossip-based broadcast protocols. Namely, we have described the basic principles of epidemic protocols, their typical parameters and operation modes. We have also described the main gossip strategies that can be found in literature, such as: eager push, lazy push, and pull; and discussed the associated trade-offs associated with the use of these techniques. To help the reader in assessing the behavior of different gossip-based broadcast solutions, we listed several metrics to evaluate the quality of the unstructured overlay networks created by peer sampling services, discussed their desirable values, and explained how they affect the epidemic dissemination process. We also offer a survey of some representative protocols, including gossip-based membership protocols, gossip-based broadcast protocols, and for completeness, protocols based on the construction of spanning trees. Finally, we described and evaluated a family of concrete peer sampling and gossip-based broadcast protocols.

As future directions for research in this field, we consider that the body of work presented in this chapter can be improved in two distinct aspects:

- The algorithms presented in this chapter may be improved by considering the (possible) heterogeneity of nodes in the system. An obvious approach to achieve this is to use an adaptive fanout (and degree) for nodes. This value should be based on a function of the resources available at each node. By using such strategy, a node which is more powerful (e.g., which has more memory, or more bandwidth) would contribute more, not only to the maintenance of the unstructured overlay network, but also to the dissemination process (e.g., by sending more messages).
- Plumtree-like protocols could benefit from techniques to improved load distribution. For instance, when the current protocol operates in single sender mode, nodes which are located at the leaf positions of the spanning tree, do not contribute (actively) to the dissemination of messages payloads.

Acknowledgments

This work was partially supported by project “Redico” (PTDC/EIA/71752/2006).

References

1. Birman, K., Hayden, M., Ozkasap, O., Xiao, Z., Budiu, M., Minsky, Y.: Bimodal multicast. ACM Transactions on Computer Systems **17**(2) (1999)
2. Carvalho, N., Pereira, J., Oliveira, R., Rodrigues, L.: Emergent structure in unstructured epidemic multicast. In: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, p. (to appear). Edinburgh, UK (2007)
3. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.: SCRIBE: A large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in communications (JSAC) **20**(8), 1489–1499 (2002). URL `citeseer.ist.psu.edu/castro02scribe.html`
4. Chu, Y.H., Rao, S., Seshan, S., Zhang, H.: A case for end system multicast. IEEE Journal on Selected Areas in Communications **20**(8), 1456–1471 (2002)
5. Deering, S., Cheriton, D.: Multicast routing in datagram internetworks and extended lans. ACM Transactions on Computer Systems **8**(2), 85–110 (1990)
6. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., Terry, D.: Epidemic algorithms for replicated database maintenance. In: PODC ’87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing, pp. 1–12. ACM Press, New York, NY, USA (1987). DOI http://doi.acm.org/10.1145/41840.41841
7. Deshpande, M., Xing, B., Lazardis, I., Hore, B., Venkatasubramanian, N., Mehrotra, S.: Crew: A gossip-based flash-dissemination system. Tech. rep., University of California (2005). URL `http://www.ics.uci.edu/~mayur/crew_ics_tr_2005.pdf`
8. Deshpande, M., Xing, B., Lazardis, I., Hore, B., Venkatasubramanian, N., Mehrotra, S.: Crew: A gossip-based flash-dissemination system. In: ICDCS ’06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, p. 45. IEEE Computer Society, Washington, DC, USA (2006). DOI http://dx.doi.org/10.1109/ICDCS.2006.24
9. Diot, C., Levine, B., Lyles, B., Kassem, H., Balensiefen, D.: Deployment issues for the IP multicast service and architecture. IEEE Network **14**(1), 78–88 (2000). URL `citeseer.ist.psu.edu/diot00deployment.html`
10. Eugster, P., Guerraoui, R., Handurukande, S., Kouznetsov, P., Kermarrec, A.M.: Lightweight probabilistic broadcast. ACM Transactions on Computer Systems **21**(4), 341–374 (2003). DOI http://doi.acm.org/10.1145/945506.945507
11. Eugster, P., Guerraoui, R., Kermarrec, A.M., Massoulie, L.: From Epidemics to Distributed Computing. IEEE Computer **37**(5), 60–67 (2004). DOI NA
12. Ganesh, A., Kermarrec, A.M., L. Massouli e.: SCAMP: Peer-to-peer lightweight membership service for large-scale group communication. In: Networked Group Communication, pp. 44–55 (2001). URL `citeseer.ist.psu.edu/ganesh01scamp.html`
13. Ganesh, A., Kermarrec, A.M., Massoulié, L.: Peer-to-peer membership management for gossip-based protocols. IEEE Transactions on Computers **52**(2), 139–149 (2003). DOI http://dx.doi.org/10.1109/TC.2003.1176982
14. Hayden, M., Birman, K.: Probabilistic broadcast. Tech. rep., Cornell University, Ithaca, NY, USA (1996)
15. Jelasity, M., Guerraoui, R., Kermarrec, A.M., van Steen, M.: The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In: Middleware ’04:

Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, pp. 79–98. Springer-Verlag New York, Inc., New York, NY, USA (2004)

16. Jelasity, M., Montresor, A., Jesi, G.P., Voulgaris, S.: The Peersim simulator. http://peersim.sf.net
17. Kermarrec, A.M., Massoulié, L., Ganesh, A.: Probabilistic reliable dissemination in large-scale systems. IEEE Transactions on Parallel and Distributed Systems **14**(3), 248–258 (2003). DOI http://dx.doi.org/10.1109/TPDS.2003.1189583
18. Koldehofe, B.: Buffer management in probabilistic peer-to-peer communication protocols. In: Proceedings of the 22th IEEE Symposium on Reliable Distributed Systems (SRDS'03), pp. 76–87. Florence,Italy (2003)
19. Leito, J., Pereira, J., Rodrigues, L.: Epidemic broadcast trees. In: Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems (SRDS'2007), pp. 301 – 310. Beijing, China (2007)
20. Leito, J., Pereira, J., Rodrigues, L.: HyParView: A membership protocol for reliable gossip-based broadcast. In: DSN '07: Proc. of the 37th Annual IEEE/IFIP Intl. Conf. on Dependable Systems and Networks, pp. 419–429. IEEE Computer Society, Edinburgh, UK (2007). DOI http://dx.doi.org/10.1109/DSN.2007.56
21. Liang, J., Ko, S., Gupta, I., Nahrstedt, K.: MON: On-demand overlays for distributed system management. In: 2nd USENIX Workshop on Real, Large Distributed Systems (WORLDS'05) (2005). URL http://cairo.cs.uiuc.edu/publications/paper-files/worlds05-jinliang.pdf
22. Pereira, J., Rodrigues, L., Monteiro, M.J., Oliveira, R., Kermarrec, A.M.: Neem: Network-friendly epidemic multicast. In: Proceedings of the 22th IEEE Symposium on Reliable Distributed Systems (SRDS'03), pp. 15–24. Florence,Italy (2003)
23. Pereira, J., Rodrigues, L., Pinto, A., Oliveira, R.: Low-latency probabilistic broadcast in wide area networks. In: Proceedings of the 23th IEEE Symposium on Reliable Distributed Systems (SRDS'04), pp. 299–308. Florianopolis, Brazil (2004)
24. van Renesse, R., Birman, K.P., Vogels, W.: Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. ACM Transactions on Computer Systems **21**(2), 164–206 (2003). DOI http://doi.acm.org/10.1145/762483.762485
25. van Renesse, R., Minsky, Y., Hayden, M.: A gossip-style failure detection service. Tech. Rep. TR98-1687, Dept. of Computer Science, Cornell University (1998). URL citeseer.ist.psu.edu/article/vanrenesse96gossipstyle.html
26. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, pp. 329–350. Springer-Verlag, London, UK (2001)
27. Rowstron, A., Kermarrec, A.M., Castro, M., Druschel, P.: SCRIBE: The design of a large-scale event notification infrastructure. In: Networked Group Communication, pp. 30–43 (2001). URL citeseer.ist.psu.edu/rowstron01scribe.html
28. Voulgaris, S., Gavidia, D., van Steen, M.: Cyclon: Inexpensive membership management for unstructured p2p overlays. Journal of Network and Systems Management **13**(2), 197–217 (2005). DOI 10.1007/s10922-005-4441-x. URL http://dx.doi.org/10.1007/s10922-005-4441-x
29. Zhao, B., Kubiatowicz, J., Joseph, A.: Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. UCB/CSD-01-1141, UC Berkeley (2001). URL citeseer.ist.psu.edu/zhao01tapestry.html

Employing Multicast in P2P Overlay Networks

Mario Kolberg

1 Introduction

The work on multicast has evolved from bottom IP layer multicast to Application Layer Multicast. While there are issues with the dep-loyment of IP layer multicast, it outperforms Application Layer Multicast. However, the latter has the advantage of an easier dep-loyment. Furthermore, as will be illustrated later in this Chapter, IP layer multicast has the potential to make parallel overlay operations more efficient. Application Layer Multicast is primarily used to send application specific messages/data to a number of nodes.

1.1 IP Layer Multicast

In terms of IP multicast there are two major groups of approaches: Host group multicast [15] and multi-destination multicast [6]. Host-group multicast approaches create a group address per multicast tree. Routers in the network then store state for each active group address. Clearly, this information grows with the number of simultaneous active multicast groups. Additionally, there is delay to create a group, and the network may only support a limited number of group ad-dresses [8].

Consequently, host group multicast is best suited for relatively small numbers of groups with potentially very large sets of recipients.

In multi-destination multicast routing, messages are sent containing the list of all the destinations for that message. Multicast-enabled routers inspect all addresses and make a routing decision for each address. If all addresses will be routed the same path, the message stays together, if there are different paths for addresses, the message gets split and forwarded separately. If a message finally only con-tains a single

Mario Kolberg
Department of Computing Science and Mathematics, University of Stirling, Stirling, Scotland,
e-mail: mko@cs.stir.ac.uk

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_30, © Springer Science+Business Media, LLC 2010

address, it is converted to a unicast message. Multi-destination routing restricts the group size to about 50 destinations.

The quantitative benefits of IP multicasting have been formulated in the Chuang-Sirbu Scaling Law which shows that the ratio of the number of edges in the multicast versus the average unicast path length equals $m^{0.8}$ (m being the multicast group size). The per link reduction in message traffic from multicast is hence $1-m^{-0.2}$.

1.2 Application Layer Multicast (ALM)

Application Layer Multicast takes the basic multicast approach to the application layer and implements multicasting functionality as an application service rather than a network service. Hence the aims and strengths of ALM are rather different to IP layer multicasting.

IP multicasting requires the implementation of the multicasting algo-rithms in the network and network nodes. The primary goal of IP layer multicasting is the reduction of multiple copies of a packet on the same network link. ALM is implemented by application nodes and hence does not necessarily result in a reduction of multiple copies of packet on links. However, ALM remedies one of the key issues with IP multicasting: easier and possibly immediate deployment over the Wide Area Network. Three major approaches have been devised for Application Layer Multicast: Mesh first, Tree first and Implicit approaches. The Mesh first and Tree first approaches commonly are deployed on unstructured P2P overlays, whereas the Implicit approach builds multicast forwarding trees on top of the Distributed Hash Table of structured overlays. Unstructured approaches either organize nodes into a mesh and construct the multicast tree on top of this mesh (Mesh first approaches) or construct the multicast tree directly (Tree first approaches). Some of the latter approaches complement the tree with additional links (mesh).

More recently considerable work has been carried out on topology aware overlays aligning the overlay with the underlying IP network topology and hence making the overlay routing (and multicasting) more efficient. Figure 1 provides an illustration of sending a message to multiple destination via (a) unicast, (b) IP multicast and (c) Application Layer Multicast.

The remaining of the chapter is organized as follows. Section 2 dis-cusses the two IP layer multicast approaches, including the Chuang-Sirbu Scaling Law estimating the efficiency gain through IP layer multicast. Section 3 concentrates on Application Layer Multicast illustrating the three approaches. Section 4 concludes the Chapter.

2 Using IP Multicasting for Parallel P2P Overlay operations

Besides using IP multicast for distribution of media, an important use for IP layer multicast in P2P systems is to make parallel opera-tions in structured P2P overlays

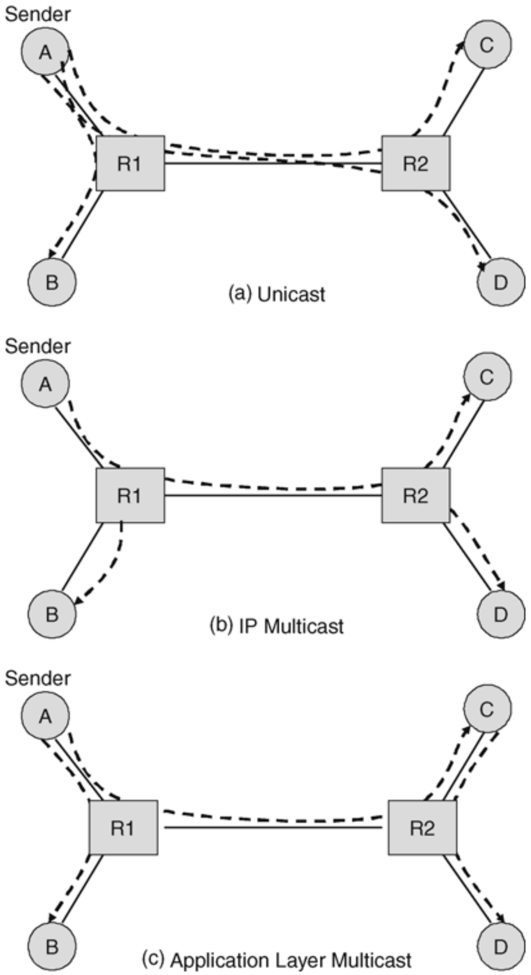


Fig. 1 Sending a packet to three nodes

more efficient [8]. Many structured overlays use parallel lookup messages (e.g., EpiChord [23], Kademlia [24]). Other overlays use also parallel messages for maintenance operations. EDRA in D1HT [25] for instance uses (log n)-way mes-sages, where n is the number of nodes in the overlay. Hence group size is typically relatively small, but group formation can be very frequent. For instance in an EpiChord network, group formation may be as high as one group per second per node. Moreover, group reuse is low, especially in high churn overlays. Here the focus is given to the second use. The following sections examine how suitable the two main IP multicast approaches, Host Group Multicasting and Multi-destination Routing, are for this purpose.

2.1 Host Group Multicasting

The prevailing host-group multicast protocols including the PIM [27] family (PIM-DM, PIM-SM, PIM-SSM and BIDR-PIM), DVMRP [33], and (to a lesser degree) CBT [4] create a group address per multicast tree, and each router stores state for each active group address. Consequently, the state in the router grows with the number of simultaneous multicast groups. Further, there is delay to create a group, and the network may support only a limited number of group addresses. For a large overlay it is impractical for each node to have a group address for each set of other nodes it sends multicast messages to. For example, if there are one million nodes in the overlay, and each peer conservatively uses 5 groups for those messages it parallelizes, then the number of groups might reach five times the number of nodes in the overlay. Worsening the problem is that the average peer session time is as low as 60 minutes in some overlays, meaning that group state needs to be replaced relatively frequently. Consequently, host group multicast is designed for relatively small numbers of very large sets of recipients.

Another issue with the existing approaches is that they are not de-signed for inter-domain multicasting. Essentially, current approaches are intra-domain solutions. However, P2P overlays are typically global networks spanning many different domains and hence mes-sages traveling between nodes may cross domains.

In summary, host group multicast is not a good choice for use in pa-rallelizing overlay network operations with many simultaneous small groups of peers involved in a message, the groups are short-lived, and span multiple domains.

2.2 Multi-Destination Multicasting

Originally, the concept of multi-destination routing was proposed in the early years of multicast protocol design [1]. Initially, it was not favored as Ammar observes [2], as subsequent protocol design fo-cused on enabling large multicast groups. However in the past sev-eral years, there has been recognition of multi-destination routing as a complementary multicast technology that is highly scalable with respect to the possible numbers of groups.

Considering that structured P2P overlays require a large number of relatively small groups which are also rather short-lived multi-destination routing has considerable advantages. Furthermore, it does not pose a bottleneck for overlay networks even with millions of nodes.

In multi-destination multicast routing, instead of two or more unicast messages sent to separate destinations, a single message is sent con-taining the list of all the destinations and the message content from the original messages. Multicast-enabled routers route the message by checking all addresses in the message. If all addresses are routed the same path, the message stays together. However, if addresses are to be routed differently, a split point is reached. At each such point, duplicate messages containing the subset of destinations for each forwarding path are created and routed. This continues until a mes-sage contains only a single address in which case it is converted to a unicast message and is routed to its destination. Figure 2 illustrates this. Here node A is sending a multi-destination multicast message to 4 nodes B, C, D and E. Routers where a split occurs are shown by filled circles. The numbers indicate the count of destination ad-dresses in the message.

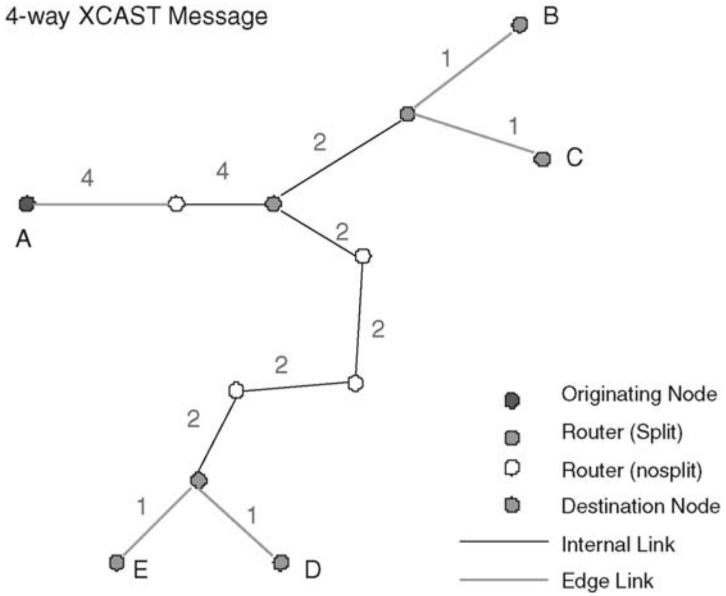


Fig. 2 Sending a 4-way multi-destination multicast message through a network

Clearly the network routers need to be multi-destination enabled to be able to check multiple addresses in a message. However, the approach can also work with only a subset of routers being able to understand multi-destination messages. In that case, these routers forward multicast packets to other multicast routers using tunnels through unicast routers.

In multi-destination multicasting, the sender has to be aware of all destinations. In host group multicasting, this is managed by the rou-ters rather than the sending node. However, multi-destination multi-cast routing does not require state in routers. Thus there is no router state constraint on the number of multicast groups. However there is additional processing overhead at each router for the forwarding al-gorithm to process the list of addresses. Whereas host-group multi-cast routers have forwarding state for each group address, for a mul-ti-destination packet with N destinations, there is O(N) work at each router to process the list of addresses and make a forwarding deci-sion for each destination. Multi-destination routers need to re-build packets at branching points.

As the destination addresses are included in the message, multi-destination multicast imposes a maximum group size. Practical group sizes appear to be those with less than 50 members. Since peers in the overlay maintain routing tables or addresses of other peers, there is no group join overhead when peers are directly reachable in the overlay. Thus the time to create a new multicast group is not a factor.

Explicit multicast (XCAST) is an experimental IP protocol for multi-destination multicast [6] and several XCAST testbeds have been deployed.

There is also work combining host group multicast and multi-destination multicast. He and Ammar [18] analyze the performance of XCAST combined with host-group multicasting.

2.3 Chuang-Sirbu Scaling Law

An estimation of the reduction in network traffic when using multi-cast rather than unicast has been captured in the Chuang-Sirbu [14] scaling law. This shows that the ratio of the number of edges in the multicast path versus the average unicast path length equals m0.8 (where m is the multicast group size). The per link reduction in mes-sage traffic from multicast is then $1 - m^{-0.2}$. The assumptions of the Chuang-Sirbu law are:

1. Receivers are randomly distributed throughout the network.
2. Multicast trees are shortest-path trees constructed using Dijkstra's algorithm and extend only to the edge router.
3. There is little or no control overhead.
4. Networks that are representative of inter-domain routing topolo-gies of the Internet, with average node degree between 3 and 4, and a size between tens and thousands of routers.
5. The multicast routing at the network layer does not support any reliability mechanism.

The first assumption relates to the placement of peers in the network and the distribution of the set of peers in a parallelized overlay operation. In structured overlays, it is generally assumed that peers are randomly distributed throughout the network. Except for topology-aware overlays, neighbors in the overlay are not close to each other in the network and are unlikely to cluster on the same subnet. As a result, parallel overlay operations which are directed at a set of adjacent overlay nodes, such as a parallel lookups, are sent to a random set of network locations.

In the case of topology-aware overlays, the multicast savings can be no worse than predicted by Chuang-Sirbu and may be better due to the sharing of the tree path. The remaining assumptions relate to the network and the multicast mechanism. Using multi-destination routing satisfies assumptions 2, 3, and 5. Structured P2P overlays are typically used for larger scale networks which satisfy assumption 4.

Further evaluation of Chuang-Sirbu has been done in [26] which derives another similar expression and confirms it with respect to various networks, and [32] which

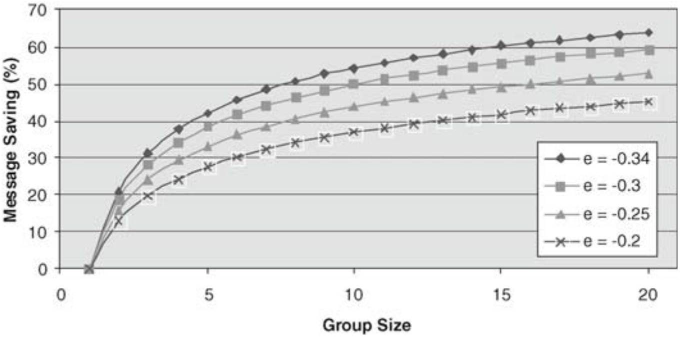


Fig. 3 Message savings for different multicast group sizes and varying multicast efficiency ε according to Chuang-Sirbu law

finds some shortcomings of Chuang-Sirbu with respect to large groups and provides a revised formulation. Chalmers and Almeroth [12] using actual multicast data sets on the Internet and synthesized multicast trees find a slightly lower multicast efficiency ε in the range $-0.34 < \varepsilon < -0.30$. These results are based on an actual multicast infrastructure in place at the time of the data collection which may constrain multicast branching points more so than in synthetic topologies. Further, their analysis includes multicast trees extended to the end points, which produces increased savings from Chuang-Sirbu if there is clustering of end points at subnets.

In the case of parallelized overlay operations, the size of the multi-cast group is within the Chuang-Sirbu formulation but is frequently at the lower end of the formulation, $m < 20$. In this range, the multi-cast efficiency exponent derived by Chalmers and Almeroth in [12] is also applicable. For brevity, here we refer to multicast savings $1 - m^{\varepsilon}$ with $-0.34 < \varepsilon < -0.20$ as the Chuang-Sirbu law. Figure 3 shows message savings versus group size for a set of values of ε.

2.4 Strengths and Weaknesses

A significant percentage of Internet traffic is due to Peer-to-Peer ap-plications. Multicast has great potential to reduce the number of dup-licate messages on network links, and hence the load on the network.

Above it has been discussed that for parallelizing structured overlay messages, multi-destination routing is a better fit than host group multicasting. However, to get the full benefit of multicast at the IP layer, multicast enabled routers need to be installed throughout the Internet, including backbone and edge routers. Clearly this means a substantial investment for the network operators. Furthermore, multicast enabled routers are more complex and hence do not reach the extremely high performance of their simpler unicast counterparts [3]. Host group multicast routers need to keep state for each group, whereas for multi-destination multicast, multiple addresses are checked for each packet slowing packet processing down. Moreover, there are a number of security and management issues still to be solved for multicast [16]. For instance, flooding attacks are easier with multicast routers. Furthermore, there is no standard billing model for multicast traffic accepted yet [22]. Naturally, these issues are easier to manage in an Intranet environment, a reason why multicast is easier deployed in such an environment.

3 Application Layer Multicast

IP multicast has great potential achieving considerable message sav-ings. However, success of IP multicast depends on the deployment of multicast enabled routers. As illustrated earlier, this deployment is rather slow and hindered by a number of issues.

To be able to benefit from some of the multicast advantages, consi-derable work has been undertaken implementing multicasting at the application layer. Like IP multicast is implemented in network nodes (routers), Application Layer Multicast is implemented by application nodes.

Application Layer Multicast [5, 20, 21] has been implemented for unstructured overlays and also for structured overlays. Approaches for unstructured overlays can be further categorized according to the routing organization.

In terms of routing table organization two approaches are distin-guished. Firstly, those approaches that build a richly connected graph (mesh) first and then construct the multicast tree using the graph are termed mesh-first approaches. The second group of ap-proaches, which create the distribution tree first and then subse-quently add additional links for control, are called tree-first ap-proaches. Sometimes Application Layer Multicast approaches which work on a structured overlay are termed implicit approaches, the structured overlay implicitly defines the mesh and the tree.

Key to the success of Application Layer Multicast approaches on P2P overlays is the efficiency of the approach. Best results have been achieved with approaches that consider the underlying topolo-gy, i.e., nodes which are close in the overlay are also close in the un-derlying IP network. This is commonly done with approaches for unstructured overlays, but more difficult with structured overlay approaches which impose a predefined overlay topology.

Performance of Application Layer Multicast approaches is typically judged by the Stretch (ratio of a one-way overlay delay between a pair of nodes over the delay between the equivalent underlay nodes), Stress (number of times a packet traverses an underlay link), and Control Overhead (number of control messages introduced by the Application Layer Multicast approach).

In the following the three categories mesh-first, tree-first and impli-cit approaches are further discussed by given more details on one or two particular approaches each.

3.1 Mesh First Approaches (NARADA)

NARADA [13] was one of the first Application Layer Multicast protocols. Like all Application Layer Multicast protocols, NARADA defines a root node, the Rendezvous Point (RP). As this is a MESH first protocol, it is based on all participating nodes being member of the mesh. If a node wants to join the mesh, it contacts the RP to get a list of all other nodes in the mesh. The RP keeps track of all nodes in the multicast group. Once the new node has the list of mesh mem-bers, it randomly selects some of these nodes and contacts them with a Join-request. If one of these nodes responds positively, the request has been successful. Otherwise, a different set of nodes will be tried.

Once the new node has joined the mesh, it will start exchanging re-fresh messages with its neighbors. Join (and leave) messages will be propagated to all members of the mesh. Hence each member is aware of new nodes joining and leaving. Refresh messages are sent periodically, so neighbors notice if a node vanishes ungracefully. Distribution of the node states to all other nodes in the mesh comes at a high maintenance cost: $O(N^2)$ with N being the number of nodes in the mesh. Hence NARADA does not scale well to large group sizes, however, it balances this with greater robustness.

In order to setup the multicast tree, all members of the mesh run a routing protocol to work out unicast paths between all pairs of nodes. If a node wants to send a multicast message, the multicast data tree can be computed using the Reverse Path Forwarding algorithm.

One important aspect of NARADA is its ability to refine the mesh. The quality of a multicast tree is directly dependant to the quality of the links that form the mesh. However, than new members join or nodes leave the mesh, random links are added to the mesh (a new node joining contacts a random set of nodes of the list from the RP). Hence NARADA nodes make periodic refinements to their links. These refinement decisions are made by the pair of nodes involved in the link locally through simple heuristics. Importantly, the RP is not involved in this decision.

3.2 Tree First Approaches

Yoid [17] is also one of the first Application Layer Multicast proto-cols but unlike NARADA creates the multicast tree directly without creating a mesh first. Clearly this gives direct control on some cha-racteristics of the trees and it is not dependant on the quality of the mesh. HMTP (Host Multicast Tree Protocol) [34] is a protocol rather similar to Yoid, with just slight variations.

Each node wishing join a tree is responsible for finding its parent in the tree. In doing so a node which wants to join the tree contacts the RP first which responds with a list of nodes which are already part of the tree. The new node then contacts nodes from this list to find potential parents. A node can become a parent if this does not cause a loop in the tree and its maximum number of children has not been reached yet. If a number of parents have been identified, the new node chooses one of the nodes according to some criteria of interest.

The main difference of HMTP to Yoid is that HMTP sends a join request to the RP. If the RP has already reached its maximum num-ber of children, it will return its list of children. HMTP will then query these nodes to find the one nearest to itself. If this one also has reached its maximum number of children, the node joining will query these nodes and again find the one closest to itself. Once it finds a node which can add the new node as a child, the new node has joined the tree. Thus HMTP adds the element of "closeness" to the selection of the nodes. In both protocols, nodes will periodically attempt to find better (i.e., closer to itself for HMTP) parents.

In terms of creating a mesh after tree construction, Yoid finds other nodes on the tree at random, which are not its neighbors. Tree links together with these additional links form the mesh and are used for tree partition recovery. HMTP does not create an explicit mesh. Each node simply periodically discovers and caches information about some other members.

3.3 Implicit Approaches

There are a number of ALM schemes which have been implemented on top of structured P2P overlays. Here the multicasting is per-formed by one- or many-to-many communication across the nodes in the overlay.

Major examples of this approach are CAN multicast [28], Scribe [11] on Pastry [30] as well as Bayeux [36] on Tapestry [35]. Here these two approaches employ fundamentally different techniques: CAN uses a flooding based replication approach whereas Bayeux and Scribe form a tree to distribute messages to group members. Flood-ing and tree based approaches are discussed in the next two chapters in more details.

It is important to note that implicit multicasting approaches are de-pendant on the underlying overlay which usually has not been spe-cifically designed with multicast in mind. Hence the structure is of-ten not optimal for multicast (with the exception of topology aware overlays). To improve on the multicast routing three approaches have been developed: proximity routing, topology-based node Id as-signment and proximity neighbor selection [10, 29]. These con-cepts are briefly discussed below.

When using proximity routing the overlay is constructed as usual, without regard for the underlying topology. However, when routing a message, not the entry in the routing whose key is closest to the messages key is necessarily selected but the entry in the routing ta-ble which is physically closest to the current node. This approach can be extended slightly to select an entry from the routing table which is a good compromise between progress in terms of the overlay id space and physical closeness. While this approach might increase the number of hops to the destination, it can substantially decrease the delay for each hop. Clearly the larger the routing tables for each node, the more choice there is. The improvements are proportional to the size of the routing tables. However, increased routing tables come at the cost of increased maintenance traffic. Hence while this approach can provide significant improvements its cost/benefit ratio is worse than the other two approaches. CAN and Chord [31] overlays have been augmented with this approach.

Topology-based node ID assignment aims at mapping the overlay onto the physical network in a way that two nodes which are close to each other in the physical network are also close in the overlay. While this approach clearly brings performance gains, it has several problems attached. The main one is that it sacrifices the uniform population of the ID space. This makes it very difficult to implement this approach on overlays which use a one-dimensional ID space, such as Chord and Pastry. Even if it was possible to implement it on such networks, the non-uniform population of the ID space will cause load balancing issues and can cause node failure clusters. The latter is due to the fact that neighboring nodes are more likely to suf-fer from correlated errors which negatively impact the robustness and also security of the overlay, especially in overlays which dupli-cate entries in neighboring nodes. Despite the issues with this ap-proach it has yielded a delay stretch of 2 relative to the IP delay, when implemented with CAN.

Proximity neighbor selection also constructs a topology aware overlay. But here it is not the node ID assignment which is changed, but rather choosing routing table entries to link to the physically closest node out of all nodes with a node ID in the desired portion of the ID space. This approach requires some flexibility in choosing routing table entries without negatively affecting the number of hops for lookups. Hence it does not work with overlays such as Chord or CAN which have a requirement that routing table entries link to par-ticular portions of the ID space. However, as Castro et al point out it works well with prefix based overlay algorithms such as Pastry.

An example of a structured overlay approach which was designed to be topology aware from the start is TOPLUS. This has been aug-mented with an Application Layer Multicast system called MULTI+ [19]. For the interested reader, Hosseini et al provide a more com-plete list and comparison of Application Layer Multicast protocols [20].

3.4 Strengths and Weaknesses

The different approaches to Application Layer Multicasting have different strengths and weaknesses. Generally, tree first approaches create shared trees, whereas the mesh first and implicit approaches create source-specific trees. But it cannot be concluded that source-specific trees are necessarily the best trees for a particular source as the choice for of a particular tree for a source node is limited by the structure of the mesh. Mesh first approaches are limited in terms of the supported multicast group size, so they are best suited for smaller groups. On the other hand, implicit approaches scale well to large group sizes. Implicit approaches also have a comparable short path length (due to the routing algorithms of the structured overlay) whe-reas for approaches based on unstructured overlays the path length is not limited. Hence latency sensitive applications are best supported by implicit approaches. In terms of control overhead, mesh first al-gorithms pose the highest load as nodes send state information to each other.

4 Conclusions

The main purpose for multicast is to make one-to-many and many-to-many communication more efficient. An IP multicast enabled network allows the source node to send data to a number of nodes, with the sender only sending a single copy of the data and the net-work nodes creating duplicate copies of the data and distributing these to all the multicast group members.

IP multicasting (either as host group multicasting or multi-destination multicasting) has the best potential to achieve this effi-ciency goal. However, host group multicasting requires the setting up of multicast groups before being able to send multicast messages. Furthermore it requires group states in routers. Both of these characteristics make it not suitable for use with P2P overlay operations. P2P overlay messages are typically sent to small, frequently changing groups whereas host group multicasting is best suited to large, infrequently changing groups. Multi-destination multicast is a better match with these requirements.

Table 1 Comparison of IP multicast and application layer multicast [20]

Issues	IP multicast	Application layer multicast
Multicast efficiency	High	Low-medium
Overhead	Low-medium	Medium-high
Ease of deployment	Low	High
OSI layer	Network	Application

However, deployment of multicast enabled routers in the Internet is slow, and hence IP multicast is not yet widely used, leading to the introduction of Application Layer Multicast. The main advantage of Application Layer Multicast is that it does not rely on router support, and hence can be very easily be deployed. However, due to the fact that the multicasting supported by end system hosts, the multicast efficiency is not as high as with IP multicast. This is especially true for implicit approaches which build the multicast trees based on a structured P2P overlay as here the link between the overlay topology and the IP topology is weak. This issue is being tackled by topology aware routing in structured overlays. The key concepts of IP Multi-cast and Application Layer Multicast are summarized in Table 1.

There is also considerable work ongoing in creating hybrid ap-proaches combining Application Layer approaches with IP multi-casting. The IRTF Scalable Adaptive Multicast (SAM) Research Group is active in this area [7, 9].

References

1. L. Aguilar, Datagram Routing for Internet Multicasting, Sigcomm 84, March 1984.
2. M. Ammar. Why Johnny Can't Multicast: Lessons about the Evolution of the Internet. Keynote - NOSDAV 2003.
3. H. Asaeda, B. Manning. Gap Analysis in IP Multicast Dissemination, in Sustainable Internet, Lecture Notes in Computer Science 4866, Springer Verlag, 2007.
4. A. Ballardie. Core Based Trees Multicasting Architecture, Request for Comments (RFC) 2201, Internet Engineering Taskforce (IETF), Sept. 1997.
5. S. Banerjee, B. Bhattacharjee. A Comparative Study of Application Layer Multicast Protocols, wmedia.grnet.gr/P2PBackground/a-comparative-study-ofALM.pdf
6. R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, Explicit Multicast (Xcast) Concepts and Options, Request for Comments (RFC) 5058, Internet Engineering Taskforce (IETF), Nov. 2007.
7. J. Buford. Hybrid Overlay Multicast Framework. IRTF SAM RG. draft-irtf-sam-hybrid-overlay-framework-02. March08, Work in Progress.
8. J. Buford, A. Brown, M. Kolberg. Exploiting Parallelism in the Design of Peer-to-Peer Overlays, Computer Communications Journal, Elsevier Science, Vol. 31(3), 2008.
9. J. Buford, M. Kolberg. Hybrid Overlay Multicast Simulation and Evalua-tion, Proceedings 6th IEEE Consumer Communications Networking Con-ference, Jan. 2009.
10. M. Castro, P. Druschel, Y. Charlie Hu, A. Rowstron. Topology-aware routing in structured peer-to-peer overlay networks, in Future Directions in Distributed Computing, Lecture Notes in Computer Science 2584, Springer Verlag, 2003, extended version: Microsoft Technical Report MSR-TR-2002-82.
11. M. Castro, P. Druschel, A-M. Kermarrec , A. Rowstron. SCRIBE: A large-scale and decentralised application-level multicast infrastructure, IEEE Journal on Selected Areas in Communications (JSAC) 2002.
12. R. Chalmers and K. Almeroth, Modeling the Branching Characteristics and Efficiency Gains of Global Multicast Trees, IEEE Infocom, Anchor-age, AK, USA, April 2001.
13. Y. Chu, S.G. Rao, S. Seshan, H. Zhang. A case for end system multicast, IEEE Journal on Selected Areas in Communications, Volume 20, Issue 8, Oct 2002 Page(s): 1456–1471.
14. J. Chuang and M. Sirbu Pricing multicast communications: A cost-based approach. In Proceedings of INET 1998.
15. S.E. Deering, D.R. Cheriton, Host groups: A Multicast extension to the Internet protocol, Request for Comments (RFC) 966, Internet Engineer-ing Taskforce (IETF), Dec. 1985.
16. C. Diot, B. Levine, J. Lyles, H. Kasserm, D. Balensiefen. Deployment is-sues for the IP multicast service and architecture, IEEE Network, Jan. 2000.
17. P. Francis, Y. Pryadkin, P. Radoslavov, R. Govindan, B. Lindell. YOID: Your Own Internet Distribution, work in progress, available from http://www.isi.edu/div7/yoid/docs/index.html.
18. Q. He, M. Ammar. Dynamic Host-Group/Multi-Destination Routing for Multicast Sessions. Telecommunication Systems, Vol. 28, No. 3–4, pp. 409–433, March 2005.
19. L. Garces-Erice, E.W.Biersack. MULTI+: A robust and topology-aware peer-to-peer multicast service, Computer Communications, Elsevier Science, Vol. 29, pp. 900–910.
20. M. Hosseini, D.T. Ahmed, S. Shirmohammadi, N.D. Georganas. A Survey of Application-Layer Multicast Protocols, IEEE Communications Surveys & Tutorials, Vol. 9(3), 2007, pp. 58–74.
21. K. Katrinis, M. May. Application-Layer Multicast, in Peer-to-Peer Systems and Applications, Lecture Notes in Computer Science 3485, Springer Verlag, 2005.
22. L. Lao, J. Cui, M. Gerla, D. Maggiorini. A Comparative Study of Multi-cast Protocols: Top, Bottom, or In the Middle?, Proceedings of 24th IEEE INFOCOM, 2005, pp. 2809–2814.
23. B. Leong, B. Liskov, and E. D. Demaine. EpiChord: Parallelizing the Chord Lookup Algorithm with Reactive Routing State Management. Computer Communications, Elsevier Science, Vol. 29, pp. 1243–1259.

24. P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric, Proceedings of IPTPS 2002, Cambridge, USA.
25. Luiz R. Monnerat and Claudio L. Amorim. D1HT: A Distributed One Hop Hash Table. In Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS), April 2006.
26. G. Phillips, S. Shenker, and H. Tangmunarunkit. Scaling of multicast trees: Comments on the Chuang-Sirbu scaling law. ACM SIGCOMM 1999.
27. Protocol Independent Multicast Group, Internet Engineering Task Force, http://www.ietf.org/html.charters/pim-charter.html
28. S. Ratnasamy, M. Handley, R. Karp, S. Shenker. Application-level Multicast using Content-Addressable Networks, Proceedings of NGC 2001, pp. 14–29.
29. S. Ratnasamy, S. Shenker, I. Stoica. Routing algorithms for DHTs: Some open questions. Proceedings of IEEE IPTPS 2002.
30. A. Rowstron, P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329–350, November, 2001.
31. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, Proceedings ACM SIGCOMM 2001, August 2001.
32. P. Van Mieghem, G. Hooghiemstra, and R. van der Hofstad. 2001, On the efficiency of multicast. IEEE/ACM Trans. Netw. 9, 6 (Dec. 2001), 719–732.
33. D. Waitzman, C. Partridge, S. Deering. Distance Vector Multicast Routing Protocol, Request for Comments (RFC) 1075, Internet Engineering Taskforce (IETF), Nov. 1988.
34. B. Zhang, S. Jamin, L. Zhang. Host multicast: A framework for deliver-ing multicast to end users, Proceedings IEEE Infocom 2002.
35. B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, J. Kubiatowicz. Tapestry: A Resilient Global-scale Overlay for Service Deployment, IEEE Journal on Selected Areas in Communications, January 2004, Vol. 22, No. 1.
36. S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, Y H. Katz, J. D. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination, Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video, 2001, pp. 11–20.

Multicast Services over Structured P2P Networks

Pilar Manzanares-Lopez, Josemaria Malgosa-Sanahuja, Juan Pedro Muñoz-Gea, and Juan Carlos Sanchez-Aarnoutse

Abstract IP multicast functionality was defined as an efficient method to transmit datagrams to a group of receivers. However, although a lot of research work has been done in this technology, IP multicast has not spread out over the Internet as much as expected, reducing its use for local environments (i.e., LANs). The peer-to-peer networks paradigm can be used to overcome the IP multicast limitations. In this new scenario (called Application Layer Multicast or ALM), the multicast functionality is changed from network to application layer. Although ALM solution can be classified into unstructured and structured solutions, the last ones are the best option to offer multicast services due to the effectiveness in the discovery nodes, their mathematical definition and the totally decentralized management. In this chapter we are going to offer a tutorial of the main structured ALM solutions, but introducing two novelties with respect to related surveys in the past: first, the systematic description of most representative structured ALM solution in OverSim (one of themost

Pilar Manzanares-Lopez
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain,
e-mail: pilar.manzanares@upct.es

Josemaria Malgosa-Sanahuja
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain,
e-mail: josem.malgosa@upct.es

Juan Pedro Muñoz-Gea
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain,
e-mail: juanp.gea.sanchez@upct.es

Juan Carlos Sanchez-Aarnoutse
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain,
e-mail: juanc.sanchez@upct.es

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_31, © Springer Science+Business Media, LLC 2010

popular p2p simulation frameworks). Second, some simulation comparatives between flooding-based and tree-based structured ALM solution are also presented.

1 From IP Multicast to ALM

In the early 1990s, IP multicast functionality was defined as an efficient method to transmit IP datagrams from a data source to a group of receivers. Although a lot of research work has been done in this technology, IP multicasting has not spread over the Internet as much as expected [5]. However, peer-to-peer (p2p) networks can be used to overcome the spread limitations of other multicast technologies. The p2p-based multicast solutions, which are also called Application Level Multicast (ALM) solutions, performs data replication at end hosts and not on the router infrastructure. The end hosts are interconnected among them by unicast connections and they are the only responsible for manage the group communication. Therefore, the multicast functionality is changed from the network level (IP multicasting at network elements) to the application level (at hosts).

ALM solutions can be classified into unstructured and structured solutions. However, the structured ALM solutions have some advantages in offering multicast services due to the effectiveness in the discovery of data and nodes, their mathematical definition and the totally decentralized management.

The main structured ALM solutions are not recent proposals (they are dated around 2002), but there is no doubt that they are still of great importance [20]. For example, in the last years, they have been used to offer Video on Demand (VoD) and Internet Protocol Television (IPTV) services in response to the current multimedia services requirements.

In this chapter we are going to offer a tutorial of the main structured ALM solutions. This tutorial expects to offer to technical and nontechnical readers a global knowledge of this alternative type of multicast communications. In order to facilitate the non-expert readability, the descriptions will be complemented with different examples. For a complete description of each solution, the referenced original papers can be consulted.

This chapter introduces two novelties with respect to other surveys in the past, which just classified and summarized (in a technical way) the different structured-based ALM solutions. First, we are going to describe a recent work done related to these solutions: the codification of the most representative structured ALM solutions in OverSim. OverSim is an open-source overlay network simulation framework for the OMNeT++ simulation environment. We are going to present how Chord-multicast (the most representative flooding-based ALM solution, which is based on Chord) and Scribe (the most representative tree-based ALM solution, which is based on Pastry) are implemented. This will allow readers to extract a methodology to simulate any kind of overlay application in OverSim. Second, based on the implementation of Chord multicast and Scribe, some comparatives of the most representative solution of flooding-based and tree-based structured ALM solutions are presented.

2 Flooding-Based Structured ALM

In the flooding-based structured ALM solutions (also called overlay-per-group solutions), all the members of a multicast group form a particular overlay. Therefore, the multicast transmission is transformed into a broadcast transmission to all the members of the overlay.

2.1 CAN-Multicast

In the middle of 2001, Content Addressable Network (CAN) [14] was introduced as a distributed infrastructure that provides hash table-like functionality (which maps key onto values) on Internet-like scales. Authors proposed the use of CAN not only to peer-to-peer file sharing systems but also to any system that requires efficient insertion and retrieval of content in a large distributed storage infrastructure. Some months later, CAN-multicast [15] was presented as an extension of CAN framework to support multicast service with the dual advantages of simplicity and scalability.

CAN based multicast uses the CAN structure to offer an application level multicast service. If all the nodes in a CAN system belong to a multicast group, multicasting a message only requires flooding the message over the CAN system. To extend this concept, if a subset of CAN nodes are members of a multicast group, CAN based multicast solution proposes to create a mini-CAN system composed of members of the multicast group.

In this solution, the underlying CAN system is used to create the mini-CAN as follows: The multicast group address is mapped onto a point, and consequently the node of the original CAN that owns this point acts as the *bootstrap* node in the construction of the mini-CAN, which is constructed by repeating the usual joining process.

It can be noticed that a node only maintains routing information for those groups for which it is a member or for which it serves as a *bootstrap* node. There is not relation between the amount of information and the number of multicast sources and, likewise, there is not relation to the number of multicast members.

Two mechanisms have been proposed to issue the *multicast* messages to the group:

- The naive flooding algorithm. On receiving a new message, a node forwards it to all its neighbors[1] (except the neighbor from which it received the message) only if that message has not been received yet. This is a simple mechanism but it generates a large amount of duplicate messages.
- The efficient flooding algorithm. In this case, each node analyzes its virtual zone and its routing table before forwarding a message. A multicast source node sends a *multicast* message to all its neighbors. However, on receiving a message, a

[1] As is defined in [14], in a d-dimensional space, two nodes are neighbors if their virtual zone coordinates overlap along $d-1$ dimensions and abut along one dimension.

node will forward it depending on its relationship with the sender node. If the node receives the message from a neighbor with which it abuts along dimension i, it forwards the message to those neighbors with which it abuts along dimension $1\ldots(i-1)$ and the neighbors with which it abuts along dimension i on the opposite side to that from which it received the message. To prevent looping, a node does not forward a message along a particular dimension if the message has already traversed at least half-way across the space from the source dimension along that dimension. Of course, a node caches the received sequence number and does not forward a message it has already received.

In Fig. 1 node A's zone abuts along dimension 2 with the source node's zone. Therefore, node A should forward the *multicast* message to its neighbors with which it abuts along dimension 1 and with which it abuts along dimension 2 on the opposite side to the sender. In this case, there is not any node that satisfies the second condition. On the other hand, node B's zone abuts along dimension 1 with the source node's zone. Consequently, node B only forwards the *multicast* message to those nodes with which it abuts along dimension 1 on the opposite side to the source node. Node C abuts with source node along dimension 2. Therefore, it will forward the message to all nodes with which it abuts along dimension 1 (that is, nodes F, D and E) and to the nodes with which it abuts along dimension 2 on the opposite direction to the sender (in this case, only node G).

As it can be observed, the efficient flooding algorithm does not eliminate the duplicate packet completely. For example, nodes D and E will forward the same message to node H. The original paper [15] proposes an extra rule to reduce the amount of duplicates: If a node P receives a message along dimension 1 and it should send the message to another node Q, node P only forwards the message to Q if the corner C_Q (the Q's coordinate that abuts P along dimension 1 and has the lowest values along dimensions $2\ldots d$) is in contact with P. In our example, C_H corner is in contact with node E. Figures 1 and 2 show the behavior of the efficient and the enhanced efficient flooding algorithms in our example.

2.2 DKS/Chord Multicast

Chord protocol [17, 18] was introduced in the middle of 2001 as a decentralized lookup service to efficiently determine the node that stores a data item in a distributed network. Some years later, in 2003, the Distributed K-ary System (DKS) [7] was introduced. DKS(N,k,f) is a family of infrastructures for building structured peer-to-peer solutions. N is the maximum number of nodes that can be in the overlay, k is the "search arity" in the network and f is the degree of fault tolerance. In fact, as DKS's authors emphasize in the paper, DKS can be seen as a generalization of the Chord system, but using different techniques to maintain the routing tables.

DKS authors propose in [6] a solution to adapt DKS (and therefore, it can also be applied to Chord) to achieve multicast replication service in large scale and highly dynamic environments. This proposal follows the same approach as CAN based

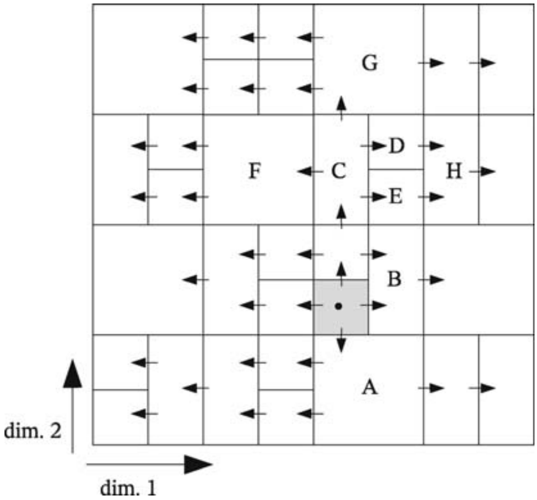


Fig. 1 An example of the efficient flooding algorithm in a CAN-multicast system. The node responsible for the shadowed zone is the multicast data source

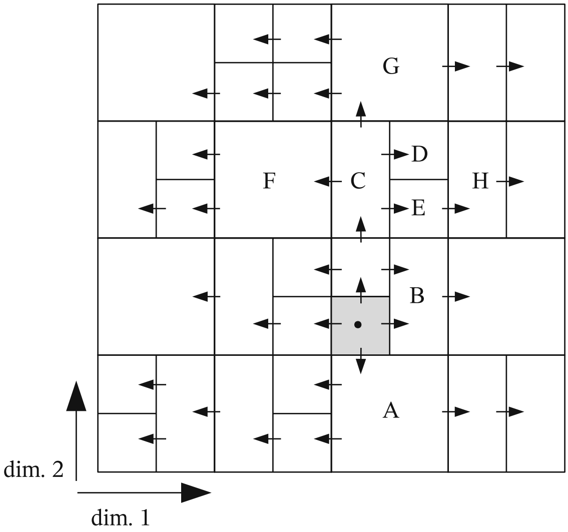


Fig. 2 An example of the enhanced efficient flooding algorithm in a CAN-multicast system. Note that now there are no double packets in any

multicast: Members of a multicast group self-organize in an instance of Chord or DKS(N,k,f). Consequently, a multicast transmission to a certain group just requires to broadcast the Chord or DKS(N,k,f) instance associated to that group. However, this proposal differs from the CAN based multicast in the flooding mechanism. As it is explained in Section 2.1, the efficient flooding algorithm (and also its enhanced version) produces redundant messages. The Chord/DKS flooding algorithm that is called “correcting broadcast” eliminates any redundant packets.

In this solution, the underlying $DKS(N,k,f)$ instance is used to create a new $DKS(N_g,k_g,f_g)$ corresponding to the multicast group g. Node n of $DKS(N,k,f)$ that first joins the group must determine the group parameters, that is, the value of N_g, k_g, f_g and also the hast function H_g used to map the nodes onto the multicast instance. These parameters and the address of node n should be available to allow other nodes to join the system.

The "correcting broadcast" algorithm is able to deliver the multicast packets to all the group members without redundant messages. It works as follows: the source node sends the *multicast* message to all the nodes in its routing table associated to the multicast group g. The routing table is analyzed level by level in the counter clockwise direction. A *multicast* message is sent to the node responsible for each interval. That message contains the multicast information and a limit value (*Limit*) that indicates the receiving node which part of the ring should it cover during the multicast process.

When a node n receives a *multicast* message from a node n', it only sends the *multicast* message to each node x that n considers responsible within the interval $(n, Limit)$. The limit value sent to each node x is computed to guarantee that the intervals covered by two arbitrary nodes are disjoint.

Figure 3 represents an example of the "correcting broadcast" procedure[2] corresponding to a multicast group that has created a $DKS(64,2,f_g)$ instance. The overlay infrastructure of a DKS instance with k=2 can be considered equivalent to a Chord system infrastructure. Node 0, the multicast source in this example, sends a *multicast* message to node 33 (the responsible node for interval $[32,0)$ at level (1). The limit value included in this message is 0. Next, it sends the *multicast* message to node 18 (the responsible node for the next interval $[16,32)$ at level (2). In this case, the limit value is 33, the previous responsible node to which it has sent a *multicast* message. Next, it sends the *multicast* message to node 8 (the responsible node for the next interval $[8,16)$ at level (3). In this case, the limit value is 18, the previous responsible node to which it has sent a *multicast* message. In the next level, the responsible node of the interval ($[4,8)$) is also node 8. Therefore, node 0 does not send the message again to node 8. Finally, node 0 sends the message to node 2 (the responsible node for the next interval $[8,16)$ at level 5 with a limit value of 8). Observing the equivalent Chord routing tables, node 0 sends the message to all the nodes in its routing table, which is analyzed from the last to the first entry. Each message includes the multicast information and the limit value.

When a node receives the *multicast* message, it repeats the above process but only sends the message to nodes in its routing table within the interval $(node, received\_Limit)$. For example, node 18 only sends the *multicast* message to nodes 29, 24 and 20, the only nodes of its routing table in the interval $(18,33)$.

[2] We consider that all nodes have updated routing tables by means of the "correction-on-use" standard procedure defined in [7].

Node routing table
(using DKS nomenclature)

	N0			N18	
L1	[00,32)	00	L1	[18,50)	18
	[32,00)	33		[50,18)	54
L2	[00,16)	00	L2	[18,34)	18
	[16,32)	18		[34,50)	37
L3	[00,08)	00	L3	[18,26)	18
	[08,16)	08		[26,34)	29
L4	[00,04)	00	L4	[18,22)	18
	[04,08)	08		[22,26)	24
L5	[00,02)	00	L5	[18,20)	18
	[02,04)	02		[20,22)	20
L6	[00,02)	00	L6	[18,19)	18
	[01,02)	02		[19,20)	20

Node routing tables
(using Chord nomenclature)

N0		N18	
$0+2^0$	02	$18+2^0$	20
$0+2^1$	02	$18+2^1$	20
$0+2^2$	08	$18+2^2$	24
$0+2^3$	08	$18+2^3$	29
$0+2^4$	18	$18+2^4$	37
$0+2^5$	33	$18+2^5$	54

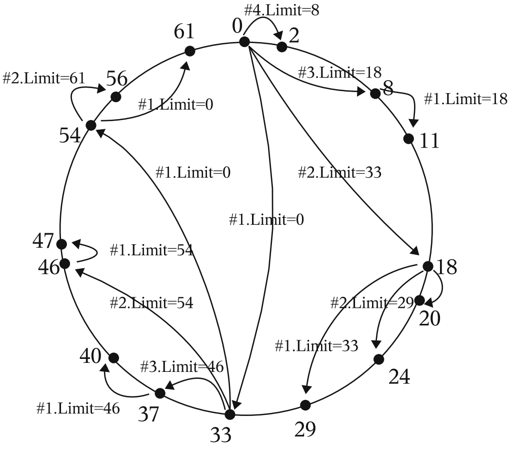


Fig. 3 Multicast transmission in a DKS multicast system, where N_g=64 and k_g=2. The overlay infrastructure of this system can be considered equivalent to a Chord system overlay. Tables in the left side are the routing tables of nodes 0 and 18 (using DKS nomenclature and also using Chord nomenclature). Graph in the right side represents the data packet generation when node 0 is the multicast source. The *#i* nomenclature in the figure indicates the order in which multicast data packets are sent by a node

3 Tree-Based Structured ALM

In tree-based structured ALM solutions (also called tree-per-group solutions), nodes from different multicast groups join the same overlay. Then, each multicast group forms a particular multicast tree built on top of the overlay network.

3.1 Scribe

At the end of 2001, Pastry [16] was presented as a scalable, distributed object location and routing substrate for peer-to-peer applications in a potentially very large overlay network of nodes connected through the Internet. One of the peer-to-peer

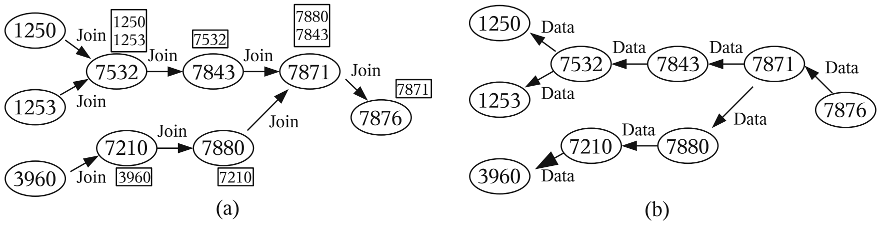


Fig. 4 (**a**) Creation of multicast group tree in Scribe. Nodes 1250, 1253 and 3960 join the multicast group whose rendez-vous node is 7876. (**b**) Data dissemination in Scribe. Any node can deliver data to the multicast group rendez-vous point (node 7876). Next, that node will start the data dissemination along the multicast tree

application which uses Pastry as substrate is Scribe [2]. Scribe, which is defined as a scalable publish/subscribe system, is in fact an ALM service: A topic (multicast group) is created, some nodes join the topic (multicast group's members) and an efficient multicast tree is built for an efficient dissemination of events (multicast data) to the topic's subscribers.

Unlike CAN based multicast and DKS/Chord multicast solutions, where the members of each multicast group must create a different CAN or DKS/Chord instance, the Scribe system is common to all the multicast groups. A multicast group is identified by the hash of the multicast group's textual name concatenated with its creator's name. Scribe node whose key-identifier is closest to the group-key-identifier will act as the rendez-vous point for the multicast group, as the root node in the multicast tree creation, and as the first multicast information forwarder.

The tree is formed by joining the routes from each member to the rendez-vous point. When a Scribe node wants to join a multicast group, it must send a *subscription* message toward the rendez-vous point of that group using the Pastry routing algorithm. Each node along the route checks if it is already part of that multicast tree. If the node is not already part of the multicast tree, it creates a children table for the multicast group and adds an entry containing the IP address and node identifier of the sender node. Then, the node sends the *subscription* message to the next node along the route to the rendez-vous point (the new branch composed of the joining node and its parent must join the multicast tree). However, if the node is already part of the multicast tree, it just accepts the source node as a child, adding a new entry in the children table of the multicast group, but the *subscription* message is not forwarded toward the rendez-vous point. As it can be seen, some scribe nodes of a multicast tree are members of that multicast group, but other scribe nodes are just "multicast information forwarders".

In Fig. 4a, node 1250 sends a *subscription* message toward the node responsible for the multicast group-key-identifier 7876. Node 7532 receives the message, creates a children table for that multicast group and adds an entry containing node 1250's information. Then, 7532 routes the message toward the multicast group root. When node 1253 joins the group, the *subscription* message also reaches node 7532. Due to now that node forms part of the multicast group tree, it just adds a new entry to the associated children table.

To offer reliability, each multicast tree must be maintained. Each parent node should periodically send *heartbeat* messages to all its children. The absence of *heartbeat* messages makes a child suspect that its parent has failed. In this case, the node sends a new *subscription* message to join the multicast tree again. On the other hand, when a Scribe node wants to leave a multicast group, it must check its children table. If there are no entries (it is a leaf node), it sends an *unsubscription* message to its parent in the multicast tree. When the parent receives the *unsubscription* message, it updates its children table removing the entry about the departing node. If the routing table is now empty, it sends the unsubscription message to its parent. Otherwise, the message stops in that node.

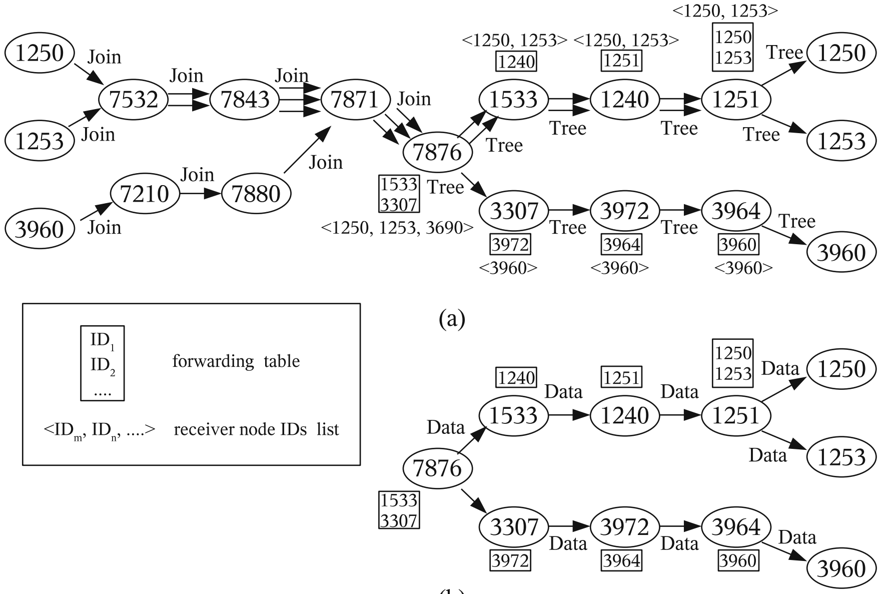
Any Scribe node can deliver information to any multicast group. The sender just routes the messages to the multicast group's rendez-vous node and then, the rendez-vous node disseminates the messages along the multicast tree (see Fig. 4b). Each node will send the *data* messages to all the nodes within the associated children table.

3.2 Bayeux

At the beginning of 2001, Tapestry [21] was presented as an overlay location and routing infrastructure that provides location-independent routing of messages directly to the closest copy of an object or service without any centralized resources. The inspiration for Tapestry, and also for Pastry, was the location and routing mechanisms introduced by Paxton, Rajamaran and Richa (PRR) in [13]. Some months later, Bayeux [22] was presented as an efficient ALM solution built on Tapestry.

Like Scribe, it supports multiple groups and it builds a multicast tree per group. However, trees are built in a quite different method. A member joins a multicast group by sending a *join* message toward the root (that is, toward the surrogate node of the multicast group-key-identifier). Then, the root node adds the identity of the new member to its list of receiver node IDs that it is responsible for, updates its forwarding table, and uses Tapestry to route another message (a *tree* message) back to the new member. Every Bayeux node along the *tree* message route adds the new member identifier to its list of receiver node IDs, updates its forwarding table and resends the message. Any Tapestry node can deliver information to any multicast group. The sender just routes the information to the multicast group's root and then, the root disseminates the messages along the multicast tree. Each node duplicates (if it is necessary) and forwards data packets according to its forwarding table.

In Fig. 5a, nodes 1250, 1253 and 3960 send a *join* message toward the multicast group root. Each time the root (node 7876) receives a *join* message, it routes a *tree* message toward the new member. That *tree* message, which goes through different nodes from the *join* message, is used to fill out the forwarding table at each involved node.



(a)

(b)

Fig. 5 (**a**) Creation of a multicast group tree in Bayeux. Multicast members send a *join* message to the surrogate node of the multicast group-key-identifier (node 7876). This node routes a *tree* message to each new member. Those *tree* messages are used to updated the forwarding table of the involved nodes. (**b**) Data dissemination in Bayeux. Any node can send data to the multicast group rendez-vous node (node 7876). Next, that node will start the data dissemination along the multicast tree

Comparing Scribe and Bayeux solutions, both create very different multicast trees. A Scribe multicast tree is formed by the Pastry routes from each member to the multicast group root. However, a Bayeux multicast tree is formed by the Tapestry routes from the root to each member. Therefore, the Scribe root node cannot automatically rebuild the tree if one of the intermediate nodes fails. In addition, Tapestry works better than Pastry under a dynamic scenario. As a consequence, Scribe network management is more intensive than in Bayeux. On the other hand, Bayeux requires more nodes to maintain group membership information (each node in a route maintains information about leaf nodes in that route). Scribe nodes only maintain information about their children. Therefore, in Scribe, the join and departure messages are both rapid to process and easier to manage than in Balleux.

4 OverSim: A Useful P2P Simulation Tool

Nowadays, there are several open-source P2P simulation frameworks available; next, we briefly describe the most relevant ones.

P2PSim [10] is a discrete event simulator for structured P2P networks written in C++. It was developed in the IRIS project at the Massachusetts Institute of Technology. There are a range of different underlying network models, including: end-to-end time graph, G2 graph, GT-ITM, random and Euclidean. It can simulate up to 3,000 nodes on a single PC. However, it is largely undocumented and therefore hard to extend. Additionally, it has a limited set of statistics.

PeerSim [11] uses query-cycle or discrete-event techniques to simulate unstructured P2P networks. It is written in Java, and it was developed in the BISON project at the University of Bologna (Italy). The components can be implemented to gather statistical data and it can simulate up to 1,000,000 nodes. However, the underlying communication network is not modeled and only the query-cycle simulator is documented. In any case, it has no any build-in facilities to simulate structured P2P systems.

PlanetSim [12] is a discrete event simulator written in Java. It was developed in the Planet project at the University of Rovira i Virgili (Spain). It has a detailed tutorial and it can simulate up to 100,000 nodes. Its design is thoroughly documented. However, it has a limited simulation of the underlying TCP/IP network and it does not provide any mechanism to gather statistics. Therefore the programmer must implement it.

OverSim [19] is a new simulation framework based on the well-known discrete event simulator OMNeT++ [9]. It has been developed in the ScaleNet project at the University of Karlsruhe (Germany). It can simulate up to 100,000 nodes and its design is thoroughly documented. Several overlay protocols are already implemented and most of them are structured P2P protocols, like Chord, Koorde and Pastry. Unstructured peer-to-peer protocols like GIA are available as well. It implements three underlying network models: In the so-called Simple model, data packets are sent directly from one overlay node to another by using a global routing table. The INET underlay model includes simulation models for all network layers (MAC, IP, TCP/UDP). Finally, in the SingleHost underlay model each OverSim instance only emulates a single host, which can be connected to other instances over an existing real network; that is, overlay nodes are simulated but the MAC, IP, TCP/UDP layers work in a real scenario. The source code and the API are well documented and the simulator is able to collect statistical data.

A more comprehensive survey of P2P simulation frameworks can be found in [8]. Among all the available tools OverSim has been chosen because of its flexibility: Its modular design and the use of the Common API [4] facilitate its extension with new protocols. In addition, its flexible underlying network scheme can be very useful during the application development. Another reason to select OverSim is that its code is well documented.

In this section it is introduced OMNeT++, the well-known network simulation software, used by OverSim to simulate physical, link and network OSI layers. Next, OverSim simulation tool is presented. Finally, the Chord-multicast and Scribe protocol implementations in OverSim are described. Scribe implementation was developed by the OverSim programmers and Chord-multicast has been developed by the authors of this chapter. Scribe ALM solution has been chosen to be studied because it was already implemented in OverSim. On the other hand, Chord-multicast has been chosen because it was very easy to implement in OverSim.

4.1 OMNeT++

OMNeT++ is a well-known open-source network simulation engine and it is one of the most used simulation frameworks for the research community. Additionally, its coding procedure is well documented and it also offers a multiplicity of network simulation models already implemented.

The development of a simulation model with OMNeT++ has several steps, which are briefly pointed out:

1. Description of the system structure using the NED language. This step consists of defining the modules which compose our system and the relationship among them. This modules are called *simple modules*, and are described in NED files. In addition, a file for the *network module* must be created, which is a module composed by the simple modules created before. The modules are interconnected among them using unidirectional gates, and they communicate among them by means of messages.
2. Implementation of the simple modules in C++. In this step the actions of each simple module are implemented. Every simple module is coded by means of a derived class by inheritance of the class *cSimpleModule*. The node functionality must be implemented in the *handleMessage()* method. This function is invoked by the simulation kernel when a message arrives to a module.
3. Compilation. In order to obtain an executable, the modules are compiled and linked with the simulation library.
4. Configuration of the simulation. It is necessary to establish the appropriate parameters for every simulation in the omnetpp.ini file.

OMNeT++ implements many features needed for visualization. The built-in GUI displays network topologies, nodes and messages. Its features allow for deeper inspection of message contents and node variables. The GUI can be disabled for faster simulation.

4.2 OverSim

OverSim is an overlay simulation tool built over OMNeT++ as simulation engine. The layered structure of this simulator appears in Fig. 6 [1]. OverSim includes all the facilities about underlay and overlay networks (and its primitives). Users only need to code the overlay application in the application layer.

The communication between the overlay layer and the application layer uses the API described and well documented in [4]. It should be noted that the application layer is at the same time divided into two different sublayers (Tier 1 and Tier 2) as shown in Fig. 6. A simple application can be programmed at Tier 1 sublayer (for example, the KBRTestApp). However, more complex applications which use the services provided by the application implemented at Tier 1 (for example, DHT) must be programmed at Tier 2 (DHTTestApp).

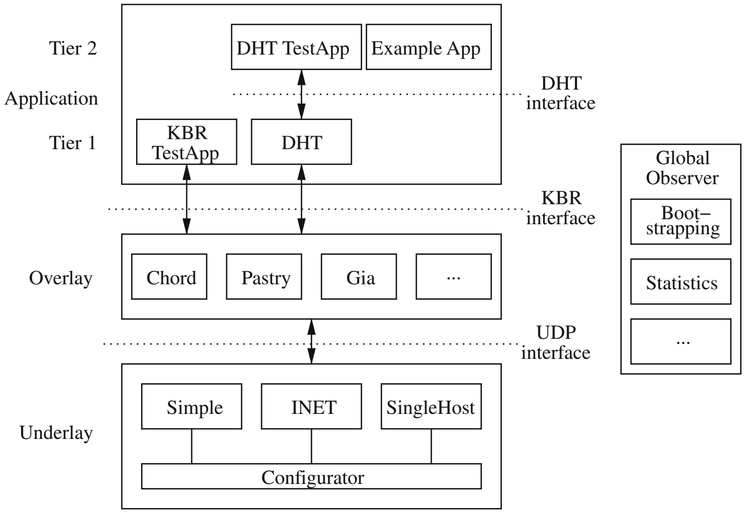


Fig. 6 OverSim layered structure

To build a P2P simulation using OverSim it is necessary to implement the modules in the application layer and to define the messages which will be interchanged among them. Then, the files are compiled in order to obtain an executable instance of the program.

4.3 Chord-Multicast Implementation

4.3.1 Simulator Structure

We have implemented a simulation model of the Chord-Multicast network for the OverSim framework. The architecture of this model can be seen in the Fig. 7. The fact that each multicast group requires the creation of a new Chord overlay network determines the most relevant characteristics of the Chord-multicast implementation. First, *n* Chord-Multicast modules (Chord-Mult-1, ..., Chord-Mult-n) are added in the overlay layer. Each of this modules is used by a different multicast group. The functionality of these modules is equivalent to the Chord module, but adding the multicast transmission facility. Secondly, the communication between the Application layer and the Chord module follows the previously described KBR interface.

However, we have created a new interface for the communication with the Chord-Multicast modules (Chord-Mult interface).

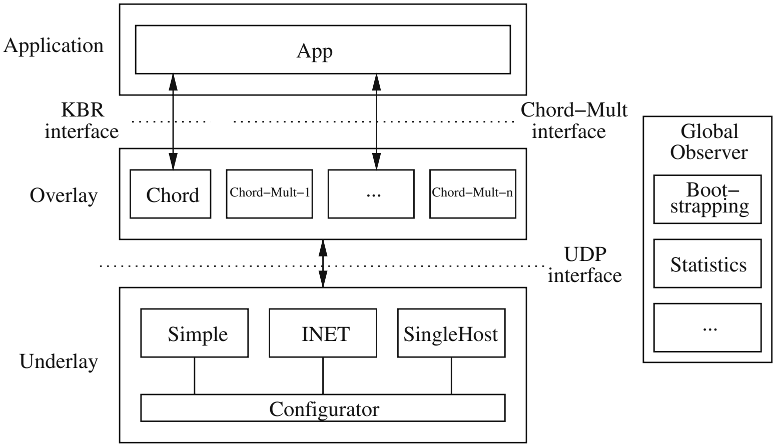


Fig. 7 Chord-multicast simulator structure

4.3.2 Functionality

When a node wants to join to a specific Chord-Multicast group, it needs to discover the bootstrap node of that multicast group. This node is obtained performing a lookup operation in the Chord network, using the multicast group identifier as the *Key*. This operation involves the use of the *route()* function of the KBR interface, which will route the query to the node responsible of the associated key. In this node, the query will be directed towards the application layer (App), which implements an storage functionality that associates the identifier of the multicast group with the IP address of the bootstrap node for that group. This information will be replied towards the requester node (that is, the new multicast node).

Next, the node uses the Chord-Mult interface to join the desired multicast group. This operation is performed using the *join(bootstrapNode)* function. The Chord-Mult module takes the bootstrap node and joins to the specific Chord-Multicast overlay network in the same way as Chord.

Once a node has joined to the multicast group, it can send a multicast message to the members of this group. That is done using the *broadcast()* operation. When the Chord-Mult module receives this request, it consults its Finger Table with direction down-up, and sends a multicast message to every node in this table. In addition, every multicast message is attached with a *limit* field that represents the previous destination node of the multicast message (the chord-multicast broadcast functionality is described in Section 2.1).

When the rest of nodes receive the multicast message, they will perform an equivalent operation, but they will make sure that the identifier of the node is always smaller than the *limit* field of the received message.

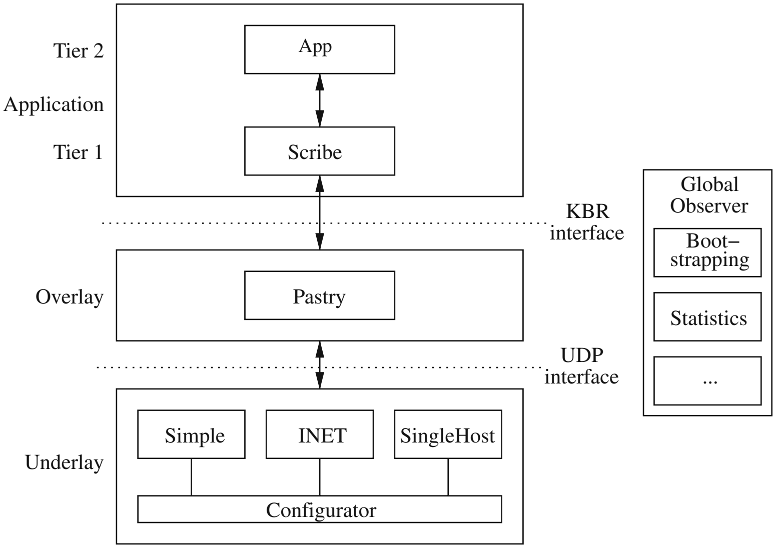


Fig. 8 Scribe simulator structure

4.4 Scribe Implementation

4.4.1 Simulator Structure

The OverSim developers have programmed a Scribe simulation model. Contrary to Chord-multicast, in scribe each multicast group uses the Pastry overlay to manage all the Scribe functionality. Therefore, the structure of the simulator strongly fits in with the OverSim structure. Figure 8 represents the architecture of this simulation model. Due to this fact, the Scribe functionality could be implemented in the application layer.

4.4.2 Functionality

In this case, the Application can communicate with the Scribe multicast functionality using three operations: *joinGroup(GroupIdentifier)*, *leaveGroup(GroupIdentifier)* and *sendDataToGroup(GroupIdentifier)*, that are used to join, to leave and to send data to a specific group, respectively.

Each multicast group is associated to a key (the GroupIdentifier). Thus, the node responsible of that key becomes the root (rendez-vous point) of the multicast group tree. Nodes join the group using the *joinGroup(GroupIdentifier)* function. This function makes two operations: first, it inserts the group into a list of groups associated to this node; and secondly, it sends a subscription message towards the rendez-vous point of that group using the route function implemented in Pastry. When a node received this kind of message, it checks if it is a member of the multicast group tree. If so, it terminates the subscription message forwarding; otherwise, it inserts its relevant node information into the message and forwards it towards the rendez-vous point, thus subscribing to the multicast group tree. In either case, it adds the information of the previous node to its list of children in the group multicast tree.

Any overlay node may multicast a message to the group using the *sendDataTo Group(GroupIdentifier)* function. This function routes the message using the GroupIdentifier as key. Upon receiving this message, the root of the multicast tree forwards the message to its children in the tree, and so on recursively until reaching the leaves of the tree. Finally, the *leaveGroup()* function deletes the group information from the list of groups the node belongs to.

5 Performance Evaluation

This section shows the performance evaluation of a flooding-based multicast approach (Chord-multicast) and a tree-based multicast approach (Scribe) using OverSim simulator. The flooding approach to provide multicast creates a separate overlay network per multicast group and leverages the routing information already maintained by a group's overlay to broadcast messages within the overlay. The tree approach uses a single overlay and builds a spanning tree for each group, on which the multicast messages for the group are propagated. In [3] authors examine the performance of tree and flooding approaches, on Pastry and CAN. Authors evaluated the delay to deliver multicast messages, the network load and the nodes load. Results show that the tree-based approach achieves lower delay and overhead than flooding-based approach. The biggest disadvantage of flooding approach is the cost of constructing an overlay for each group. They also show that multicast trees built using Pastry provide higher performance than one built using CAN. In the same way, the flooding approach works better in CAN than in Pastry.

In this work we evaluate Chord-multicast and Scribe using OverSim simulator. The underlay model used in the simulations has been the Simple network model. With this model the underlying network is not simulated, that is, the messages are directly sent from one overlay node to another one. The churn model for the dynamic scenario is the *LifetimeChurn* model, where the *lifetime* and *deadtime* follow a Weibull distribution with mean 10,000 seconds. Results are obtained for different scenarios, varying the number of nodes and the multicast group sizes. In all the figures, the X-axis values (a/b) represent an scenario composed of a Chord (or Pastry) nodes, of which only b nodes join the same multicast group. For example, the

500/125 value represents an scenario composed of 500 Chord (or Pastry) nodes, of which 125 nodes join the same multicast group.

The result associated to each scenario is the mean value of the results obtained after running five independent simulations. The confidence intervals obtained in all the cases are very reduced. For that reason, to simplify the figures, they are not represented.

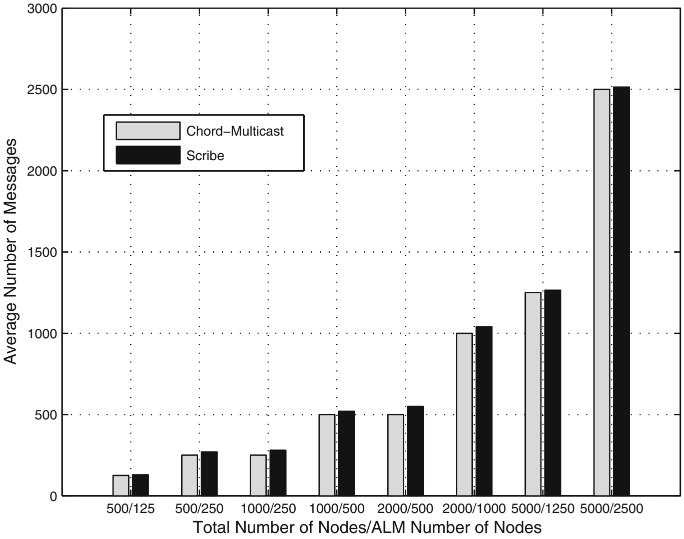


Fig. 9 Average number of application layer data messages in a multicast transmission

Figure 9 represents the average number of data messages that a multicast transmission involves. That is equivalent to the number of nodes involved in the multicast process. In the Chord-Multicast scenario the results match with the multicast group size, and they are independent of the total number of nodes in the network. Using the Chord multicast system, the multicast members create a mini-Chord. Then, the Chord-multicast procedure distributely generates just one data packet for each member. On the other hand, in the Scribe scenario the average number of messages is always greater than in the Chord-Multicast case. Each multicast tree in Scribe is formed by Scribe nodes joined to the group, but also by Pastry nodes that act just as forwarders, and all the nodes in the tree receives the data message to contribute to the forwarding procedure to all the multicast group members. However, this excess number of data messages is not so important as it could be expected. In most of the scenarios, the size of the key space (usually 2^{160}) is much greater than the size of the network. In this situation, Pastry routing algorithm allows a node to reach the rendez-vous point node in a few hops (the average path length has a complexity of $O(logN)$). Therefore, the generated Scribe tree has a very limited depth, and consequently the number of non-multicast forwarder nodes is minimized. Another important conclusion in Scribe is that the excess number of data packets (respect to

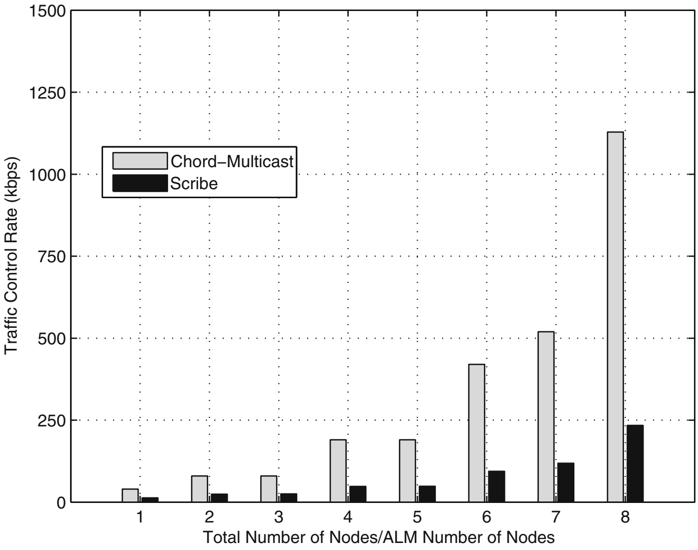


Fig. 10 Control traffic rate in chord-multicast and scribe

the multicast group size) depends on the total size of the overlay network, although that dependence is not strong due to the reason explained before.

Figure 10 represents the control traffic rate in the Chord-Multicast and Scribe scenarios after a multicast group is formed (that is, the control traffic associated to the joining of all the nodes is not considered). In Chord-Multicast, it is assumed that the control traffic is due to *stabilize()*, *fix_fingers()* and *notify()* functions. Every node runs *stabilize()* periodically to learn about newly joined nodes. The function *stabilize()* in node *n* notifies (by means of *notify()* function) to its successor of its existence, giving the successor the chance to change its predecessor to *n*. Each node periodically calls *fix_fingers()* function to initiate its finger table and to actualize and verify all the finger table entries. The stabilize message is sent every 20 seconds, and it has a length of 256 bits. The notify message has a length of 256 bits. Finally, *fix_fingers* messages are sent every 120 seconds, and they have a length of 264 bits.

In Scribe, it is assumed that the control traffic is composed by the *heartbeat* messages, the *refresh* messages and the control traffic due to the routing table maintenance in Pastry. Periodically, each non-leaf node in the tree sends a *heartbeat* message to its children. A child suspects that its parent is faulty when it fails to receive *heartbeat* messages. Upon detection of the failure of its parent, a node calls Pastry to route a *join* message to the group's identifier. Pastry will route the message to a new parent, thus repairing the multicast tree. On the other hand, children table entries are discarded unless they are periodically refreshed by an explicit message from the child. The heartbeat message is sent every 3 seconds, and it has a length of 168 bits. On the other hand, *refresh* messages are sent every 10 seconds, and the have a length of 368 bits.

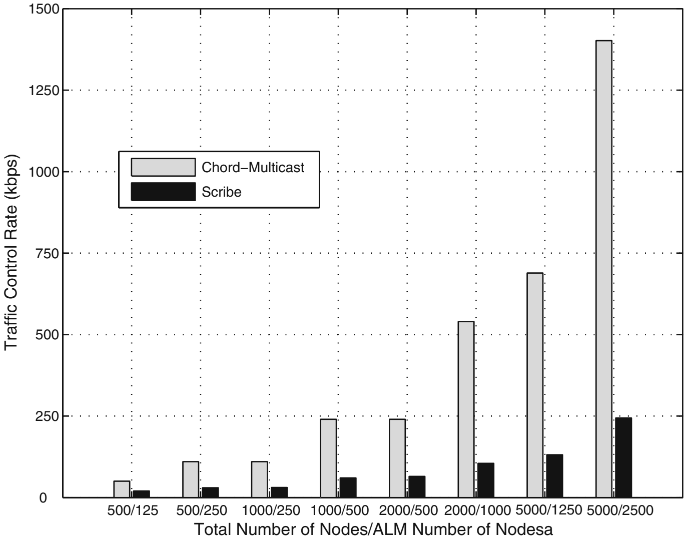


Fig. 11 Control traffic rate in chord-multicast and scribe in a dynamic scenario

In this figure, it can observed that the control traffic rate is always much greater in Chord-Multicast than in Scribe. In both cases, this control traffic is mainly due to the Chord ring and Pastry infrastructure maintenance. In Chord-multicast, each node must check each routing table entry independently of the number of nodes. However, in Pastry only the filled entries are checked. If more than one multicast group coexisted at the same time, the Pastry control traffic would be shared by all of them. However, in Chord-multicast, each multicast group requires its own overlay network infrastructure maintenance.

It is also interesting to point out that the scenario studied in Fig. 10 corresponds with an ideal scenario, without node joining, leaving and failures (that is, without churn). Therefore, that figure represents in fact background maintenance traffic. Figure 11 represents the results obtained in a dynamic scenario. In this case, the control traffic is always higher in both ALM systems. However, this increase is higher in Chord multicast than in Scribe.

Finally, Fig. 12 shows the average number of overlay hops to reach all the multicast group members. This parameter is directly related with the delay and the bandwidth required to perform the multicast communication. In Scribe, the number of overlay hops depends on the overlay network size as well as on the multicast group size. Obviously, in Chord-multicast the number of overlay hops only depends on the multicast group size. In addition, the obtained simulation results seem to suggest that, in Chord-multicast scenarios, the increment of this parameter with the multicast group size follows a logarithmic law, which is beneficial for large multicast group sizes.

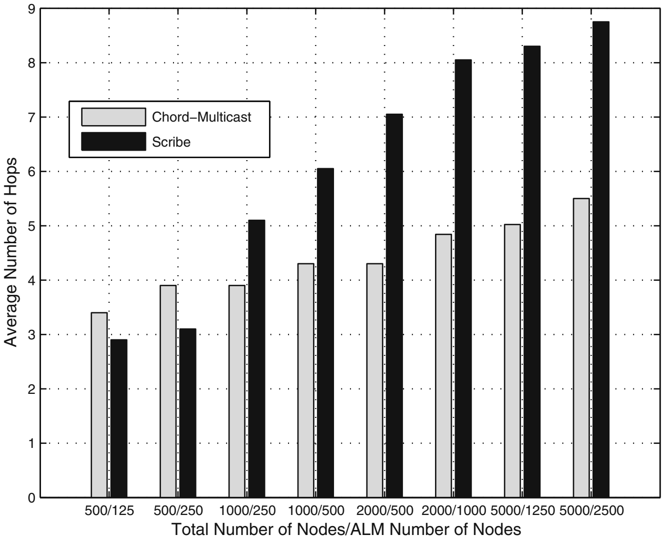


Fig. 12 Average number of overlay hops to reach all the multicast members

6 Summary

IP Multicast technology has not spread over the Internet as much as expected. Nowadays, overlay networks have become an alternative solution to offer multicast services. In these solutions, which are called Application Level Multicast (ALM), multicast end hosts are the only elements in charge of the multicast functionality. In this paper we have described the main ALM solutions based on structured P2P solutions, which are classified into flooding-based solutions and tree-based solutions.

Some performance metrics of the most representative solution of both alternatives (Chord-multicast and Scribe) have been evaluated by means of simulation. Among the available open-source P2P simulation framework, OverSim has been chosen because of its flexibility and well documented code. These characteristics have allowed to easily implement Chord-multicast, contributing to the OverSim community.

7 Future Research Directions

The multicast services over structured P2P networks can facilitate the rapid deployment of the Information Technologies in those scenarios in which IPv4 infrastructure cannot. For example, multimedia distribution services like VoD or live-TV (IPTV) can be implemented using, among others, multicast structured P2P services for both managing and transmission purposes. Nevertheless, a research work must be done in order to guarantee the demanding QoS requirements of these services.

Traditionally there are two scenarios in which multimedia contents distribution can take place: ISP private distribution networks and Internet. However, recently a new scenario has arisen in the market. It is known as in-home and in-building (or community) network. In this case, the network is commonly built using the low-power communication infrastructure with the PLC (Power Line Communications) technology as a transmission media. However, this media presents strong asymmetries among the set of outlets (i.e., the connection point). This fact recommends the use of self-organized multicast distribution systems to distribute the information and manage the group membership. It is an open question how each structured ALM solution works in this new scenario and what are their performance metrics.

Acknowledgement

This research has been supported by project grant TEC2007-67966-C03-01/TCM (CON-PARTE-1) and it is also developed in the framework of "Programa de Ayudas a Grupos de Excelencia de la Region de Murcia, de la Fundacion Seneca, Agencia de Ciencia y Tecnologia de la RM (Plan Regional de Ciencia y Tecnologia 2007/2010)". Juan Pedro Muñoz-Gea also thanks the Spanish MEC for a FPU (AP2006-01567) pre-doctoral fellowship.

References

1. Baumgart, I., Heep, B., Krause, S.: OverSim: A Flexible Overlay Network Simulation Framework. In: Proceedings of 10th IEEE Global Internet Symposium (GI'07), May 11, pp. 79–84. IEEE (2007)
2. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.: Scribe: A Large-scale and Decentralized Application-level Multicast Infrastructure. IEEE Journal on Selected Areas in Communications **20**(8), 100–110 (2002)
3. Castro, M., Jones, M.B., Kermarrec, A.M., Rowstron, A., Theimer, M., Wang, H., Wolman, A.: An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer Overlays. In: Proceedings of the IEEE INFOCOM, San Francisco, April, IEEE (2003)
4. Dabek, F., Zhao, B., Druschel, P., Stoica, I.: Towards a Common API for Structured Peer-to-Peer Overlays. In: Proceedings of the 2nd International Workshop on Peer-to-Peer Systems, 20-21 February, pp. 33–44. Springer (2003)
5. Diot, C., Levine, B.N., Lyles, B., Kassem, H., Balensiefen, D.: Deployment Issues for the IP Multicast Service and Architecture. IEEE Network, Special Issue on Multicasting **14**(1), 78–88 (2000)
6. El-Ansary, S., Alima, L.O., Brand, P., Haridi, S.: Efficient Broadcast in Structured P2P Networks. In: Proceedings of the 2nd. International Workshop on Peer-to-Peer Systems (IPTPS), LNCS 2735, Berkeley, CA, USA, February 20–21, pp. 304-314. Springer (2003)
7. Ghodsi, A., Alima, L.O., El-Ansary, S., Brand, P., Haridi, S.: DKS(N,k,f): A Family of Low Communication, Scalable and Fault-Tolerant Infrastructures for P2P Applications. In:

Proceedings of the 3rd. International Workshop on Global and P2P Computing on Large Scale Distributed Systems (CCGRID2003), Tokyo, Japan, May 12-15, pp 344–350. IEEE (2003)

8. Naicken, S., Livingston, B., Basu, A., Rodhetbhai, S., Wakeman, I., Chalmers, D.: The State of Peer-to-Peer Simulators and Simulations. ACM Computer Communications Review, **37**(2), pp. 95–98. ACM (2007)
9. OMNet++ Discrete Event Simulation System. http://www.omnetpp.org/
10. P2PSim: A Simulator for Peer-to-Peer Protocols. http://pdos.csail.mit.edu/p2psim/.
11. PeerSim: A Peer-to-Peer Simulator. http://peersim.sourceforge.net/.
12. PlanetSim: Object Oriented Simulation Framework for Overlay Networks. http://planet.urv.es/trac/planetsim/.
13. Plaxton, C.G., Rajaraman, R., Richa, A.W.: Accessing Nearby Copies of Replicated Objects in a Distributed Environment. In: Proceedings of the 9th. ACM Annual Symposium on Parallel Algorithms and Architectures, pp. 311–320, New York. ACM (1997)
14. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: Proceedings of ACM SIGCOMM, San Diego, CA, USA, August 27-31, pp. 161–172. ACM (2001)
15. Ratnasamy, S., Handley, M., Karp, R., Shenker, S.: Application-level Multicast using Content-Addressable Networks. In: Proceedings of the 3rd. International Workshop of Networked Group Communication (NGC), LNCS 2233, London, UK, November 7-9, pp. 14–29. Springer (2001).
16. Rowstron, A., Druschel, P.: Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), LNCS 2218, Heidelberg, Germany, November 12–16, pp. 329–350. Springer (2001)
17. Stoica, I., Morris, R., Karger, D.R., Kaashoek, M.F., Hari, Balakrishnan H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proceedings of ACM SIGCOMM, San Diego, CA, USA, August 27–31, pp. 149–160. ACM (2001)
18. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. IEEE/ACM Transactions on Networking. **11**(1), 17–32 (2003)
19. The OverSim P2P Simulator. http://oversim.org/.
20. Yiu, W.P.K., Jin, X., Chang, S.H.G.,: Vmesh: Distributed segment storage for peer-to-peer interactive video streaming. IEEE Journal on Selected Areas in Communications **25**(9), 1717–1731 (2007)
21. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J. D.: Tapestry: A Resilient Global-Scale Overlay for Service Deployment. IEEE Journal on Selected Areas in Communications **22**(1), 41–53 (2004)
22. Zhuang, S.Q., Zhao, B.Y., Joseph, A.D., Katz, R.H., Kubiatowicz, J.D.: Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination. In: Proceedings of the 11th. International Workshop on Network and Operanting Systems Support for Digital Audio and Video (NOSSDAV01), Port Jefferson, NY, USA, June 25–26, pp. 11–20. (2001)

Multicast Routing in Structured Overlays and Hybrid Networks

Matthias Wählisch and Thomas C. Schmidt

Abstract Key-based routing has enabled efficient group communication on the application or service middleware layer, stimulated by the need of applications to access multicast. These developments follow a continuous debate about network layer multicast that had lasted for about 30 years history of the Internet. The IP host group model today still faces a strongly divergent state of deployment. In this chapter, we first review the key concepts of multicast and broadcast data distribution on structured overlays. Second, we perform a comprehensive theoretical analysis examining the different distribution trees constructed on top of a key-based routing layer. Characteristic performance measures of the multicast approaches are compared in detail and major structural differences are identified.

Overlay multicast overcomes deployment problems on the price of a performance penalty. Hybrid approaches, which dynamically combine multicast in overlay and underlay, adaptively optimize group communication. We discuss current schemes along with its integration in common multicast routing protocols in the third part of this chapter. Finally, we reconsider and enhance approaches to a common API for group communication, which serves the requirements of data distribution and maintenance for multicast and broadcast on a middleware abstraction layer, and in particular facilitates hybrid multicast schemes.

Matthias Wählisch
HAW Hamburg, Department of Informatik, Berliner Tor 7, 20099 Hamburg, Germany;
Freie Universität Berlin, Institut für Informatik, Takustr. 9, 14195 Berlin, Germany,
e-mail: waehlisch@ieee.org

Thomas C. Schmidt
HAW Hamburg, Department of Informatik, Berliner Tor 7, 20099 Hamburg, Germany,
e-mail: t.schmidt@ieee.org

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_32, © Springer Science+Business Media, LLC 2010

1 Introduction

In recent years the Internet community experienced two significant disruptions. The advent and overwhelming success of Napster and successors from 1999 on demonstrated an imperative desire of Internet users to take advantage of transparent end-to-end application services. The Internet, originally designed as a logical end-to-end overlay on top of heterogeneous physical networks, apparently had failed to serve these needs in its current server-centric and NAT-burdened state of deployment. In the year 2001, when Napster failed legally and early versions of Gnutella broke down technically, proposals for using an abstract name space to combine nodes and content emerged, which organized within Distributed Hash Tables (DHTs).

Structured peer-to-peer systems offer multicast services in an infrastructure-agnostic fashion. They are reasonably efficient and scale over a wide range of group sizes. However, they do not allow for layer 2 interactions and thus do not facilitate unrestricted scaling in shared end system domains. Stability issues for tree-based overlay multicast under churn arise as well, as the departure of branching nodes close to the root may have disastrous effects on data distribution. These drawbacks may be mitigated by hybrid approaches, where overlay multicast routing only takes place among selected nodes which are particularly stable and form a virtual infrastructure. Hybrid multicast schemes inherit major efficiency from the IP layer, while sustaining ease in deployment and infrastructure-transparency from selected group distribution in overlay networks.

In this chapter, we first review the key concepts of multicast and broadcast data distribution on structured overlays in Section 2. In Section 3, a comprehensive theoretical analysis of distribution trees constructed on top of a key-based routing layer analytically derives distributions of key performance measures, e.g., hop counts and the replication loads per node. Current approaches to a multicast deployment in hybrid architectures are presented in Section 4. Section 5 is dedicated to redesigning a common API for group communication, which serves the requirements of data distribution and maintenance for multicast and broadcast on a middleware abstraction layer, and in particular facilitates hybrid multicast schemes. A final discussion concludes this contribution.

2 Overview of Structured Approaches to Group Communication

Derived from structured P2P routing and distributed indexing, several group communication services have been developed as application layer or overlay multicast with the aim of seamless deployability. Among the most popular approaches are multicast on CAN [29], Bayeux [43] as derived from Tapestry and Scribe [8, 9] based on Pastry. These approaches essentially branch in two algorithmic directions. The first uses DHTs to generate a structured sub-overlay network of group members, which thereafter is flooded (CAN, Prefix Flooding). The second class erects distribution trees. Identifying rendezvous points from group ID hashes, Scribe generate shared trees from reverse path forwarding, while Bayeux constructs shortest paths trees from source-specific client subscriptions. Recently, bi-directional shared trees have been introduced by BIDIR-SAM [38].

Group communication protocols should fulfill the following minimal performance requirements:

Coverage The protocol arranges packet delivery at all group members.
Uniqueness From overlay routing, each peer receives a multicast (or broadcast) packet at most once.

The latter prevents protocols from inducing loops, and thus limits the load on peers.

DHT-based multicast performance has been thoroughly studied in [10] with the comparative focus on tree-based and flooding approaches. The separate construction of mini-overlays per group as needed for a selective flooding showed to incur significant overhead. In addition, flooding was found to be outperformed by forwarding along trees, where a shared group tree combined with proximity-aware routing as in Scribe could minimize the overlay delay penalty down to a factor of two.

2.1 Flooding of (Sub-)Overlays

The main idea of flooding approaches is to create a sub-overlay on top of the base overlay network and to assign members of a multicast group to it. Such a group specific overlay can be flooded according to a broadcast scheme derived from the key-based routing protocol in use. No group-specific multicast states are required, as data transmission will solely rely on unicast routing within the sub-overlay. Multicast join and leave procedures are thus identical to joining and leaving a structured overlay network.

Each peer within the dedicated multicast overlay is a multicast source or a receiver. Consequently, forwarding of multicast packets is limited to group members and does not involve arbitrary peers. Depending on the bootstrap mechanism, sub-overlays may be created per group or per source-specific channel. Only in the first case, multicast senders are also required to participate in data reception.

2.1.1 Multicast on CAN

A well-known approach of group-oriented flooding is Multicast on CAN [29]. Based on the Content Addressable Network (CAN) [28, 30], source(s) and receivers of a multicast group G create a so called “mini”-CAN, which is flooded thereafter using the CAN broadcast forwarding algorithms.

For bootstrapping a multicast mini-CAN, the group address is hashed into the d-dimensional CAN identifier space, and serves as identifier of the group-specific bootstrap node. It will be contacted using the base overlay to perform the regular CAN bootstrap mechanism, i.e., identification and assignment of CAN zones.

Possible bootstrap load at a responsible peer can be reduced by varying the group address mapping under multiple hash functions.

A source within the group-specific mini-CAN forwards multicast data to all its neighbors, which then recursively distribute packets within the group-specific mini-overlay. For a detailed description, we refer the reader to the previous chapter. Even though avoided by an optimized forwarding rule, nodes may receive messages more than once. Multicast on CAN, thus, does not comply with the uniqueness constraints.

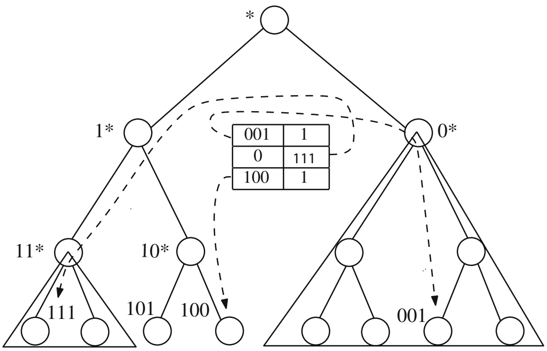
2.1.2 Prefix Flooding

Another flooding approach to multicast is the so called prefix flooding [10, 40], typically implemented on top of Pastry [31]. In contrast to multicast on CAN, prefix flooding distributes data using a (prefix-based) multicast distribution tree, which inherently avoids message duplications. Based on the overlay peer IDs composed of an alphabet with k digits, all N nodes within the sub-overlay can be naturally arranged in a prefix tree, branching recursively at longest common prefix of at most $(k-1)\ln N$ neighboring vertices. Leaves are labeled with overlay identifiers of the peers, while inner vertices represent the shared prefix of their children. If a broadcast packet is sent from the root of the tree towards its leaves, the packet will be replicated wherever prefixes branch. To decide on packet replication, a peer receiving a message is required to determine the current branching position on the distribution tree. This context awareness is achieved by sending packets carrying the prefix currently addressed. IP addresses of prefixes will be resolved according to unicast destinations as provided by the key-based routing layer.

Implementation for Pastry: The idea of prefix flooding naturally corresponds to the prefix routing of Pastry. The Pastry routing table of a peer directly reflects the elements of a prefix tree (see Fig. 1). Every row in a Pastry routing table represents a level of the prefix tree, every column a child of an inner vertex. Each peer holds a subset of the prefix tree in its routing table. Merging the routing tables of all peers from a group specific Pastry network would form the multicast distribution tree, provided the routing tables are complete at each node. The latter can be easily achieved by a proactive routing table maintenance.

Pastry peers flood their routing tables. Thereby they flood the prefix tree, which corresponds to the overlay broadcast in prefix space. In detail, the idea is as follows: A source sends its data to all routing table entries. Each destination prefix corresponds to the root of a sub-tree. The receiving peers determine their position on the tree, i.e., the level of descend D in the prefix tree at which they receive the data, and forward the packets downwards. This is equal to sending data to all routing table entries starting at row $D+1$. Note that the tree position can be encoded in the row number, which reduces the packet size as compared to carrying the entire destination prefix.

The prefix flooding fully complies to the minimal performance requirements.



001	1
0	111
100	1

Fig. 1 Pastry routing table for peer 101 and the corresponding spanning prefix tree using a binary key space. Next hop pointers are highlighted by *dashed lines*

2.1.3 Distributed Tree Construction

Recently, the authors in [3] introduced the Distributed Tree Construction (DTC). The underlying idea is to predefine a geometric area of the overlay space that covers all members of the distribution tree. The source thereafter initiates flooding of this area, which is achieved by means of the key based routing in use. Data is replicated until it hits the border of the designated area. DTC works on the assumption that each peer is informed about its immediate neighbors and the geometry of the overlay space. The authors specify DTC for Chord and CAN.

Relying on the assumption of complete neighborhood sets, the performance requirements are equally fulfilled by DTC.

2.2 Source-Specific Distribution Trees

Source-specific multicast (SSM) approaches construct distribution trees per (S,G) channel, i.e. for a group G *and* a source S. Multicast subscriptions are coupled with sender addresses, which receivers are required to be aware of. Source addresses may be learned out-of-band like the group address G, or autonomously via DHT lookup mechanisms.

The only scheme that follows a pure source-specific distribution model on top of a structured overlay is Bayeux [43]. Bayeux operates on the suffix-based routing scheme Tapestry [41, 42]. The creation of a group proceeds from hashing a source-specific group identifier, which thereafter is used as the name of a trivial file placed at the source node. Using Tapestry location services, the source root of a session announces that dummy document into the entire network. Thereby clients will learn about the group and source ID tuple to perform source-specific subscriptions.

Any receiver subscription (*join* message) is routed to the source via the overlay. The source replies with a *tree* message triggering intermediate peers to set up multicast forwarding states. This mechanism establishes paths from the source to the receivers that are optimal even in the presence of asymmetric paths. To enable tree

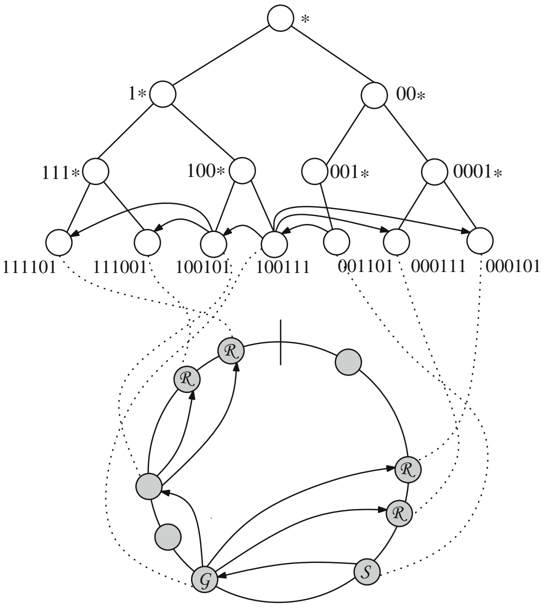


Fig. 2 Scribe: A source $\mathscr{S}$ distributes data to $\mathscr{G}$. The owner of key $\mathscr{G}$ forwards data towards all receivers $\mathscr{R}$

maintenance, the multicast source acts as a centralized group controller that provides a complete receiver tracking. In proceeding along this line, Bayeux admits a linear growth of state information and violates scalability in the number of multicast receivers. A more detailed description of Bayeux is given in the previous chapter.

2.3 Shared Distribution Trees

Shared trees distribute multicast traffic from any source and comply to the general model of Any Source Multicast (ASM). The most prominent example of ASM on the overlay is Scribe [9, 32], which constructs a rendezvous-point-based shared tree in analogy to the native multicast routing protocol PIM-SM [16]. The multicast distribution tree is constructed from receiver subscriptions on reverse paths to the source. Pastry is used as the underlying (unicast) key-based routing, guiding a proximity-aware forwarding. Any multicast source sends data to a dedicated rendezvous point (RP), the root of the distribution tree (cf. Fig. 2). This root node is defined by the hash key of the multicast group ID. Forwarding states are stored in multicast children tables, maintained in parallel to Pastry routing. The detailed tree construction scheme and forwarding algorithm are elaborated in the previous chapter

SplitStream [8], a distributed version of Scribe, allows for load sharing among multiple RPs. The implications of (distributed) rendezvous points on Scribe and SplitStream performance have been examined in [2].

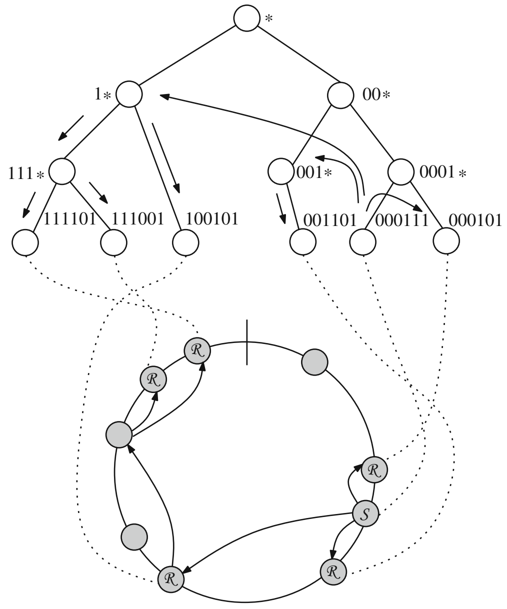


Fig. 3 BIDIR-SAM: A source sends data to prefixes that cover receivers. Prefix-directed routing forwards to nodes that represent the selected prefixes

2.4 Bi-directional Shared Distribution Trees

Bi-directional shared trees span the overlay network and enables source specific data transmission from any overlay node. Protocols that establish spanning trees need to generate multicast forwarding states from a generic group identifier. But in contrast to shared trees, the root of the tree is virtualized and multicast sources are enabled to inject data without utilising a rendezvous point or flooding of sub-overlays.

The concept of a bi-directional shared tree with source-specific shortest paths has been introduced with the BIDIR-SAM [38] protocol. Its main idea is to construct a prefix-based multicast distribution tree, in which leaves are labelled with overlay IDs of the multicast listeners. Multicast branching is performed at inner vertices. Each inner vertex can be mapped to a DHT member, if the label represents a prefix of the overlay node address. Such a prefix is said to be associated with the node. A peer associated with a prefix will be resolved based on a proximity-aware neighbor set as for example provided by Pastry (cf. Fig. 3).

In contrast to the prefix flooding (cf. Section 2.1), multicast forwarding proceeds according to a separate multicast forwarding table, which holds the prefixes of neighbors covering multicast listeners. Every peer may act as a multicast forwarder, potentially serving as an intermediate destination for a prefix it shares. Consequently, a newly arriving multicast receiver needs to announce its (appropriately aggregated) prefix to adjacent overlay nodes. Prefix neighbors will then store the announced prefix in their forwarding tables.

A prefix represents the root of a subtree, which may subsume multiple multicast listeners. Only the first join and last leave has to be propagated beyond this subtree.

Group membership management is thus bound to the smallest subtree covering the joining (or leaving) receiver and further multicast listeners. *Join* or *leave* propagation within subtrees follows the prefix flooding scheme (cf. Section 2.1). The first joining and last leaving receiver of the group need to flood their membership message in the complete (unicast) overlay network, while signaling at intermediate tree levels remains restricted to an exponentially decreasing group of neighbors.

It is worth noting that error resilience of this approach directly follow from the prefix structure. In contrast to other scalable approaches around, inner vertices of the tree are labeled by prefixes and remain valid as long as any node throughout the overlay shares its value. Thus a volatile peer does not change the tree structure, but may initiate an update of the underlying (unicast) key-based routing table. BIDIR-SAM is not vulnerable to multicast tree disaggregation, but as stable as the DHT itself. An arbitrary redundancy can be achieved by sending data to different prefixes in parallel.

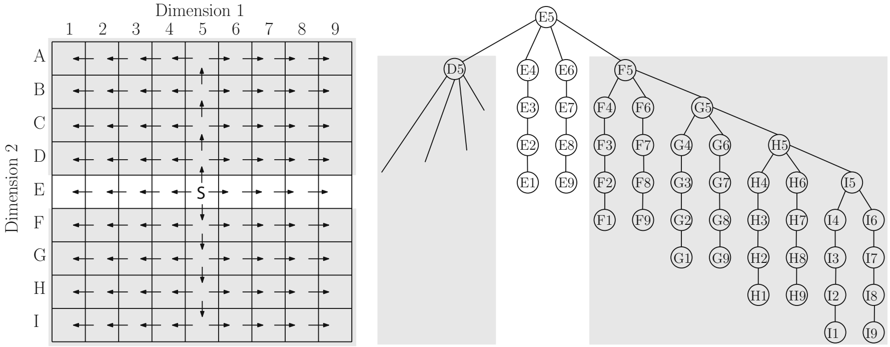
3 Properties of Distribution Trees

All structured approaches to overlay multicast described in the previous section either forward data directly according to a shared or source-specific tree, or data dissemination is well approximated by some tree. The structural properties of these distribution trees characterize the main performance values, such as the branching, i.e., the multicast replication load experienced at end nodes, the hop count distribution, which implicitly assign the delay distribution. In most cases, these measures are accessible by analytical means, thus allowing for a direct, unbiased comparison on arbitrary scales.

The trees employed in the various approaches differ significantly, as will be shown in the remainder of this section. We will introduce structural properties and analytically derive the full probability distribution functions for characteristic quantities, whenever the tree shaping can be isolated from the underlying Internet topology. The latter holds in most cases, in which schemes either delegate topological adaptation to the key-based routing, or remain agnostic of network proximity.

3.1 CAN Flooding: k-ary Trees with Node Chains

Multicast on CAN devises flooding of a D-dimensional hypercube from any origin within the overlay. To avoid duplicate packets, CAN optimizes forwarding with respect to the dimension of packet traversal. As a result, packets traverse along higher dimensions first, while the majority of forwarding hops are conducted in the lowest dimension. For uniformly partitioned hypercubes, this forwarding order leads to a unique data delivery, and is visualized in Fig. 4. Data is then distributed according



(a) The hypercube (b) The corresponding tree

Fig. 4 CAN optimized flooding and its distribution tree

to a tree which is shown correspondingly. This distribution tree admits the shape of a highly unbalanced k-ary tree with $k = 2D$, in which D is the CAN dimension. Branching only occurs at early transits in higher dimensions, but due to dimensional separation, the optimized flooding is gradually led into long linear chains that forward in the lowest dimension. Further we will assume a uniform partitioning, as CAN inherently attempts to approximate uniformly sized zones [29].

We will now derive the distribution of the replication load RPL in a multicast network of N nodes, which is a function of the dimensions a node forwards to. The multicast source issues $2D$ packets to its neighbors in the mini CAN. An inner CAN node receiving a packet from the direction of dimension d will cause a packet replication of $RPL(d) = 1 + 2 \cdot (d-1)$; a boundary node will terminate straightforward transmission and replicate a packet only $RPL(d) = 2 \cdot (d-1)$ times. Forwarding in the highest dimension D will occur at a multiplicity of $\left(\sqrt[D]{N} - 1\right)$, while traversal in lower dimensions d will admit frequencies of $\left(\sqrt[D]{N} - 1\right) \cdot \left(\sqrt[D]{N}\right)^{D-d}$. In dimension d, $\left(2 \cdot \sqrt[D]{N}\right)^{D-d}$ forwarding hops hit boundary zones. Observing that the total number of transmissions is $N = \left(\sqrt[D]{N}\right)^{D}$, and combining these values will lead to the probability distribution (P) of the replication load.

For $1 \leq d \leq D, N > 3^D$,

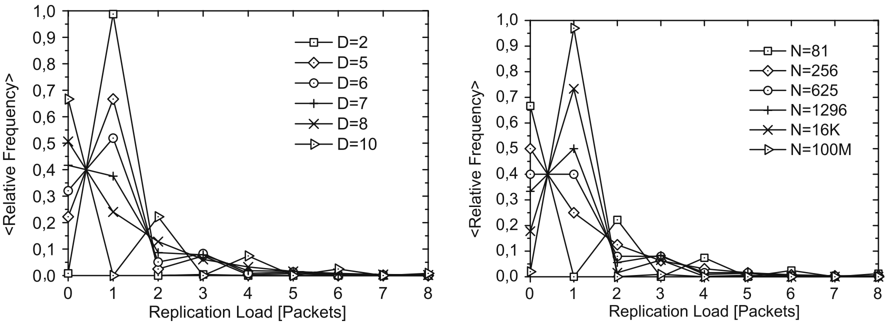
$$P\{RPL(d) = j\} = \begin{cases} \frac{\left(\sqrt[D]{N}-3\right)}{\left(\sqrt[D]{N}\right)^d} & \text{if } j = 2 \cdot d - 1 \\ \frac{2}{\left(\sqrt[D]{N}\right)^d} & \text{if } j = 2 \cdot d - 2 \\ \frac{1}{N} & \text{if } j = 2 \cdot D \end{cases} \tag{1}$$

As N grows large for a fixed dimension, this distribution concentrates at the value $j = 1$, indicating that the forwarding process proceeds according to long, dominant chains. This formally confirms the nature of the distribution tree as shown in the example of Fig. 4. Correspondingly, the mean values of the branching process

$$< RPL(d) >= \frac{(N-1)\cdot\left(\sqrt[D]{N}+1\right)}{N\cdot\left(\sqrt[D]{N}-1\right)} \tag{2}$$

converges rapidly to 1 with increasing network size.

The distribution of equation (1) is visualized in Fig. 5 for different dimensions and network sizes.[1] The actual shapes of curves only depend on the value of $\sqrt[D]{N}$, and admit smooth, evenly distributed weights only for $N = 4^D$. Hence at a given dimension, CAN achieves optimal balance in packet distribution for network sizes close to the minimal number of nodes needed to fill the corresponding hypercube. This observation somewhat contradicts the common believe that CAN serves particularly well for large-scale content distribution.



(a) Varying dimensions, $N = 59049$

(b) Varying network sizes, $D = 4$

Fig. 5 Distribution of CAN replication load for different parameter values, *bold lines* drawn for $N = 4^D$

Second, we take a closer look at hop counts in CAN. If we index a CAN zone $Z_{i_1,\cdots,i_D}$ according to its coordinates and renumber zones such that the multicast source is positioned at the origin, then the number of hops in CAN required to reach a zone Z reads $H(Z_{i_1,\cdots,i_D}) = \sum_{k=1}^{D} |i_k|$. These CAN indices can be considered uniformly distributed independent random variables with $-\lfloor \sqrt[D]{N}/2 \rfloor \leq i_k \leq \lfloor \sqrt[D]{N}/2 \rfloor$, for any CAN node picked at random. It immediately follows that the average number of hops $< H(Z_{i_1,\cdots,i_D}) > = \frac{D}{4}(\sqrt[D]{N})$, the distribution of the hop count in CAN, however, turns out to be surprisingly complex.

Consider $\tilde{\imath}_k$ to be independent variables uniformly distributed on $1,\ldots,\lfloor \sqrt[D]{N}/2 \rfloor$, and $\tilde{I}_K = \sum_{k=0}^{K} \tilde{\imath}_k$. Then $\tilde{I}_K$ differs from $H()$ only by indices of zero values. $\tilde{I}_K$ is the sum of independent identically distributed uniform random variables, admitting a distribution f_k that is the k-fold convolution of the uniform. In detail [15],

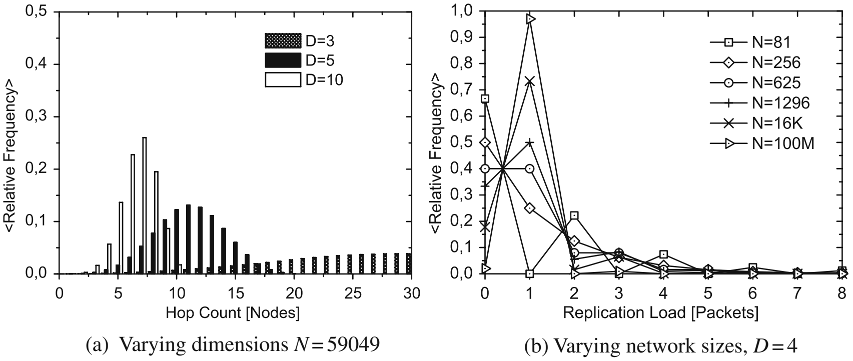
[1] Interconnecting lines are used to enhance visibility.

$$f_k(j) = a^{-k} \sum_{\nu=0}^{\infty} (-1)^{\nu} \binom{k}{\nu} \binom{j - a\nu - 1}{k-1}, \text{ where } a = \lfloor \sqrt[D]{N}/2 \rfloor.$$

The hop count distribution $h(j)$ can be regained from $f_k(j)$ by explicitly accounting for zero values at indices. Let $\mathscr{I}_k$ be the set of all events with exactly k indices equal to zero. Expanding conditionals then yields the hop count distribution

$$\begin{aligned} h(j) &= \sum_{k=0}^{D} h \big\|_{\mathscr{I}_k}(j) \cdot P(\mathscr{I}_k) \\ &= \sum_{k=0}^{D} f_k(j) \cdot \binom{D}{k} \cdot \left(1 - 1/\sqrt[D]{N}\right)^k \left(\sqrt[D]{N}\right)^{k-D}. \end{aligned} \tag{3}$$

This distribution grows remarkably wide. In fact, from the central limit theorem [15] it follows that the dispersion $\sigma(h) \sim \sqrt{D}\,\sqrt[D]{N}$, which increases for a fixed dimension with N. Figure 6, displaying the hop count distribution of equation (3) for different dimensions and network sizes, reflects the divergence of hop count values attained in CAN flooding. Moderate multicast group sizes N already lead to fluctuations of about 100%.



(a) Varying dimensions $N = 59049$ (b) Varying network sizes, $D = 4$

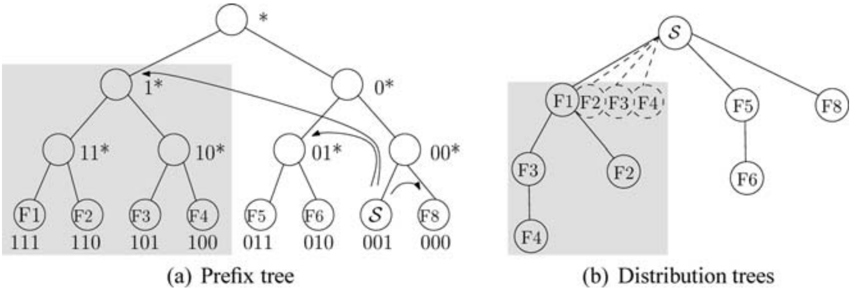
Fig. 6 Hopcount distribution of CAN for different parameter values

Actual delay and jitter values derive from embedding the distribution tree into the underlay. Assuming independent exponentially distributed link weights in the underlay with mean $\frac{1}{\alpha}$, will lead to an overall composite delay distribution at multicast receivers of the Erlang type with parameters $H(.)$, α.[2] Jitter values experienced on data reception will thus be increased by a factor of $\frac{1}{\alpha}$.

[2] A simplified, erroneous derivation of CAN delay properties can be found in [20].

3.2 Prefix-directed Forwarding: k-ary Trees with Node Redundancy

Prefix-directed forwarding may guide the flooding of an overlay network, but may as well be combined with a multicast group management and allow for selective group communication as was shown above for BIDIR-SAM. In both cases, the prefix-based distribution tree forms an abstract structure, which the key-based routing layer uses to instantiate a distribution tree. Commonly, several nodes share a given prefix and thereby serve as possible instances for inner nodes of the actual forwarding tree. This redundancy at inner nodes not only enhances the robustness of the tree-based routing, but may be exploited for a proximity next hop selection. Correspondence options between prefix and distribution trees are displayed in Fig. 7.



(a) Prefix tree

(b) Distribution trees

Fig. 7 Prefix-directed forwarding and possible instances of distribution trees

For a given overlay key space of alphabet size k, we want to derive basic properties of the corresponding k-ary prefix tree. Therein we assume N overlay nodes $\{\mathscr{N}\}$ to be uniformly placed at leaf nodes. In particular, for any node K with key $\mathscr{K}$, the probability of attaining a specific digit x reads

$$P(\delta_i(\mathscr{K}) = x) = \frac{1}{k}, \text{ where } \delta_i(\mathscr{K}) \text{ denotes the } i\text{−th digit of } \mathscr{K}.$$

Consider an arbitrary prefix $\mathscr{C}$ of length j. The probability for a random overlay node to share this prefix equals $(\frac{1}{k})^j$, any (ordered) sequence of keys, l keys with prefix $\mathscr{C}$ and $N-l$ keys not sharing $\mathscr{C}$, occurs with probability $(\frac{1}{k^j})^l(1-\frac{1}{k^j})^{N-l}$. Denoting the longest common prefix by LCP, and accounting for all possible orderings yields the node distribution in prefix space,

$$P\left(|\{\mathscr{N} \in \{\mathscr{N}\} \| \, LCP(\mathscr{C},\mathscr{N}) = \mathscr{C}\}| = l\right) = \binom{N}{l}\left(\frac{1}{k^j}\right)^l \left(1-\frac{1}{k^j}\right)^{N-l}, \quad (4)$$

which is Binomial. The prefix $\mathscr{C}$ of length j correspondents to the root of a subtree T_{h-j} of height $h-j$. Hence equation (4) also describes the distribution of nodes populating subtrees.

The keys representing the overlay nodes span a *random recursive k-ary tree* with *inhomogeneous branching rates*. An inner vertex represented by the prefix $\mathscr{C}$ at level $j-1$ on the path to a given node S with key $\mathscr{S}$, will be a branch of the prefix tree, if a node K with key $\mathscr{K}$ exists, such that $LCP(\mathscr{S},\mathscr{K}) = \mathscr{C}$. The latter is equivalent to the existence of a node that attains a dedicated prefix of length $j-1$, and any of $k-1$ from k values at the j-th digit. The probability that none of the $N-1$ remaining nodes carry a dedicated prefix of length j equals $\left(1-\frac{1}{k^j}\right)^{N-1}$. Hence the branching probability of the overlay prefix structure at level $j-1$ reads

$$P_{Branch}(j-1) = \left(1-\left(1-\frac{1}{k^j}\right)^{N-1}\right)\cdot(k-1) \tag{5}$$

Any multicast group arranges within this overlay prefix structure. Consider a group G of g receivers. We assume that receivers are independently chosen among overlay nodes with the uniform probability $r_g = \frac{g}{N}$.[3] The BIDIR–SAM algorithm aggregates multicast receivers according to longest prefixes.

For a given prefix $\mathscr{C}$, the probability that an arbitrary receiver $\mathscr{G}$ shares $\mathscr{C}$ is fundamental to the problem. In detail, for a multicast group G, the probability that a prefix $\mathscr{C}$ of length j is attained by at least one of the g receivers reads[4]

$$\begin{aligned} &P(|\{\mathscr{G}\in G \| \, LCP(\mathscr{C},\mathscr{G}) = \mathscr{C}\}| \geq 1) \\ &= 1-\left(1-\frac{g}{k^jN}\right)^N = 1-e^{-\frac{g}{k^j}} + \mathscr{O}\left(\frac{1}{N}\right). \end{aligned} \tag{6}$$

It is worth noting that in large networks the prefix distribution of multicast receivers is effectively independent of the network size.

To derive expressions for the replication load, we note that multicast forwarding occurs in combination with a destination prefix of length j, which rules out next hops on the tree of shorter prefix length. Denote by $RPL(j,g)$ the multicast replication load at a node in an overlay participating in BIDIR-SAM with prefix length j. Then

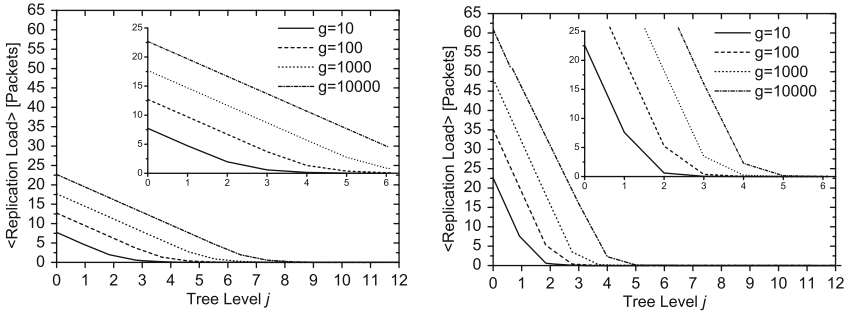
$$<RPL(j,g)>= \sum_{i=j}^{h-1}(k-1)(1-e^{-\frac{g}{k^i}}). \tag{7}$$

Characteristic distributions of the replication load are drawn in Fig. 8, showing a steady decrease in packet replication on the way from source to receivers. Smaller alphabets noticeably smoothen the distributions, which suggests that k serves as a tuning parameter for the multicast distribution tree.

We now want to derive the hop count distribution for sparsely scattered receivers in a prefix tree. On the path from the source to the receivers, a multicast packet traverses an overlay hop, whenever the distribution tree branches at the corresponding prefix $\mathscr{C}$ of length j. Taking the branching rate of equation (5) yields a recurrence relation for the hop frequency with respect to the tree height h:

[3] This assumption is supported in both, theory by [27] and Internet measurements by [11].

[4] For the sake of brevity, we omit proofs, which have been elaborated in [38].



(a) $k=4$

(b) $k=16$

Fig. 8 Mean replication load in BIDIR-SAM as function of the prefix length for different alphabets

$$f_{h,k,N}(j) = f_{h-1,k,N}(j) + \left(1-\left(1-k^{-j}\right)^{N-1}\right)\cdot(k-1)\cdot f_{h-1,k,N}(j-1) \qquad (8)$$
$$\text{with } f_{1,k,N}(0) = 1,\ f_{1,k,N}(1) = \left(1-\left(1-k^{-1}\right)^{N-1}\right)(k-1).$$

Solving equation (8) leads to the hop count distribution $f_{h,k,N}$ attained at prefix routing on N overlay nodes with independent uniformly distributed identifiers:

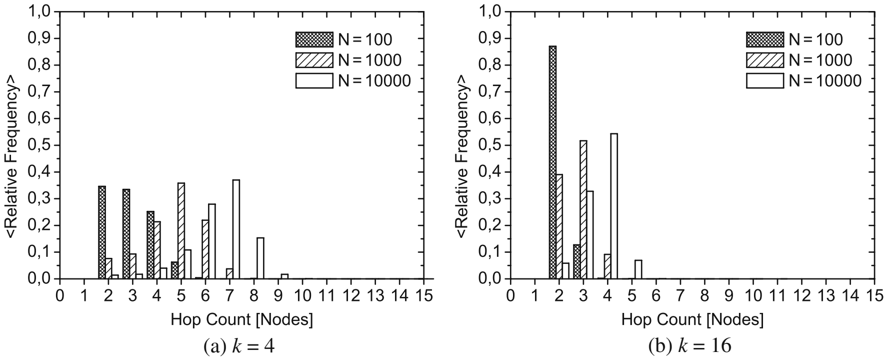
$$f_{h,k,N}(j) = \binom{h}{j}\cdot\prod_{i=0}^{j}\left(1-\left(1-k^{-i}\right)^{N-1}\right)\cdot(k-1)^{j}. \qquad (9)$$

Note that hop counts only depend on the general overlay and remain independent of multicast receivers.

The hop count frequency $f_{h,k,N}(j)$ is plotted in Fig. 9 in normalized form. Mean and width of the distributions grow as k decreases, acting in opposite direction of the branching properties investigated above. A BIDIR-SAM instance optimizing replication and signaling load by using a small prefix alphabet will encounter a moderate increase of routing hops in packet delivery.

3.3 Reverse Path Forwarding: Topology-Induced Trees

Scribe and SplitStream generate distribution trees from forwarding *JOIN* messages on reverse paths from receivers to the rendezvous point(s). This approach is common to multicast routing protocols in the Internet, and suggests a transfer of modeling results for multicast tree structures valid in the Internet, cf. [36]. Despite of this analogy, extensive simulations performed by Birrer and Bustamante, as well as the authors [2, 38] reveal deviant results: The majority of branches tend to directly attach to the rendezvous point(s), generating very high, unbalanced replication loads.



(a) $k = 4$

(b) $k = 16$

Fig. 9 Normalized hop count distributions for $h = 128$, $k = 4$ and $k = 16$

In addition, largely fluctuating delays cause alienating jitter at receivers, which tend to build up in SplitStream.

Observing the distributions of packet delay and replication being incompatible with corresponding Internet measures raises the question about the fundamental differences of data-driven trees in overlay and underlay. The main conceptual distinction follows from the method of tree establishment. Distribution trees generated from reverse paths are shaped according to (reverse) unicast route selection, which is optimal as long as the routing table entries are invertible. Even though routes in the Internet frequently show asymmetries due to BGP routing policies, backbone links can commonly be used in inverse direction without performance penalty.

In DHT-based group communication, however, the direction of tree establishment is important, as routes are commonly highly asymmetric. Next hop neighbors are selected according to a local, node-dependent view of the routing system, and – more importantly – may follow a proximity awareness. As a consequence, paths tend to largely disagree with respect to their direction. As for the example of Scribe, the key-based routing of Pastry associates prefixes to nodes not uniquely, but with respect to node proximity. Two different peers are likely to select a different destination for the same prefix. Thus diverse paths will be established according to underlay vicinity, even though packets could jointly traverse on the same link when following the RP point of view. This results in many divergent, reversely selected paths, which are likely to overlap only at their destination end point. As such, the RP is burdened with an unbalanced replication load. At the same time, first multicast hops, selected reverse oder, are likely to admit the largest distances in the underlay, leading to largely varying hop delays at the root of the distribution tree(s).

4 Hybrid Multicast

Structured peer-to-peer systems offer multicast services in an infrastructure-agnostic fashion. They are reasonably efficient and scale over a wide range of group sizes. However, they neither take advantage of network layer multicast where available, nor allow for layer 2 interactions and thus do not facilitate unrestricted scaling in shared end system domains. Hybrid multicast schemes attempt to combine the potentials of underlay and overlay multicast, targeting at a simultaneous deployment of multicast services on the network layer wherever available, and on the overlay otherwise.

Structured approaches to overlay multicast or broadcast can serve a variety of needs, as has been shown in the previous section. A careful selection of a scheme with respect to deployment objectives allows for an appropriate balancing of data flows, facilitates a trade between state maintenance and path lengths, and may overcome stability issues at inner nodes of the tree structure. However, they show a performance gap between IP and application layer multicast that widens, when mobility is introduced. Frequent handoffs and topological re-arrangements degrade the stability of distribution trees and the efficiency of proximity selection. Garyfalos and Almeroth [18] derived from fairly generic principles efficiency measures for source specific multicast in different metrics. Overlay trees uniformly admitted degradations up to a factor of four over native IP layer multicast in the presence of MIPv6 [22] mobility management. These drawbacks may be mitigated by hybrid approaches, as well, by placing overlay multicast routing among selected nodes, which are particularly stable and form a virtual infrastructure.

Hybrid multicast schemes offer the potential to inherit major efficiency from the IP layer, while sustaining ease in deployment and infrastructure-transparency from selected group distribution in overlay networks. Such approaches differentiate the end-to-end design argument [33] with respect to the inhomogeneous nature of the global Internet: While customer-oriented end system networks, which are mainly built on top of multicast enabled network access technologies, do significantly profit of utilizing network layer multicast services, the flow-oriented transition networks of the Internet core do not.

The design of interconnecting end system domains on the basis of a structured overlay can be transparently implemented by inter-domain multicast gateways (IMGs), leading to the hybrid architecture shown in Fig. 10, and a common group API as introduced in Section 5. It gives full multicast admission control to local operators and may be interpreted as a globally distributed service peering. It will enable inter-domain distribution trees to multicast group services, which remain invisible to the Internet core, while inheriting the full potential of scalability, self-organization, redundancy and error resilience from the overlay network protocols in use. Such adaptive schemes of cooperative routing in underlay and overlay bear the potential to optimise stability and performance, while sustaining ample flexibility for deployment.

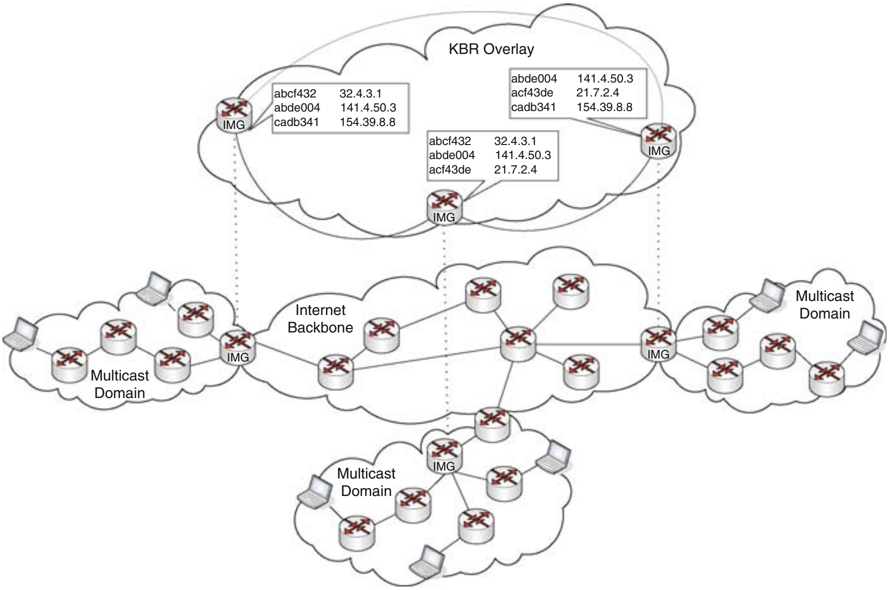


Fig. 10 Inter-domain multicast gateway (IMG) architecture

4.1 Inter-Domain Multicast Tunneling and Overlays

The deployment history of multicast is tightly bound to bridging multicast-agnostic regions of the Internet. From its beginning, the Multicast backbone (Mbone) was built on static IP tunnels, serving as virtual point-to-point interconnects between isolated multicast domains. This inflexible makeshift is currently converted into an adaptive, dynamic Automatic Multicast Tunneling (AMT) scheme [35], which allows for on demand creation and termination of inter-domain multicast encapsulation. AMT provides full support for group receivers, but limits sender support to the Source-specific Multicast (SSM) flavor of multicast.

AMT introduces relays and gateways, designed to cooperate in transmitting multicast traffic via unicast tunnels across multicast-unaware regions of the Internet. Relays receive native group traffic from the domain they service, encapsulate and forward it to corresponding gateways that have joined the relay. Gateways receive multicast traffic as proxies of a multicast-enabled region, a domain or only a single host, and redistribute data natively. Based on IGMPv3/MLDv2 [7, 37] signaling, each relay keeps state in a recipient list for each gateway which has joined a particular group or source-specific channel (source, group). State management may grow linearly in receiver numbers and raises scalability concerns. Unicast tunneling itself remains in conflict with efficient packet distribution and may cause significant link stress.

The lack of group-specific intelligence inherent to the tunneling layer must be considered as the major shortcoming of traditional multicast deployment strategies.

Overlay techniques may overcome this deficit and likewise allow for encapsulation. Conceptually, a structured peer-to-peer multicast overlay can distribute group data from relays to gateways according to *selective* point-to-multipoint paths. It is thereby well suited to replace the inefficient mesh of unicast tunnels commonly deployed.

Combining the AMT gateway discovery mechanisms with structured P2P overlay multicast, Buford [5, 6] introduces such an architecture of enhanced scalability while sustaining transparent interconnects between separate multicast domains. This early work not only attempts to integrate inter-domain multicast tunneling with a variety of structured multicast overlay protocols, but in particular provides a link to activities of the IETF. The generic peer-to-peer protocol of the P2PSIP working group will include a mandatory support of a DHT [21]. Thus, it may be reasonable to assume that DHT substrates will populate the future Internet. These may then also be used as underlying routing infrastructure for multicast protocols.

In the following section, we will work out an architecture which provides transparent support of general multicast eliminating limitations of AMT, and describe its interaction with IP layer group management and routing.

4.2 Hybrid Shared Tree Architecture

The Hybrid Shared Tree (HST) architecture, which is composed of a prefix-based P2P group communication scheme and a relay agent forwarding data between native and overlay multicast, has been introduced in [39]. The core component of the HST is the Inter-domain Multicast Gateway. The corresponding application interacts with native and overlay multicast (OLM) components via the API of an enhanced group communication protocol stack, and thereby enables relaying. In this section, we assume the existence of such a stack and its middleware, both will be described in detail in Section 5.

4.2.1 The Inter-Domain Multicast Gateway

The Inter-domain Multicast Gateway (IMG) transparently forwards multicast data between the overlay and the native network. This gateway will participate in multicast traffic originating from its attached network, which it will forward into the overlay according to the multicast receiver domains of this group. It will also advertise group membership and receive data according to any subscription from its IP multicast domain. With respect to an easy deployment, the IMG should account for current multicast techniques.

The IMG represents a transition point between overlay and underlay. In this role it translates between the different protocols. A structured overlay multicast protocol does not provide any explicit group management to discover the presence of underlay receivers, since applications use direct API calls on the host. An IMG, which may represent a complete multicast domain consisting of multiple receivers and sources, acts in this sense as a proxy: It aggregates and then delegates underlay states to the overlay routing, as well as data originating from underlay sources to the overlay routing.

In this architecture, the multicast overlay serves as the routing backbone, connecting multiple multicast domains. Hence, the construction and destruction of distribution branches will be triggered by the underlay, which includes receivers, sources, both or none of them (cf. Fig. 11). The IMG must be informed about listener and sender activities within the native network by its group communication stack. On the arrival of new multicast parties in the underlay, this will happen as follows:

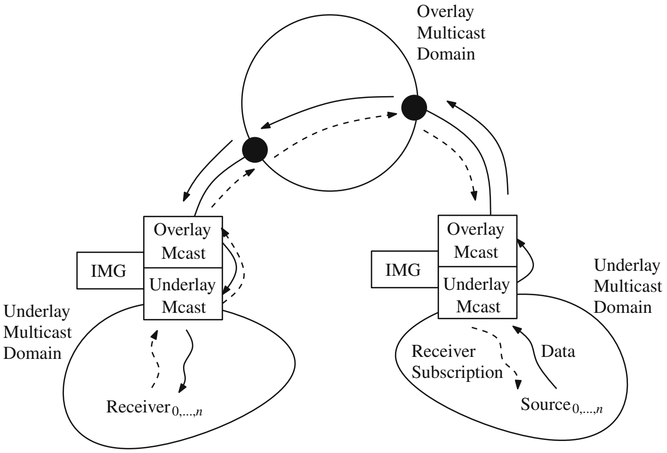


Fig. 11 Schematic view of general inter-domain multicast gateway scenarios

Multicast Source: If the IMG learns about new underlay sources, it immediately sends a corresponding `join` to the underlay group management to receive its traffic. The IMG thereby holds multicast data independent of other receivers including those in different multicast domains. It will send this data internally to the overlay routing instance, which distributes the message with respect to its forwarding states. If there is no subscription in the overlay, the data will be discarded by the routing protocol and not transmitted into the overlay. In contrast, an update about new sources propagated on the overlay takes no effect, as joins are initiated only based on underlay subscriptions.

Multicast Receiver: For each receiver subscribing to a group as the *first* member in the underlay network, the IMG invokes a `join`, processed by the middleware and delegated to the overlay multicast routing protocol. The OLM instance initiates an overlay subscription. Data from the source domain can be transmitted without additional signaling in the underlay, as data is held at the corresponding

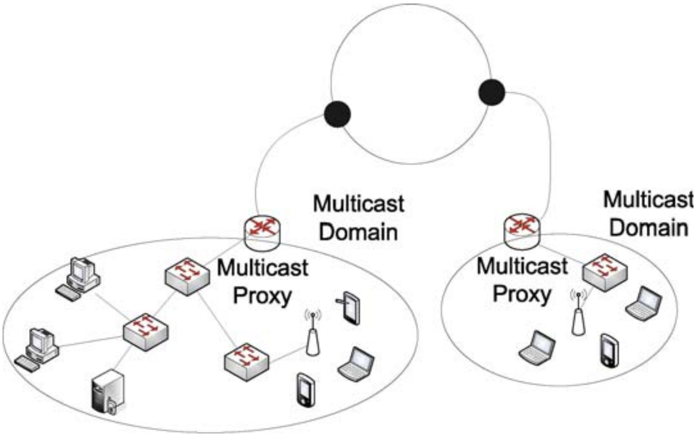


Fig. 12 Two small size multicast domains connected via an overlay

IMG. Multicast streams arriving from the overlay in the receiver domain will be forwarded transparently by the middleware to the IMG, which distributes it into the native network. If the IMG recognizes that the *last* receiver in the underlay network left the multicast group, it has to send a `leave` message into the overlay to cut multicast branches.

In the following, we describe integrations of the protocol concepts in current multicast scenarios. We distinguish between two cases: small size multicast domains without a routing infrastructure, and large size domains which are served by native multicast forwarders.

4.2.2 Connecting Small Size Domains

A small size multicast domain consists of one IP network, two of which are visualized in Fig. 12. Native multicast data communication is supported by group management and routing protocols. Group management is available in IP by IGMP/MLD [7, 37].[5] Implementing MLD signaling represents the minimal requirement for multicast enabled devices and can be assumed in any multicast domain. Multicast receivers send MLD (un-)subscriptions to a standardized group address, such that (potential) routers can track the presence of multicast listeners. The router part of MLD allows to monitor domain-wide group members by a query report scheme. The IMG operates in the MLD router part.

Packets destined to a multicast group address may be broadcasted in switched ethernet domains or selectively forwarded based on MLD snooping operated by domain switches. An MLD snooping-enabled device should transmit group mem-

[5] In the following, we will only refer to MLDv2, omitting IGMPv3 which implements the same scheme for IPv4.

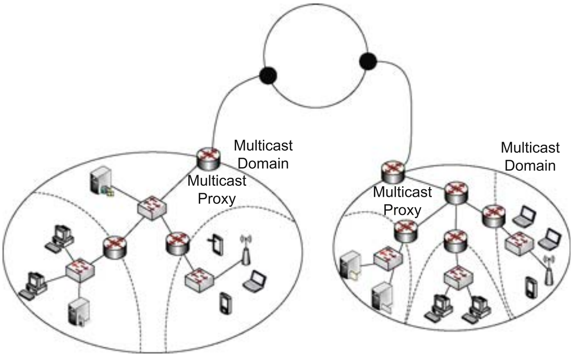


Fig. 13 Two large size multicast domains covering multiple layer 3 networks (*dashed lines*)

bership messages and multicast data only to routers and subscribed receivers, i.e., it implements multicast on layer 2 [12]. Although MLD snooping is not standardized, its deployment is common practice [12]. However, an MLD node running the router part sends periodically group membership queries. This allows not only for learning about active receivers, but also for preventing the suppression of MLD signaling and source data by layer 2 devices.

Based on the `groupSet` call provided by the common group API (see Section 5), the IMG requests the MLD state table, which provides information about active listeners. In particular, the IMG will be informed about the first and last multicast receiver. Correspondingly, the IMG initiates `join` and `leave` calls towards the overlay.

Interconnecting multiple domains without deploying a multicast routing protocol is specified in [17]. The IMG operates in concordance with this standard which is solely based on IGMP/MLD. Running the IMG as MLD proxy easily allows to connect small multicast networks which are not neighboring. On the one hand, this approach reduces deployment complexity as the IMG can be placed all over the multicast domain without obligation to maintain a multicast routing protocol. On the other hand, this scheme limits its scope by the layer 3 region, because MLD signaling is not forwarded across routers.

4.2.3 Connecting Large Size Domains

Connecting multicast islands by an IMG MLD proxy architecture requires a layer 2 access in each LAN. It does not scale to establish a separate proxy in any layer 2 domain of a corresponding larger network. In addition, most larger network domains have established a local host-group routing which provides domain-wide multicast, but fails on global multicast connectivity. In such cases, an IMG should be incorporated into the local routing infrastructure to interconnect larger native multicast islands (cf. Fig. 13).

Like in MLD proxy scenarios, a hybrid multicast gateway must be aware of all groups within a multicast domain to initiate corresponding states in the overlay. In contrast to link-local domains, which can solely be monitored by a group membership protocol, group states are distributed in routed multicast sites. Hence, an IMG requires an interface to the routing infrastructure, where subscriptions occur. The place within the network depends on the multicast routing protocol deployed. In rendezvous point (RP) schemes like PIM-SM, all receiver subscriptions and source data will be registered at the RP. Flooding schemes like DVMRP distribute the information across all neighboring routers while states are shared on a rendezvous link in BIDIR-PIM.

In the following, we sketch methods to integrate the IMG in the two most interesting multicast routing architectures.

PIM-SM

The Protocol Independent Multicast Sparse Mode (PIM-SM) [16] establishes rendezvous points (RP). These entities receive listener and source subscriptions of a domain. To be continuously updated, an IMG has to be co-located with the RP. Whenever PIM register messages are received, the PIM routing instance must signal a new multicast source within the group communication stack. Subsequently, the IMG joins the group and a shared tree between the RP and the sources will be established, which may change to a source specific tree after a sufficient number of data has been delivered. Source traffic will be forwarded to the RP based on the IMG join, even if there are no further receivers in the native multicast domain.

Designated routers of a PIM-domain send receiver subscriptions towards the PIM-SM RP. The reception of such messages invokes an internal update call at the IMG, which initiates a join towards the overlay routing protocol. Overlay multicast data arriving at the IMG will then transparently be forwarded in the underlay network and distributed through the RP instance.

BIDIR-PIM

Bidirectional PIM [19] is a variant of PIM-SM. In contrast to PIM-SM, the protocol pre-establishes bidirectional shared trees per group, connecting multicast sources and receivers. The rendezvous points are virtualized in BIDIR-PIM as an address to identify on-tree directions (up and down). Routers with the best link towards the (virtualized) rendezvous point address are selected as designated forwarders for a link-local domain and represent the actual distribution tree.

The IMG should be placed at the RP-link, where the rendezvous point address is located. As source data in either cases will be transmitted to the rendezvous point address, the BIDIR-PIM instance of the IMG receives the data and can signal new senders towards the stack.

The first receiver subscription for a group within a BIDIR-PIM domain needs to be transmitted to the RP to establish the first branching point. Using an update invocation, an IMG will thereby be informed about group requests from its domain, which are then delegated to the overlay.

5 A Common API for Group Communication

A structured overlay network consists of three functional groups: a routing scheme, a set of services and the applications. The routing, based on a decentralized key approach, is responsible for locating peers associated with specific key ranges. Such routing algorithm is independent of the applications built upon it. Services like group communication supplement the structured overlay and can be developed independently of both, the underlying overlay routing and the application. A well designed protocol architecture should separately account for these components and offer pluggable modular building blocks.

Modeling routing, services and applications in a self consistent way leads to a layered architecture which will attain flexibly interchangeable modules from common interface definitions between all components. Flexibility emerges from two perspectives. Firstly, the development process may be simplified as modules can be reused. Secondly, deployed modules can be combined from tailored units. In the scenario of structured P2P networks, an application can be composed of overlay implementations best suited for current requirements on performance and service needs. This may not only reduce complexity, but also diminish maintenance overhead from unwanted services.

In the following, we will explain the common API for key-based routing, the Dabek model, which usually serves as basis for higher services like group communication. Further on, we present approaches to implement a modular group communication stack merging native with structured overlay multicast and thus facilitating hybrid deployments (cf. Section 4).

5.1 General Design Principles for Structured Overlay Networks – The Dabek Model

The first ideas towards a layered architecture with a common API for structured overlays have been presented by Dabek et al. in [13]. The concept is known as the Dabek model and has been implemented in numerous simulators [1, 4, 24, 26] and DHT stacks [14, 34]. The basis of the Dabek model is a unified overlay routing interface.

Dabek et al. observed that many peer-to-peer services and applications are built upon a key-based routing (KBR) module which can locate peers for multiple purposes. Based there on, the authors suggest a compound P2P layer consisting of three tiers: Tier 0 represents the fundament of overlay communication, the KBR layer. Tier 1 implements higher level abstractions, complemented by tier 2 for end user applications and further, higher level services.

Dabek's model focuses on tier 0. The meaning of tier 1 and 2 has not been fully elaborated. Both tiers may accommodate services with direct access to the KBR layer. Reusable abstractions for specific purposes are dedicated to reside on tier 1.

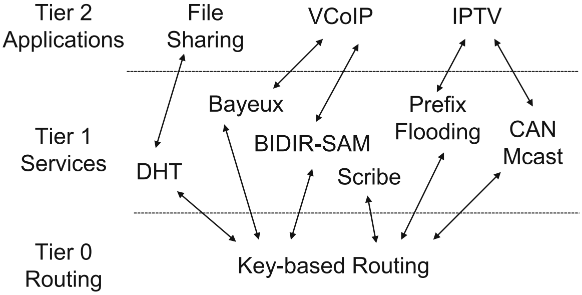


Fig. 14 A compound P2P layer according to the Dabek model [13] with typical application classes. Interactions between the layer components are highlighted by *arrows*

The DHT abstraction[6] is an example for this. This component provides a store-and-retrieval key management service on the overlay for applications like file sharing. On top of the DHT abstraction, further services may be available, such as an additional caching to be used by 'end-user' applications or specific store and retrieval functions. This concept of stacked services may actually result in additional tiers, not foreseen by the model. To avoid such complicacies, we assume that tiers 0 to 2 of the Dabek model reflect a routing, a service and an application layer (cf. Fig. 14).

5.1.1 The Common Key-Based Routing API

The key-based routing layer provides the option to forward data along overlay hops. With respect to the concepts of layered architectures, an implementation may delegate messages to the KBR component that performs the routing. Hence, all application data will be encapsulated in overlay messages and delivered by the KBR. However, this may be inefficient, because it can cause double encapsulation, at the service and the KBR layer. Additionally, a tier 1 service may follow a different routing strategy than the overlay paths. Thus, a common API should remain open with respect to adaptable message forwarding as well as KBR state access.

The KBR API by Dabek et al. consists of two functional groups. The first part provides function calls for sending and receiving data messages above tier 0, and also for controlling the overlay routing path for messages initiated by the service or application. These are subsumed as message routing. The second part allows access to the KBR routing states at peers.

The *message routing* calls offer all required primitives to perform an overlay routing configurable by higher layers. They are summarized in Table 1 with respect to the implementation at the specific layers. In the following, we describe the use of these different functions. Starting at a source peer that calls `route`, data will be forwarded towards the (destination) key by the KBR protocol. This request can

[6] Although, DHTs are equipped with a KBR, the aim of distributed hash tables is the management of key-value-pairs. This is a conceptually different task compared to the KBR.

be supplemented by a predefined first hop, the `hint`. At all intermediate peers between source and destination, the KBR implementation invokes `forward` at the tier above. This upcall informs the application about its intention to forward the message for key `K` to the `nextHopNode`. All passed parameters can be modified. Thus, tier 1 and 2 applications can effectively override the default KBR (overlay) routing behavior by changing the `nextHopNode`. Finally, the message will be provided to the application at the peer responsible for `K` via `deliver`.

Table 1 The KBR message routing API calls implemented at the corresponding layers [13]

Tier	message routing
1-2	`forward(key↔K, msg↔M, nodehandle↔nextHopNode)`
	`deliver(key→K, msg→M)`
0	`route(key→K, msg→M, nodehandle→hint)`

For a detailed description of the *state access* functions, we refer to [13]. An overview is given in Table 2. These calls are limited to *local* operations and *do not* involve communication with other nodes. Applications can invoke these functions from higher tiers to inquire on the local peer routing states. Nevertheless, local lookups will fail whenever the requested information is not locally available. To facilitate a global peer access, the message routing functions can be used.

Table 2 The KBR state access API calls implemented at the corresponding layers [13]

Tier	state access
1-2	`update(nodehandle→n, bool→joined)`
0	`nodehandle [] local_lookup(key→K, int→num, boolean→safe)`
	`nodehandle [] neighborSet(int→num)`
	`nodehandle [] replicaSet(key→k, int→max_rank)`
	`boolean range(nodehandle→N, rank→r, key↔lkey,key←rkey)`

The routing as well as the state access functions require a common `key` parameter that corresponds to the overlay address. Consequently, all applications using one of the primitives must be aware of the hash function in use.

5.1.2 Limitations of the KBR API

The generic approach of the KBR API does not provide information about the actual overlay routing protocol or implementation-specific parameters, like the key length in KBR or the dimension in CAN (cf. Section 2.1). Such meta information are of interest for services which implement cross layering or operate adaptive to the underlying KBR. An example are routing services that are derived from the local state information, but want to construct their own forwarding tables. Using the common API, these services can retrieve generic destination values, but are not enabled to reconstruct the underlying routing structure.

To make the protocol parameters visible to upper tiers, the common API needs extensions. One possible concept would include dynamic maps to present the KBR specific configurations in the form of key value pairs. The corresponding schemes can be implemented by information bases similar to Management Information Bases (MIBs). The implementation of this approach requires only a getter call. The actual parameter set is then defined by the protocol instance.

In using such a rich, compound P2P layer concept, existing key-based routing implementations can be enhanced by new services without changing the KBR component. For this purpose, a new service is only required to implement the common KBR calls. However, more complex services such as overlay multicast need to provide an own common API towards applications.

5.2 Towards a Structured P2P Group Communication API

P2P group communication is strongly driven by the lack of infrastructure support for multicast. Although overlay multicast (OLM) is mainly used as a supplement for expanding native multicast regions, its mediator role between application and network layer may shape and enhance group communication. This is one reason, why an OLM scheme should be implemented in a flexible, common group communication framework which simultaneously accounts for specifics of the overlay and underlay. The core architecture for an enhanced group communication stack is shown in Fig. 15.

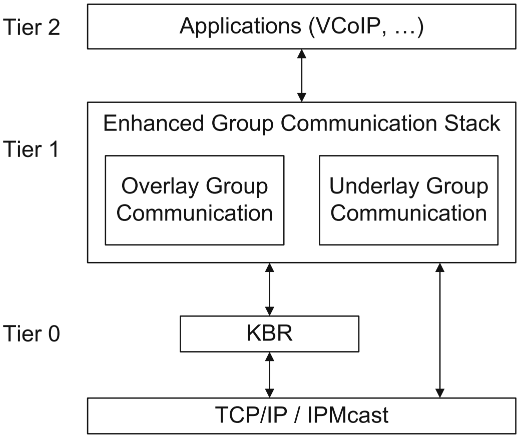

Tier 2
Applications (VCoIP, ...)
Tier 1
Enhanced Group Communication Stack
Overlay Group Communication
Underlay Group Communication
Tier 0
KBR
TCP/IP / IPMcast

Fig. 15 Generic stack architecture for cooperated underlay and overlay multicast

The main task of an OLM component is the distribution of data according to the host group model. Destined for a group address, messages are replicated along an existing (virtual) routing infrastructure similar as in native networks. On the unicast side, routing in structured overlays can be provided by the KBR layer and the access may be decoupled from a specific implementation using the common primitives. Thus, structured overlay group communication requires at least the KBR layer.

The KBR layer connects peers to a unicast network. Multicast domains will be established, when overlay nodes form a group. Group creation and maintenance is operated by the OLM protocol. A special case is broadcast, which inherently floods all nodes of the network. This may happen in contrast to the establishment of selected routing paths. Hence, multicast and broadcast follow different distribution schemes and require a different API towards the application. A modular OLM stack should account for these conceptual different challenges and solutions.

Besides flexibility, an OLM design should also be guided by a compatibility principle with respect to native network services. On the one hand, the API calls, e.g., joining and leaving groups, should be compliant to well-known functions such that application developers are not distracted. On the other hand, the OLM middleware should provide an interface which allows transmission of data to both, overlay and native multicast networks.

Overlay group communication differs from native IP networks with respect to addressing. An IP stack is fed with the correct network address type to perform routing, whereas the key-based routing layer maps an arbitrary identifier to the deployed key space which commonly is not invertible and does not allow to recover original IDs. Applications need to regain those identifiers explicitly, and it may be an advantage for the overlay routing to be aware of the application addresses, as well. Based on application addresses, the overlay scheme may for instance aggregate or scope groups. Thus, the design of an OLM middleware should preserve common group functions, but also include support for specific aspects of the layers involved, and remain open for additional functionalities arising from structured overlay networks.

5.2.1 Current State of the Art

The Dabek model (cf. Section 5.1) proposes the multicast abstraction CAST [13]. The idea is to provide overlay services in a generic multicast module. CAST consists of a set of interfaces for group management and data distribution (`join(groupId)`, `leave(groupId)`, `multicast/anycast(msg, groupId)`) as well as a basic multicast routing on top of the KBR layer. Routing within CAST is built upon a dedicated tree management and forwarding scheme which is similar to Scribe. Calling the `join` function initiates a subscription message routed towards the hash of the group id, employing the KBR `route`. At all intermediate peers, the upcall `forward` is invoked to establish corresponding multicast states in CAST. On top of CAST, the authors have foreseen further multicast implementations, e.g., Scribe or Bayeux.

This approach of a universal routing protocol as part of the middleware layer may conflict with the forwarding strategy pursued by services above CAST. A simple example can be identified in the difference between Scribe [9] and Bayeux [43] (cf.Section 2). While Scribe creates a rendezvous point-based shared tree according to reverse path forwarding, Bayeux sets up source-specific states along the path from the source to the new receiver. Thus for Bayeux, CAST would establish multicast states in the opposite direction. Indeed, the idea of providing the P2P layer with a generic multicast routing logic is valuable for applications agnostic to routing services, but fails in general for overlay multicast modules. Group specific routing forms the core component of overlay multicast, which may change by approach and domain-specific demand. In this sense, a service abstraction should only provide an interface definition, but not a routing logic.

Another application layer multicast (ALM) middleware architecture including a wrapper API is currently presented in [23]. The authors assume that an ALM protocol consists of the following parts: group management, topology management and traffic management, which can be adopted by different ALM protocols. For the latter, they propose an API for interoperability and transparency. The API calls support the selection of different transport protocols as well as a native networks and an overlay transmission mode.

The motivation of the suggested API is to provide a unique interface for supporting structured and unstructured ALM protocols. As mentioned by the authors, the requirements slightly differ which is reflected in the API design. Unstructured group management introduces functions unknown in the context of structured overlay multicast, an external group management for example. Many unstructured multicast schemes rely on central management and provide a global view on the group structure. In general, the simultaneous support of centralized and decentralized approaches poses a severe challenge to a common middleware. Actually, such a cooperative deployment scenario is less likely than KBR protocols jointly operating with network layer multicast, as both are fully distributed and explicitly neglect a server infrastructure.

The authors in [25] present a middleware for unstructured application layer multicast only. They decompose the ALM component in several functional units, e.g., a metric estimator or a logic net to maintain and optimize the overlay network. The proposed API does not account for a transparent overlay and underlay group communication and consists only of simple receive/send calls.

5.2.2 Current Challenges

Current concepts for the implementation of structured application layer multicast [13, 23, 25] deal with the modularization of the OLM/ALM component and a corresponding API definition. The APIs presented either focus on a common tree construction interface or a direct adoption of native group management calls. However, the concepts do not account for two further important aspects:

(a) Which type of addresses may join the application?

(b) How does the OLM API provide dedicated broadcast?

Commonly, a structured overlay network does not restrict the address space to any specific type. Each address will be handled as a string and hashed to the same identifier space without further syntactical or semantical processing. Nevertheless, the overlay may require special addresses for group communication to predefine a subset of group members like a broadcast address. Applications operate in different contexts and denote communication parties with respect to a domain-specific namespace. Special addresses should be available in all namespaces to allow for its continuous use.

Dedicated broadcast can be offered by a structured OLM API, after all applications have joined a specific "all-hosts" multicast group. This obviously does not meet native broadcast, which works without active receiver subscriptions. The problem can easily be fixed by distinguishing broadcast and multicast data. Supplementing the API with an additional broadcast interface may be one simple solution. However, broadcast and multicast operate on the same overlay, which may results in key collisions while hashing identical identifiers, since there is no reserved address space. Consequently, the OLM middleware should foresee a specific, but well-known broadcast address which can also be used to identify broadcast and multicast data on the same channel.

There are two natural options to guarantee that broadcast addresses used by different applications are always mapped on the same overlay key. The API may define a broadcast address which belongs to a specific context, but is obligatory for all applications. Application programmers then have to use this specific address and need to account for context switches, if regular group communication is based on a different namespace.

Alternatively, broadcast addresses should be embedded in every namespace. These multiple, dedicated addresses can then be mapped by the middleware to a common identifier which does not conflict with multicast addresses. For this reason, an OLM should be aware of namespaces, and each namespace should include a unique broadcast address. This can be implemented, e.g., by using the natural 255.255.255.255 for IPv4 or the link-local all-nodes address for the IPv6 namespace. Application layer addresses like SIP URLs can reserve the asterisk for broadcast identification.

Based on the distinction of broadcast and multicast data, a group communication framework can identify data and follow the distribution algorithms implemented. OLM approaches which do not support a dedicated broadcast scheme may assure an all-node reachability by an automatic pre-subscription via multicast.

5.3 Architectural Components

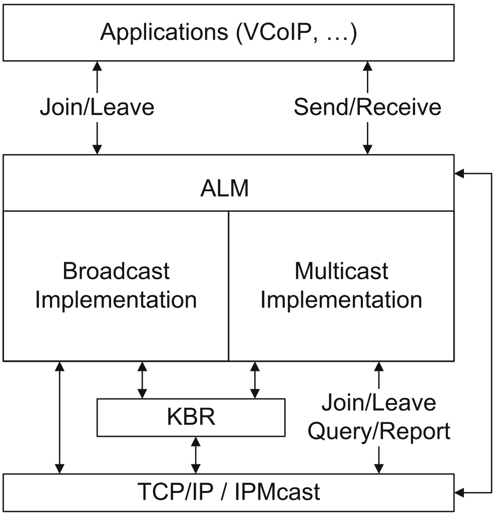
The design goals of an application layer multicast service for structured overlay networks are twofold. On the one hand, the *architecture* for the OLM component itself needs to be defined along with its placement in the global system. On the other hand, a generic *API* for each of the interchangeable modules must be identified.

There is a common sense in the literature to modularize P2P components, even though the perspectives differ. The Dabek model introduces a common 3 tier peer-to-peer stack with a generic routing layer for structured overlays while contributions in the field of ALM protocols focus more on decomposing the ALM service. In the following, we describe the architectural components required for an enhanced group communication stack and the corresponding API.

5.3.1 Architectural Overview

Overlay multicast supplements nodes without a global multicast connectivity with a wide-area group communication service. Thus, it is important to provide a transparent (virtual) network stack to application developers beyond the P2P context.

Fig. 16 An application layer multicast middleware embedded in a P2P stack



The overall architecture is displayed in Fig. 16. The group communication stack consists of a middleware, underlay and overlay multicast modules. The middleware manages data exchange between the application and group services. Depending on availability and application requests, it creates a transparent overlay or a native network communication channel. In addition to common multicast interfaces for applications, the middleware provides a service API reflecting group communication states.

Overlay data will be handled by the broadcast or multicast implementation, depending on the destination address in use. Since broadcast will be delivered without explicit subscription, it is only the multicast implementation that internally provides join and leave calls.

The overlay multicast protocols operate on overlay unicast communication. For this reason they are connected with the key-based routing layer via the common API (see Section 5.1). The KBR can be used twofold: On the one hand, it may operate as a transmission layer delivering data to overlay peers. On the other hand, it provides group protocol implementations with unicast routing states. In the latter case, overlay multicast and broadcast data need not use the KBR layer for transmission, but may sent data directly to peers. Therefore, the multicast and broadcast components need to provide an interface to the IP layer, as well.

5.3.2 An API Proposal

The overlay routing is based on hashed keys. As the applications are unaware of any overlay specifics, the mapping of the underlay destination address to the key space should be performed on the ALM layer as already suggested by Dabek et al. Nevertheless, it is important to preserve the original address, because an application may receive streams directed to different multicast addresses from different receivers via the same communication channel. The IP address noted in the IP header cannot be used, as the root of a key may change due to volatile peers.

As an overlay is used in a specific context, the identifiers selected by the applications are likely to belong to the same namespace. Thus, we suggest to pre-initialize the communication channel between application and group stack with the corresponding context to simplify the API calls.

The destination group address configured by the application is a common application layer or network address and is denoted by `address`. In contrast, overlay IDs are identified as `key`. The application can choose, if it sends the data to the underlay or overlay. We denote the corresponding data type `mode` which also allows to leave the decision at the group communication stack, if the mode is unspecified. In the following, we will describe the API calls used between the group communication stack and the application.

Table 3 The multicast communication API calls implemented at the corresponding layers

Tier	message routing
2	`void receive(address→a, message→msg)`
1	`init(namespace→n)`
	`void join(address→a, mode→m)`
	`void leave(address→a, mode→m)`
	`void send(address→a, mode→m, message→msg)`

At first, we explain calls to function for sending and receiving multicast data, thus, reflecting typical source and receiver instances (cf. Table 3). After the group communication stack is preinitialized with a namespace via `init`, a multicast listener invokes the `join` call for group subscriptions. The join call is implemented by the ALM stack. Depending on the mode, this may result in an IGMP/MLD join, if the address equals a valid IP multicast address. The address of joins towards the overlay will be pre-processed by the middleware to implement, e.g., group aggregation and broadcast. The middleware creates a corresponding overlay key. A peer is able to unsubscribe via `leave`. Multicast data will be accepted via `receive` which is implemented by the application and delivers overlay and underlay messages. The address passed to this upcall represents the destination used by the application instance of the source.

A multicast source can distribute data via `send` implemented at the middleware. If the overlay parameter has been configured, the middleware decides to forward it to the broadcast or multicast module based on the destination address.

To request multicast states, we define the group service calls summarized in Table 4. All registered multicast groups will be returned by the `groupSet`. The result distinguish between sender and listener states. The information can be provided by group management or routing protocols.

Table 4 The multicast state access API calls implemented at the corresponding layers

Tier	message routing
2	`address updateListener(mode→m)`
	`address updateSender(mode→m)`
1	`nodehandle [] groupSet(mode→m)`
	`nodehandle [] neighborSet(mode→m)`
	`bool designatedHost(address→a)`

A routing specific function is the `neighborSet` call which can be invoked at the middleware to get the set of multicast routing neighbors. If the host has the role of a designated forwarder or querier, the `designatedHost` call returns true. Such an information is provided by almost all multicast protocols to handle packet duplication, if multiple multicast instances serve on the same subnet.

Applications on top of the group communication stack can be informed about state changes by corresponding upcalls. The event of a new receiver subscription or leave will be signaled via `updateListener`. Thereupon, the group service may call `groupSet` to pull updated information. A source state change will be notified via `updateSender`.

Based on this set of primitives for multicast communication and state access, hybrid multicast can be implemented according to section 4.

6 Discussions and Conclusions

In this chapter, we presented and analyzed approaches to application layer group communication based on structured overlay networks. These protocol concepts range from flooding of group-specific overlays, over shared and source-specific schemes, to the more recent concept of bi-directional shared trees. Overlay packet distribution in all protocols is established on top of a common key-based unicast routing layer.

Operating in a push model, all schemes under consideration (approximately) distribute data according to a tree. We extracted and discussed key properties of the corresponding tree structures, based on theoretical models for the geometric forwarding in can CAN and for prefix trees, or on current experimental evaluations for data driven trees. Structures were identified to differ significantly, leading to a large variation in performance measures such as replication load, hop counts and delays. Optimized tree construction and data transmission throughout the underlay are key controls for efficient group communication in structured overlays. Corresponding insights into the inherent properties of distribution schemes may serve as an á priori measure in future deployment decisions.

Flooding protocols are more suitable for broadcast scenarios, in which network nodes handle only one group of all network members, and reduce the multicast complexity by omitting group management. Flooding can be naturally achieved on the basis of CAN geometry or the key-based prefix structure. The key advantage of CAN, i.e., strictly limited replication costs for a fixed dimension, where shown to weaken at the price of hop chains growing large with increasing node numbers.

Data driven trees, which do not exploit DHT structures for tree construction, attain minimal group maintenance costs when trees are constructed from reverse paths. However, these shared trees admit degenerate structures and operate at degraded performance, since link selection in the key-based routing layer follows asymmetric rules. As a consequence, sources tend to deliver data along unbalanced trees and suboptimal paths. Such a problem does not arise, if the source constructs its tree according to forward routes, as it is done in source-specific schemes, or for forwarding states that allow for bi-directional traversal. Prefix directed forwarding was shown to be the most balanced approach, complying strict logarithmic bounds in replication load, hop counts and signaling costs. Routes are selected in forward direction, and distribute data along paths that are optimally chosen from the source to destinations.

Overlay multicast is increasingly recognized as a supplement for the current Internet backbone routing, which has not been globally multicast-enabled, yet. Unlike conventional mono-layer solutions, hybrid schemes may augment the native multicast that is well-adopted in enterprise domains by group services on structured overlays, thereby bridging the inter-domain multicast gap. A wide deployment of such approaches requires a seamless integration with current native multicast protocols. This requires a group communication stack to be simultaneously aware of overlay and underlay protocols. Offering such a multi-layer integration to applications requires a common API. In this chapter, we explored both, the ingredients required for a scalable hybrid architecture and the conceptual expressiveness needed at a common group communication programming interface.

6.1 Future Research Directions

From a general perspective, conceptually antithetic approaches compete for serving group communication best: Mesh-based pull models that commonly operate on unstructured P2P routing run in contention with tree-based push models that have implementations on both, structured and unstructured routing layers. Both have been studied since several years. Greedy pull models as used in popular applications like BitTorrent or PPLive are considered to deliver superior, more flexible application performance, while the traditional, tree-based multicast push model minimizes network resource consumption and facilitates an overall optimum for large data distribution tasks.

P2P networks produce an unintentional, but outstanding load on the Internet backbone and at ISP peering points, having led to ongoing debates and traffic blocking. In response, the IETF has started work on application-layer traffic optimizations, paving the way for dynamic adaptation of streams and thus minimizing the P2P load. Efficiency, infrastructure-compliance and reliability of data distribution has thus turned back into focus, forcibly stressed by the increasing success of P2P applications on large scale. Hand in hand with open deployment questions, research is drawn back to analyzing and designing optimized structures for a compliant data dissemination, which is most likely facilitated by protocols based on (multi-) trees.

More recently, approaches rose focus on a merging of overlay and native multicast. Up until now, no clear analysis is available about a most feasible choice of the overlay multicast paradigm, nor about a dedicated protocol best fitting such scenarios. Keeping in mind that ongoing research investigates solutions for re-architecting the Internet, the identification of suitable building blocks for hybrid group communication remains an open, challenging topic.

Global multicast complicates this optimization process as group members may be arbitrarily distributed over several ISPs. At the moment, there has been no research conducted in this promising field.

References

1. Baumgart, I., Heep, B., Krause, S.: OverSim: A Flexible Overlay Network Simulation Framework. In: M. Faloutsos, et al. (eds.) Proceedings of the 10th IEEE Global Internet Symposium, pp. 79–84. IEEE Computer Society, Washington, DC, USA (2007)
2. Birrer, S., Bustamante, F.E.: The Feasibility of DHT-based Streaming Multicast. In: MASCOTS '05: Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 288–298. IEEE Computer Society, Washington, DC, USA (2005). DOI http://dx.doi.org/10.1109/MASCOT.2005.73
3. Bradler, D., Kangasharju, J., Mühlhäuser, M.: Optimally Efficient Prefix Search and Multicast in Structured P2P Networks. Technical Report TUD-CS-2008-103, TU Darmstadt (2008)
4. Brown, A., Kolberg, M.: Tools for Peer-to-Peer Network Simulations. IRTF Internet Draft – work in progress 0, P2PRG (2006)
5. Buford, J.: Hybrid Overlay Multicast Framework. IRTF Internet Draft – work in progress 2, SAM RG (2008)

6. Buford, J.: SAM Overlay Protocol. IRTF Internet Draft – work in progress 1, SAM RG (2008)
7. Cain, B., Deering, S., Kouvelas, I., Fenner, B., Thyagarajan, A.: Internet Group Management Protocol, Version 3. RFC 3376, IETF (2002)
8. Castro, M., Druschel, P., Kermarrec, A.M., Nandi, A., Rowstron, A.I.T., Singh, A.: SplitStream: High-Bandwidth Content Distribution in Cooperative Environments. In: M.F. Kaashoek, I. Stoica (eds.) Peer-to-Peer Systems II. Second International Workshop, IPTPS 2003 Berkeley, CA, USA, February 21–22, 2003 Revised Papers, *LNCS*, vol. 2735, pp. 292–303. Springer–Verlag, Berlin Heidelberg (2003)
9. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.: SCRIBE: A large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in Communications **20**(8), 100–110 (2002)
10. Castro, M., Jones, M.B., Kermarrec, A.M., Rowstron, A., Theimer, M., Wang, H., Wolman, A.: An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer Overlays. In: Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2003), vol. 2, pp. 1510–1520. IEEE Computer Society, Washington, DC, USA (2003)
11. Chalmers, R.C., Almeroth, K.C.: On the topology of multicast trees. IEEE/ACM Trans. Netw. **11**(1), 153–165 (2003). DOI http://dx.doi.org/10.1109/TNET.2002.804835
12. Christensen, M., Kimball, K., Solensky, F.: Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches. RFC 4541, IETF (2006)
13. Dabek, F., Zhao, B.Y., Druschel, P., Kubiatowicz, J., Stoica, I.: Towards a Common API for Structured Peer-to-Peer Overlays. In: M.F. Kaashoek, I. Stoica (eds.) Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22,2003, Revised Papers, *LNCS*, vol. 2735, pp. 33–44. Springer–Verlag, Berlin Heidelberg (2003)
14. Druschel, P., et al.: FreePastry. http://freepastry.rice.edu/FreePastry/ (2008)
15. Feller, W.: An Introduction to Probability Theory and Its Applications, vol. 1, 3rd edition edn. Wiley & Sons, New York (1968)
16. Fenner, B., Handley, M., Holbrook, H., Kouvelas, I.: Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 4601, IETF (2006)
17. Fenner, B., He, H., Haberman, B., Sandick, H.: Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding (”IGMP/MLD Proxying”). RFC 4605, IETF (2006)
18. Garyfalos, A., Almeroth, K.: A Flexible Overlay Architecture for Mobile IPv6 Multicast. IEEE Journal on Selected Areas in Communications **23**(11), 2194–2205 (2005)
19. Handley, M., Kouvelas, I., Speakman, T., Vicisano, L.: Bidirectional Protocol Independent Multicast (BIDIR-PIM). RFC 5015, IETF (2007)
20. Hazarika, S., Towsley, D.: Delay Analysis of Application Level Multicast on Content Addressable Networks. In: Proceedings of Globecom 2004, pp. 1271–1277. IEEE Comm. Soc., Dallas, TX (2004)
21. Jennings, C., Lowekamp, B.B., Rescorla, E., Baset, S.A., Schulzrinne, H.: REsource LOcation And Discovery (RELOAD). IETF Internet Draft – work in progress 00, P2PSIP Working Group (2008)
22. Johnson, D.B., Perkins, C., Arkko, J.: Mobility Support in IPv6. RFC 3775, IETF (2004)
23. Lim, B., Ettikan, K.: ALM API for Topology Management and Network Layer Transparent Multimedia Transport. IRTF Internet Draft – work in progress 0, individual (2008)
24. López, P.G., Pairot, C., Mondéjar, R., Ahulló, J.P., Tejedor, H., Rallo, R.: PlanetSim: A New Overlay Network Simulation Framework. In: T. Gschwind, C. Mascolo (eds.) Proceedings 4th International Workshop on Software Engineering and Middleware (SEM 2004). Revised Selected Papers, *LNCS*, vol. 3437, pp. 123–136. Springer–Verlag, Berlin Heidelberg (2005)
25. Mimura, N., Nakauchi, K., Morikawa, H., Aoyama, T.: RelayCast: A Middleware for Application-level Multicast Services. In: S. Matsuoka, Y. Ishikawa (eds.) Proceedings of the 3st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID ’03), pp. 434–441. IEEE Computer Society, Washington, DC, USA (2003)

26. Naicken, S., Livingston, B., Basu, A., Rodhetbhai, S., Wakeman, I., Chalmers, D.: The State of Peer-to-Peer Simulators and Simulations. SIGCOMM Computer Communication Review **37**(2), 95–98 (2007)
27. Phillips, G., Shenker, S., Tangmunarunkit, H.: Scaling of multicast trees: comments on the chuang-sirbu scaling law. In: SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication, pp. 41–51. ACM Press, New York, NY, USA (1999). DOI http://doi.acm.org/10.1145/316188.316205
28. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A Scalable Content-Addressable Network. In: SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 161–172. ACM, New York, NY, USA (2001)
29. Ratnasamy, S., Handley, M., Karp, R.M., Shenker, S.: Application-Level Multicast Using Content-Addressable Networks. In: J. Crowcroft, M. Hofmann (eds.) Networked Group Communication, Third International COST264 Workshop, NGC 2001, London, UK, November 7-9, 2001, Proceedings, *LNCS*, vol. 2233, pp. 14–29. Springer–Verlag, London, UK (2001)
30. Ratnasamy, S.P.: A Scalable Content-Addressable Network. Ph.D. thesis, University of California, Berkeley (2002)
31. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pp. 329–350 (2001)
32. Rowstron, A., Kermarrec, A.M., Castro, M., Druschel, P.: Scribe: The Design of a Large-Scale and Event Notification Infrastructure. In: J. Crowcroft, M. Hofmann (eds.) Networked Group Communication. Third International COST264 Workshop, NGC 2001. Proceedings, *LNCS*, vol. 2233, pp. 30–43. Springer–Verlag, Berlin Heidelberg (2001)
33. Saltzer, J.H., Reed, D.P., Clark, D.D.: End-to-End Arguments in System Design. ACM Trans. Comput. Syst. **2**(4), 277–288 (1984). DOI http://doi.acm.org/10.1145/357401.357402
34. Shudo, K., Tanaka, Y., Sekiguchi, S.: Overlay Weaver: An Overlay Construction Toolkit. Computer Communications **31**(2), 402–412 (2008). Special issue on foundations of peer-to-peer computing
35. Thaler, D., Talwar, M., Aggarwal, A., Vicisano, L., Pusateri, T.: Automatic IP Multicast Without Explicit Tunnels (AMT). Internet Draft – work in progress 9, IETF (2008)
36. Van Mieghem, P.: Performance Analysis of Communications Networks and Systems. Cambridge University Press, Cambridge, New York (2006)
37. Vida, R., Costa, L.H.M.K.: Multicast Listener Discovery Version 2 (MLDv2) for IPv6. RFC 3810, IETF (2004)
38. Wählisch, M.: Scalable Adaptive Group Communication on Bi-directional Shared Prefix Trees. Technical Report B-08-14, FU Berlin, Mathematik/Informatik (2008). URL `http://www.inf.fu-berlin.de/inst/pubs/index08.html`
39. Wählisch, M., Schmidt, T.C.: Between Underlay and Overlay: On Deployable, Efficient, Mobility-agnostic Group Communication Services. Internet Research **17**(5), 519–534 (2007). Selected papers from the TERENA networking conference 2007
40. Wählisch, M., Schmidt, T.C., Wittenburg, G.: Broadcasting in Prefix Space: P2P Data Dissemination with Predictable Performance. Technical report, Open Archive: arXiv.org (2008). URL `http://arxiv.org/abs/0811.3387`
41. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.D.: Tapestry: A Resilient Global-Scale Overlay for Service Deployment. IEEE Journal on Selected Areas in Communications **22**(1), 41–53 (2004)
42. Zhao, B.Y., Kubiatowicz, J.D., Joseph, A.D.: Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and. Technical Report UCB/CSD-01-1141, University of California at Berkeley (2001)
43. Zhuang, S.Q., Zhao, B.Y., Joseph, A.D., Katz, R.H., Kubiatowicz, J.D.: Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination. In: J. Nieh, H. Schulzrinne (eds.) Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '01), pp. 11–20. ACM, New York, NY, USA (2001)

Multicast and Bulk Lookup in Structured Overlay Networks

Ali Ghodsi

Abstract Structured overlay networks are often used to implement a Distributed Hash Table (DHT) abstraction. In this chapter, we argue that structured overlay networks are suitable for doing efficient group communication. We provide algorithms that enable a node to efficiently broadcast a message to all other nodes in a structured overlay network, without inducing any redundant messages. We also provide algorithms that enable any node to efficiently send a message to all nodes in a specified set of identifiers. Such algorithms have found usage in many structured overlay networks that implement range queries. Similarly, we provide algorithms that enable any node to efficiently send a message to the nodes responsible for any of the identifiers in a specified set of identifiers. Finally, we look at a case study of implementing efficient Application Level Multicast (ALM) using the group communication algorithms on top of structured overlay networks.

1 Introduction

In this chapter, we show how the interconnectivity of the nodes in a structured overlay network can be used for group communication. In other words, we provide algorithms which enable any node to send a message to all nodes on the ring. We also provide algorithms that enable any node to efficiently send a message to all nodes in a specified set of identifiers, e.g., broadcast a message to all nodes with identifiers in the set $\{3,4,21,22\}$. Similarly, we provide algorithms that enable any node to efficiently send a message to the nodes *responsible* for any of the identifiers in a specified set of identifiers.

Ali Ghodsi
SICS, Box 1263, SE-164 29 Kista, Sweden, e-mail: ali@sics.se

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_33, © Springer Science+Business Media, LLC 2010

1.1 Motivation

The lookup operation provided by structured overlay networks is sometimes insufficient for some applications, since it is limited to finding keys that *exactly match* the provided query. The lookup operation does not provide for complex queries containing wild-card expressions, such as "find all items which have a key containing the keyword `music`".

One solution to the above problem is to broadcast a query to all nodes, or some of the nodes, which are part of the structured overlay network. In fact, many popular peer-to-peer applications – such as Skype, Kazaa, and Gnutella – build a network where the nodes are randomly interconnected. Queries are recursively flooded or broadcast to all neighbors. The lack of structure in the random networks, however, leads to disadvantages. For example, some nodes might receive the same message redundantly from different neighbors, wasting bandwidth. Furthermore, the freedom of letting nodes randomly interconnect might lead to the overlay network partitioning into several components which have no interconnection, even though all nodes are connected in the underlay network. Group communication on top of structured overlay networks does not suffer from these disadvantages [6, 7, 15]. On the contrary, when using structured overlay networks, it is possible to guarantee that the time complexity is logarithmic to the number of nodes in the structured overlay network, and that the message complexity is equal to the number of nodes in the structured overlay network.

Group communication can be used as a basic building block to provide *overlay multicast*. Our approach will be to create one structured overlay network per multicast group, and whenever a node requests to multicast information to a group, it broadcasts the information to all the nodes within the structured overlay network that represents the multicast group. We will also show how this scheme can take advantage of IP multicast where it is available, and thus only use the overlay to connect different IP multicast capable networks.

Aside from doing group communication, broadcast algorithms have many uses when constructing structured overlay networks. In particular, a variant of broadcast is useful, whereby a node sends a message to all nodes which have identifiers in a certain interval. Such an algorithm can often be used when implementing range queries, since they often involve reaching all nodes in an interval of the identifier space.

2 Preliminaries

Our goal is to construct general group communication algorithms which can be used on as many structured overlay networks as possible. Hence, we will strive to depend as little as possible on the structure induced by the routing pointers. Our aim is to make our algorithms applicable to any ring-based structured overlay networks. Nevertheless, ring-based structured overlay networks significantly differ in how they place routing pointers. For example, in many structured overlay networks – such as Pastry [32], Tapestry [37] – routing pointers are placed according to prefixes of the identifiers of the nodes. Other structured overlay networks – such as Chord [34], DKS [19], and SkipNet [21] – place pointers according to the relative distance of nodes on the ring. We will disregard such differences and represent the routing pointers uniformly.

Our assumption is that the system is ring-based, i.e., each node has an identifier from an identifier space, $\mathscr{I} = \{0, 1, \cdots, N-1\}$, of a fixed size N, and each node has a pointer to its *predecessor* – the first node met going in clockwise direction on the ring – and *successor* – the first node met going in anti-clockwise direction on the ring. The predecessor and successor pointers are represented by *pred* and *succ*, respectively. Thus, *n.succ* denotes the successor of node n. Furthermore, we represent the routing pointers of each node in monotonically increasing distances. That is, all the pointers of a node n are represented by a function, rt, where $rt(1)$ points to the successor of n, $rt(2)$ points to the node in the routing table of n which is the second closest to n going in clockwise direction starting at n, etcetera. Hence, rt sorts all the pointers of a node n according to their clockwise distance to n, with $rt(1)$ being the successor of the node, and $rt(K)$ being the predecessor of the node, given that the node n has K pointers. The variable K might vary at different nodes and times.

We avoid redundant pointers to simplify our algorithms. For example, in Chord [35], some pointers of a node might be pointing at the same node, i.e., $rt(3) = rt(4) = 10$. Our algorithms will be simplified if all the pointers are unique. Therefore, we represent the pointers by another function u. The successor of a node is always represented by $u(1)$, and $u(2)$ points to the second closest node etcetera, while guaranteeing that $u(i) \neq u(j)$ for any two distinct i and j. We assume that a node has M unique pointers. Again, M might vary at different nodes and times. For convenience, we sometimes use $u(0)$ to denote the identifier of the local node, i.e., $u(0) = n$ at node n, but it is not counted by M.

For the sake of preciseness, we now formally define the above definitions. We will use the notation $x{\oplus}y$ for $(x+y)$ modulo N for all $x,y{\in}\mathscr{I}$, where $N = |\mathscr{I}|$. Similarly, $x{\ominus}y$ is defined as $(x-y)$ modulo N for all $x,y{\in}\mathscr{I}$. For example, if the size of the identifier space is 16, then $15{\oplus}2 = 1$, while $1{\ominus}2 = 15$.

Distances on the identifier space are measured in clockwise direction. Hence, the distance d between any two identifiers x and y is defined as:

$$d(x,y) = y \ominus x$$

The *successor of an identifier* is its closest node in clockwise direction. Hence, the successor S of an identifier x for a set of nodes $\mathscr{P}$ is defined as:

$$S(x) = x \oplus \min\{d(x,y) \mid y \in \mathscr{P}\}$$

The *successor of a node p* is therefore defined by the function *succ* at node p as:

$$succ = S(p \oplus 1)$$

The above function u can be formally defined as follows. Let the set R contain all the pointers of a node n, i.e., $R = \{rt(k) \mid 1 \leq k \leq K\} \cup \{n\}$. We define the *local successor function* at a node as:

$$s(i) = i \oplus min(\{j \ominus i \mid j \in R\})$$

The function u at node n can now be defined for the domain $[0, |R|-1]$ as :

$$u(i) = \begin{cases} n & \text{if } i = 0 \\ s(u(i-1) \oplus 1) & \text{otherwise} \end{cases} \tag{1}$$

We will initially assume that all pointers are correct and no failures occur.

Interval Notation

We now introduce some notation to make our discussions about the identifiers and intervals on the ring more precise. The whole identifier space can be represented by an interval of the form $[x,x)$ or $(x,x]$ for an arbitrary $x \in \mathscr{I}$, where the start of an interval is excluding the first identifier if the left bracket is round, (, and it is including the first identifier if it is square, [. Similarly, the end of an interval is including the last identifier if the right bracket is square,], and excluding the last identifier if it is round,). For any $x \in \mathscr{I}$, we note that $[x,x] = \{x\}$ and $(x,x) = \mathscr{I} \backslash \{x\}$. Hence, a node n is responsible for $(n.pred, n]$. For example, if the size of the identifier space is 16, then $(2,10]$ is the set of identifiers $3, 4, \cdots, 9, 10$. The interval $(10,2]$ is equivalent to the identifiers $0, 1, 2$, and $11, 12, 13, 14, 15$.

Sets of Identifiers

We now connect the interval notation to a set representation. Often a notation of the sort $(i, j]$ is used to represent intervals of the identifier space. Such an interval is a compact representation of a set of identifiers. For example, in an identifier space of size 16, the interval $(14,3]$ represents the set of identifiers $\{15, 0, 1, 2, 3\}$. It is therefore possible to apply the operations available for sets on intervals, such as taking the union or intersection of two intervals. For example, the interval $[11,15]$ represents the set of identifiers $\{11, 12, 13, 14, 15\}$. Therefore, the union of the intervals, $(14,3] \cup [11,15]$, is the set of identifiers $\{0, 1, 2, 3, 11, 12, 13, 14, 15\}$. Similarly, the intersection of the intervals, $(14,3] \cap [11,15]$, is the set of identifiers $\{15\}$. It might, of course, not be possible to represent a set of identifiers as a single interval.

In our algorithms, we make extensive use of the basic set operations on intervals. The reason for this is that the semantics of the operations are well defined. In a system implementation, these can be implemented and optimized as fit.

Resource Consumption

We use *message complexity* as a measure of resource consumption. The message complexity of an algorithm is the total number of messages exchanged by the algorithm. Sometimes, the message complexity does not convey the real communication overhead of an algorithm, as the size of the messages is not taken into account. Hence, on a few occasions, we use *bit complexity* to measure the total number of bits used in the messages by some algorithm.

Time complexity will be used to measure the time consumption of an algorithm. We assume that the transmission time takes *at most* one time unit and all other operations take zero time units. The worst case time complexity is often the same if we assume that the transmission of a message takes *exactly* one time unit, but for some algorithms the worst time complexity increases if we assume that the time it takes to send a message takes *at most* one time unit.

2.1 Desirable Properties

The group communication algorithms we present in this chapter share the following correctness properties:

- *Termination*. The algorithm eventually terminates.
- *Coverage*. All the *designated* nodes that are *reachable* from the initiation to the termination of the algorithm should receive the message.
- *Non-redundancy*. Each node receives the message at most once.

The two first properties are liveness properties, while the last property is a safety property. The last two properties bear some more discussion. We discuss the terms *designated* and *reachable*, as it comes to the coverage property.

The particular group communication algorithm determines what constitutes a designated node. For example, in a broadcast algorithm all nodes are designated nodes. In some other algorithms, perhaps only a subset of the nodes are considered, e.g., all nodes with identifiers in a certain interval.

The nodes that can be visited by traversing the successor pointers, starting at some initiator and stopping whenever the initiator is reached or overshot,[1] are regarded as nodes reachable from that initiator. We assume that the pointers of any node, as defined by the function u, always point at some node which is reachable from the initiator.

The reason that the coverage property is technically involved is to avoid idiosyncrasies that are theoretically possible in a ring-based structured overlay network. For example, one can construct a *loopy* ring, where the state of the successor and predecessor pointers seems correct from the perspective of any two neighboring nodes, but the overall ring structure is inconsistent. Figure 1 shows a loopy ring, where

[1] We say that a node p overshoots an identifier i if p routes to j when $d(p,i) \leq d(p,j)$.

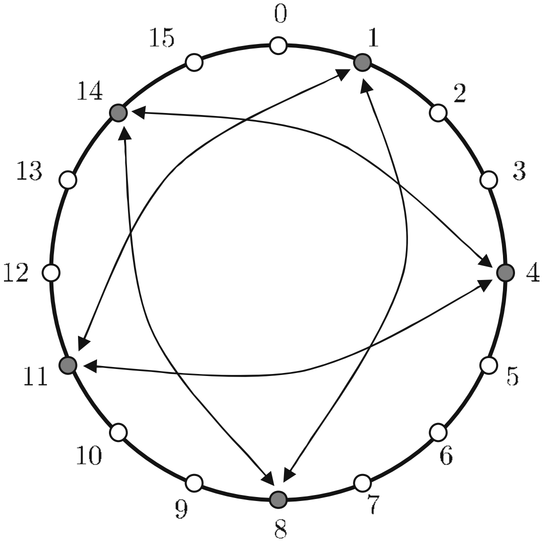


Fig. 1 A loopy network where $(u.succ).pred = u$ for every node u, but for every node u there is a node v between u and $u.succ$

pointers between the successor and a predecessor of a node are symmetric, i.e., for every node u, u's successor has a predecessor pointer pointing at u. However, between any node and its successor there is another node. Consequently, if the ring is traversed starting at some initial node and stopping whenever the initial node is reached or overshot, not all the nodes in the system have been visited. It is not clear how such a loopy network could occur. There are algorithms for making such ring states consistent [26, 27], but it is not in the scope of our group communication algorithms to deal with such scenarios.

By the last correctness property, non-redundancy, we mean that the same message should never be transmitted more than once to the same node. This is different from non-redundant *delivery* of messages, which is adopted by others [23, pg 33f]. The latter is achieved by associating a globally unique identifier with every invocation of the algorithm, and filtering any message with previously seen identifiers. Thus, the application is delivered the message at most once, even though the node might receive the same message multiple times. This comes at the cost of transmitting redundant messages, and maintaining state associated with previously received messages.

3 Broadcast

In this section we provide algorithms for broadcasting to all nodes in the structured overlay network. Hence, every node is considered to be a designated node by the algorithm.

A naïve broadcast algorithm would start at the initiating node and traverse the successor pointers, ensuring that all nodes receive the message. The algorithm would terminate whenever the successor pointer of a node points to the initiator or overshoots the initiator. In other words, if p initiates the algorithm, a node q would only forward the request to its successor if $q.succ \in (q,p)$.

The naïve algorithm is correct. The algorithm would eventually terminate, even in the presence of churn, as it is impossible to keep forwarding the message to a successor infinitely without ever using a successor pointer which points at, or overshoots, the initiator. The algorithm would cover all reachable nodes that remain in the system until the algorithm terminates. No node would ever get the message more than once, as the condition for forwarding ensures that nodes that previously received the message are not forwarded to.

The algorithm would, however, incur a message complexity of $O(n)$ and a time complexity of $O(n)$, where n is the number of nodes in the system.

3.1 Simple Broadcast

The naïve algorithm can be improved. Assume the identifier space is $\{0,\cdots,1023\}$, and only nodes with odd identifiers exists, i.e., 1, 3, 5, $\cdots$, 1023. Assume node 1 is to broadcast a message, and it happens to have node 511 in its routing table. A simple improvement of the naïve algorithm would be to let the initiator 1 *delegate* responsibility to node 511 to traverse the successor pointers in the range 512 to 0 on the ring, while 1 itself traverses the successor pointers in the range 2 to 510. Our message complexity will still be n, but our time complexity will be halved to $\frac{n}{2}$. It is obvious that all nodes in the system would be covered, and that no redundant messages would be sent. Our *simple broadcast algorithm* applies the idea of delegation recursively for every pointer that points to a node that has not previously been sent to.

The simple broadcast algorithm, shown by Algorithm 19, works as follows. The initiating node partitions the identifier space into M parts, and delegates to each routing neighbor the responsibility to cover all nodes in its part. More concretely, the initiating node i delegates responsibility to node $u(M)$ to cover all nodes in the interval $(u(M),i)$, node $u(M-1)$ is delegated responsibility to cover $(u(M-1),u(M))$, etcetera, down to node $u(1)$ (successor of i) to cover nodes in $(u(1),u(2))$. Each delegatee further partitions the part to which it has been delegated into pieces which it further delegates to other nodes. This is recursively repeated until no node has any routing pointers left in the interval it has been delegated, whereby the algorithm terminates.

We now argue why the simple broadcast algorithm is correct.

Termination. The algorithm only forwards a message to a node $u(i)$ if $u(i)$ is in the interval $(n,limit)$. Hence, the beginning of the interval strictly increases (modulo arithmetic) toward *limit* each time a message is forwarded. Similarly, the end of the interval ($limit$), either stays the same or decreases toward n. Hence, each time a node

Algorithm 19: Simple broadcast algorithm

```
receipt of StartSimpleBcast($msg$) from $app$ at $n$
    sendto $n$ : SimpleBcast($msg$, $n$)
end receipt event

receipt of SimpleBcast($msg$, $limit$) from $m$ at $n$
    Deliver($msg$)
    for $i := M$ downto $1$ do
        if $u(i) \in (n, limit)$ then
            sendto $u(i)$ : SimpleBcast($msg$, $limit$)
            $limit := u(i)$
        end if
    end for
end receipt event
```

sends a message to some other node, it delegates to it a strict subset of the interval that itself was responsible for. Since the intervals are discrete and have a finite size, eventually there is either no node in the interval or the interval is empty. Hence, eventually the algorithm terminates.

Non-redundancy. We have shown that each node delegates a subset of its own interval to any node it forwards to. Hence, a tree is induced by the broadcast, where the initiator is the root of the tree, and where a node is the immediate parent of all the nodes it directly sends a message to. We show that every node delegates non-overlapping intervals to all its children. This is is a consequence of the fact that the pointers at any given node are unique and a node never sends a message to itself, since 0 (and consequently $u(0)$) is not covered by the index i in the for-loop. Hence, every node n is delegated responsibility to cover an interval $(n, limit)$, and n sends a message to any neighbor in that interval, delegating each of them non-overlapping subsets of $(n, limit)$. Furthermore, the interval delegated to a node by a node n never contains the identifier of any of n's children.

By structural induction on the broadcast tree, the same node cannot occur more than once in the subtree of any node. The statement is true for the base case, which is any tree containing only one node. For the induction step, assume the hypothesis is true for the subtrees, $T_1, \cdots, T_n$ that are formed by an arbitrary node p's respective children. Then it is also true for the subtree formed at p, since the intervals of each tree is non-overlapping and p does not occur in any of those intervals. Hence, no node ever receives the same message twice.

Coverage. By structural induction on the broadcast tree, the subtree at a node p covers all the nodes that have an identifier in that interval. It is true for the base case, which is any tree containing only one node p. Since the interval given to p is of the form $(p, limit)$, no node in the system can have an identifier in that interval, otherwise one of them would be p's successor $u(1)$, contradicting that p is the only node. For the induction step, assume the hypothesis is true for the subtrees of p's respective children. Then all the intervals delegated to p's children have been covered, which only leaves the identifiers of p's children to be covered for p to ensure that its delegated interval is covered. But those identifiers are covered since p directly sends a message to its children. Hence, p and its children will cover all the nodes in the interval delegated to p. Since the initiator starts with the whole identifier space as its delegated interval, all nodes will be covered.

The simple broadcast algorithm is in fact broadcasting over the k-ary tree [1], which systems such as Chord, Pastry, and DKS use. Consequently, the time complexity of the algorithm is $\log_k(n)$, for n nodes, given that the pointers are selected according to the k-ary tree principle [1]. For example, in Chord, the time complexity will be $\log_2(n)$. Furthermore, the message complexity is n, since all nodes receive the message and no node receives the message more than once. Most importantly, no single node ever needs to send a message to more than M nodes, given that it has pointers to M nodes.

Figure 2 shows the example of a ring and Fig. 3 shows the how a simple broadcast would disseminate over that ring if a broadcast was initiated by node 1.

3.2 Simple Broadcast with Feedback

The operation provided by the simple broadcast algorithm is useful for some applications, such as overlay multicast or publish/subscribe systems. It is, however, not sufficient if the broadcasting node wishes to receive a feedback or a response from the nodes it is broadcasting to. This is especially the case if broadcasting is used to implement arbitrary or complex queries. A naïve solution would be to inform all nodes of the identity of the initiator, and let them directly send their feedback to the initiator. But this is not scalable as the initiator quickly becomes a bottleneck as the number of nodes becomes large.

The *Simple Broadcast with Feedback Algorithm* (Algorithm 20) efficiently collects responses from all nodes after broadcasting. It extends the simple broadcast algorithm by letting each node maintain a set, *Ack*, of all nodes that it has sent the message to. Each node also has a variable *par*, pointing to the parent, from which it received the broadcast message. Every node waits to receive a response from each of the nodes in its *Ack* set before it sends its own response together with the gathered responses to *par*. Naturally, nodes that are delegated an interval in which they have no neighbors, can immediately send their response to *par* as they do not need to wait on a response from any node. As a side note, these will be the nodes that are the leafs of the induced broadcast tree.

The feedback algorithm has twice the time and message complexity as the simple broadcast algorithm.

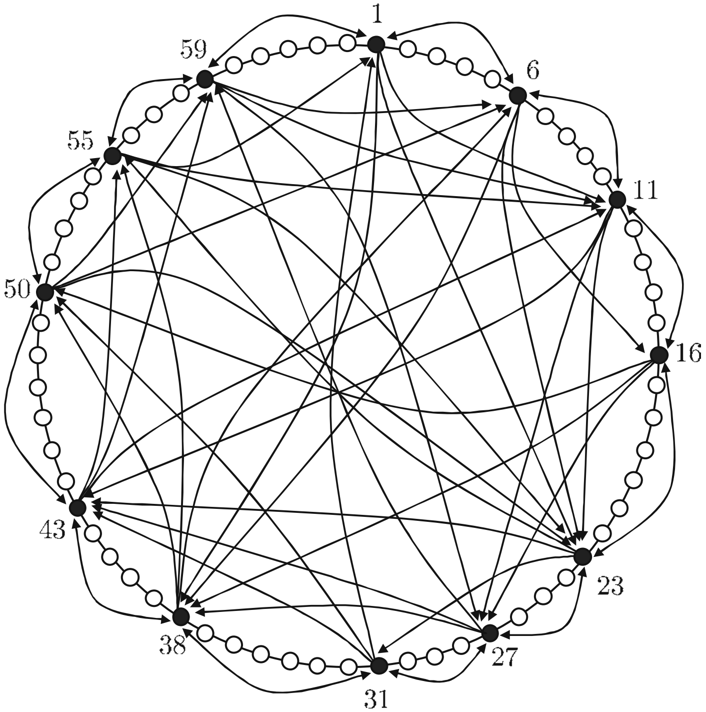


Fig. 2 The figure shows an identifier space $\{0,\cdots,63\}$ and 12 nodes with identifiers $1,6,11,16,23,27,31,38,43,50,55$, and 59. Single arrowed lines represent a routing pointer and each double arrowed line represents a node's predecessor pointer and the corresponding successor pointer

4 Bulk Operations

In this section we generalize the problem and provide two operations: *bulk operation* and *bulk owner operation*. Bulk operation takes a set of identifiers I, referred to as the *bulk set*, and sends a message to all nodes that have identifiers in the set I. Hence, the designated nodes are all nodes with an identifier in the bulk set I. Bulk owner operation takes a set of identifiers I and broadcasts to the nodes that are *responsible* for the identifiers in the set I. The notion of responsibility is the same as in Chord, where every node n is responsible for all identifiers between its predecessor and itself, i.e., the identifiers $(n.pred, n]$. Hence, the designated nodes are those which are responsible for an identifier in the bulk set I.

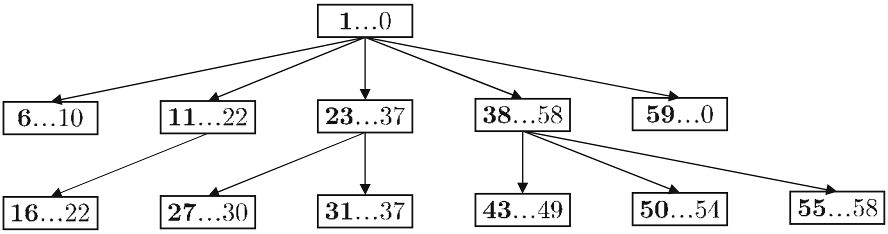


Fig. 3 The figure shows how a simple broadcast initiated by node 1 would disseminate in the system depicted by Fig. 2. Each box represents the node receiving the broadcast message (*in bold*) and the interval that it is delegated responsibility to cover

Naïve Solution

The simple broadcast algorithms presented in the previous section can easily be modified to only broadcast to parts of the ring. The initiating node i can set $limit$ to any identifier, and the simple broadcast algorithm will ensure that only nodes in the range $[i, limit)$ get the broadcast message.

It would be desirable if an initiating node i could broadcast to any node in an arbitrary interval $(k, m]$. This could naïvely be achieved if the initiating node first routes an ordinary lookup message to the successor of k, which then broadcasts to everyone in $(k, m]$. There is, however, a more efficient solution which we describe next.

Motivation

A main motivation for bulk operation is that it can be used to build a *bulk lookup* or *bulk insert* operation. In many applications, such as file systems built on top of structured overlay networks, it is desirable to lookup many keys in parallel. We have encountered file systems built on top of structured overlay networks, in which thousands of simultaneous lookups were issued to fetch a large file [3, 33]. In such cases, there is a significant overhead induced by marshaling and sending thousands of lookups. With the bulk owner operation, the identifiers of all those keys would be described by the bulk set I, which is then used to do parallel lookups. The advantage of the bulk operation is that it guarantees that a node will send at most as many messages as it has pointers, which is often $O(\log n)$ pointers in an n node system. Furthermore, if the routing pointers are placed according to the k-ary principle, such as in Chord, the worst case time complexity of the whole operation is $O(\log n)$, in an n node system. Bulk operation improves the bit complexity when compared to making parallel lookups for every identifier. Hence, it does not trade bit complexity for message complexity by sending fewer, but larger messages. The reason for this is

Algorithm 20: Simple broadcast with feedback algorithm

```
receipt of StartBcast($msg$) from $app$ at $n$
    sendto $n$ : Bcast($msg$, $n$)
end receipt event

receipt of Bcast($msg$, $limit$) from $m$ at $n$
    $FB$ := Deliver($msg$)
    $par := n$
    $Ack := \emptyset$
    for $i := M$ downto 1 do
        if $u(i) \in (n, limit)$ then
            sendto $u(i)$ : Bcast($msg$, $limit$)
            $Ack := Ack \cup \{u(i)\}$
            $limit := u(i)$
        end if
    end for
    if $Ack = \emptyset$ then
        sendto $par$ : BcastResp($FB$)
    end if
end receipt event

receipt of BcastResp($F$) from $m$ at $n$
    if $m = n$ then
        sendto $app$ : BcastTerm($FB$)
    else
        $Ack := Ack - \{m\}$
        $FB := FB \cup F$
        if $Ack = \emptyset$ then
        sendto $par$ : BcastResp($FB$)
        end if
    end if
end receipt event
```

that multiple lookups often end up sending the same message several times between the same pair of nodes.

The bulk operation algorithm has two extreme cases. If the whole identifier space is used as the bulk set I, it reduces to the simple broadcast described in the previous section. If only one identifier (singleton) is used as the bulk set I, the algorithm reduces to a simple lookup. We therefore think that the bulk operation should be the basic operation provided in structured overlay network APIs.

4.1 Bulk Operations

The bulk operation algorithm is given by Algorithm 21. It is very similar to the simple broadcast algorithm (Algorithm 19). The algorithm is initiated by sending a BULK message with two parameters. The first parameter I is a set of identifiers, while the second parameter *msg* is the message to be sent to all nodes with identifiers in I.

One major difference between the bulk algorithm and the broadcast algorithm is that in the broadcast algorithm, every node that received a message also delivered it to the application, while in the bulk algorithm some nodes might be acting as forwarders, which never deliver messages to the application. This is necessary and can be seen if the bulk algorithm is used with a singleton containing one identifier. Then it might be that the only way for the algorithm to reach its destination is to route through other nodes.

Another difference with the simple broadcast algorithm is that it only sends a message to a node $u(i)$ if $u(i)$ has an identifier in I, or if there are identifiers in I which are between $u(i)$ and $u(i+1)$, in which case $u(i)$ is the closest preceding node which can forward the request closer to any potential nodes in that range. This is implemented by creating a set of identifiers $J := [u(i), limit)$, and only sending a message to $u(i)$ if the intersection of I and J is non-empty. When sending a message to $u(i)$, only identifiers in the intersection of I and J are delegated to $u(i)$.

Whenever an interval $I \cap J$ is delegated to some node $u(i)$, that interval is removed from I to ensure that no two nodes are delegated overlapping intervals.

Algorithm 21: Bulk operation algorithm

```
receipt of BULK($I$, $msg$) from $m$ at $n$
    if $n \in I$ then
        Deliver($msg$)
    end if
    $limit := n$
    for $i := M$ downto 1 do
        $J := [u(i), limit)$
        if $I \cap J \neq \emptyset$ then
            sendto $u(i)$ : BULK($I \cap J$, $msg$)
            $I := I - J$
            $limit := u(i)$
        end if
    end for
end receipt event
```

Figure 4 shows an example of how the bulk message disseminates in the system depicted by Fig. 2. Node 1 initiates the algorithm, and wishes to broadcast to all nodes in the interval $[30,45]$. Note that a bulk message is sent to node 27 with the responsibility of covering the interval $[30,30]$. This might seem unnecessary as node 27 is only a forwarder that does not deliver the message to the application layer,

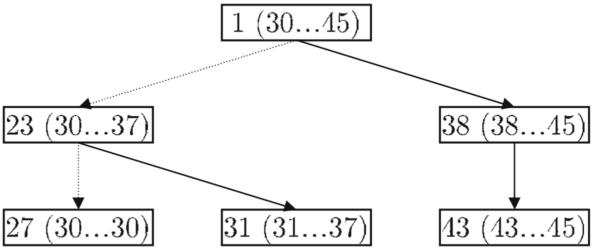


Fig. 4 The figure shows how a bulk message would disseminate in the system depicted by Fig. 2. The bulk operation is initiated by node 1, who wishes to send a message to all nodes in the interval $[30,45]$. Each box represents the node receiving the broadcast message and in parenthesis the interval that it is delegated responsibility to cover. *Dotted arrows* indicate that the receiving node is merely a forwarder which will not deliver the message to the application

nor does it forward the message to any other node. Nevertheless, node 23 which delegated the interval $[30,30]$ to node 27, has to ensure that it covers all nodes in the region $[30,37]$ and does not know whether a node with identifier 30 exists or not. Therefore it delegates that interval to node 27, which knows that no such node exists.

4.2 Bulk Operations with Feedbacks

Algorithm 22 shows the algorithm for doing bulk operation with feedback from all the nodes with identifiers in a prescribed bulk set I.

In the simple broadcast with feedback, the nodes that are merely forwarding the message will not provide any feedback to the initiator. Nevertheless, they will forward, on behalf of other nodes, feedback on its way back to the initiator, hence forwarding nodes should be placed in the waiting *Ack* set.

4.3 Bulk Owner Operations

We now extend the bulk operation to introduce the bulk owner operation, which is designed to reach all the nodes that are responsible for an identifier in the bulk set I.

Algorithm 21 reaches every node that has an identifier in the bulk set I. These nodes should also be reached by the bulk owner algorithm, since every node is responsible for its own identifier. Sometimes, however, a node n responsible for an identifier in the bulk set is not itself in the bulk set, i.e., $n \notin I$. We first show how Algorithm 21 can be naïvely changed to accomplish this, and thereafter we optimize it.

Algorithm 22: Bulk operation with feedback algorithm

```
receipt of BulkFeed($I$, $msg$) from $m$ at $n$
    if $n \in I$ then
        $FB$ := Deliver($msg$)
    else
        $FB := \emptyset$
    end if
    $par := m$
    $Ack := \emptyset$
    $limit := n$
    for $i := M$ downto 1 do
        $J := [u(i), limit)$
        if $I \cap J \neq \emptyset$ then
            sendto $u(i)$ : BulkFeed($I \cap J$, $msg$)
            $I := I - J$
            $Ack := Ack \cup \{u(i)\}$
            $limit := u(i)$
        end if
    end for
    if $Ack = \emptyset$ then
        sendto $par$ : BulkResp($FB$)
    end if
end receipt event

receipt of BulkResp($F$) from $m$ at $n$
    if $m = n$ then
        sendto $app$ : BulkFeedTerm($FB$)
    else
        $Ack := Ack - \{m\}$
        $FB := FB \cup F$
        if $Ack = \emptyset$ then
            sendto $par$ : BulkResp($FB$)
        end if
    end if
end receipt event
```

The simplest way to ensure that all the designated nodes are reached is to add the statements found in Algorithm 23 after the for loop in Algorithm 21 . Hence, each node n first executes the statements in Algorithm 21 and thereafter checks to see if there are any identifiers in the bulk set I that are between itself and its successor, i.e., $(n, u(1)]$, in which case it sends a message to its successor $u(1)$.

The naïve extension has one major drawback, it might deliver the same message to the same node multiple times. Algorithm 24 optimizes the naïve extension to avoid the sending of redundant messages.

We modify the algorithm to add another parameter R, which holds a set of identifiers. A node sending a BulkOwn message sets R to the set of identifiers which the destination node is potentially responsible for. Initially, R is equal to the bulk set

Algorithm 23: Extension to bulk operation

```
$J := (n, u(1)]$
if $I \cap J \neq \emptyset$ then
    sendto $u(1)$ : BULK($I \cap J$, $msg$)
end if
```

I. This set R is always checked when receiving a BULKOWN message to determine if there are any identifiers in R that the local node is responsible for.

The first modification to avoid redundant messages is to let every node keep a boolean variable *sendsucc* which indicates whether the node sent a message to its successor or not. If *sendsucc* is true after executing the for loop, the node will not again send a message to its successor.

The last described modification to the algorithm does not ensure that a node will not send a redundant message. Even if *sendsucc* is false, the successor might have received the BULKOWN message from some parent of the current node. The second modification is to include a parameter *next* whenever sending a message to any node m, where *next* is the closest successor of m which is known to have received the BULKOWN message. Hence, a node will not send a message to its successor if *next* is its successor, or if *sendsucc* is true. Initially, *next* is set to the identifier of the initiator.

5 Fault-Tolerance

So far we have assumed that failures do not occur. If the goal is to have best effort delivery, then the broadcast and the bulk algorithms will be providing best effort delivery in the presence of failures. However, this is not the case with the algorithms which provide feedback. In those, every node, prior to sending its feedback, waits to receive feedback from all nodes to which it has sent a message. If any of those nodes fail, the waiting node will block forever, making the whole algorithm deadlock.

A straightforward approach to ensure termination for all algorithms, is to introduce timeouts. Hence, a node waiting for feedback, can time out and send its response back to its parent. A premature timeout might result in some nodes sending their feedback to a parent which has timed out. Such messages can simply be ignored by the node which timed out.

A best effort delivery might, however, be inadequate. For example, assume the initiating node attempts to broadcast to all nodes in the system. Also assume that it has $M = \log_2(n)$, pointers placed as in Chord (or according to the k-ary principle when $k = 2$). In such a case, the broadcast will partition the whole space into M intervals and delegate responsibility to each routing pointer $u(i)$. The pointer $u(M)$ will be pointing to the predecessor of the initiator, and $u(M-1)$ will be delegated roughly half of the identifier space. If $u(M-1)$ fails, then roughly half of the nodes will not be reached by the broadcast. This is exacerbated as $u(M-1)$ might not have

Algorithm 24: Bulk owner operation algorithm

```
receipt of StartBulkOwn($I$, $msg$) from $m$ at $n$
    sendto $n$ : BulkOwn($I$, $I$, $n$, $msg$)
end receipt event

receipt of BulkOwn($I$, $R$, $next$, $msg$) from $m$ at $n$
    $MS := R \cap (u(M), n]$
    if $MS \neq \emptyset$ then
        Deliver($msg$, $MS$)
    end if
    $limit := n$
    $lnext := next$
    $sentsucc :=$ false
    for $i := M$ downto 1 do
        $J := (u(i), limit]$
        if $I \cap J \neq \emptyset$ then
            $K := (u(i-1), u(i)]$
            sendto $u(i)$ : BulkOwn($I \cap J$, $I \cap K$, $lnext$, $msg$)
            $I := I - J$
            $limit := u(i)$
            $lnext := u(i)$
            if $i = 1$ then
                $sentsucc :=$ true
            end if
        end if
    end for
    $J := (n, u(1)]$
    if $I \cap J \neq \emptyset$ and $sentsucc =$ false and $next \neq u(1)$ then
        sendto $u(1)$ : BulkOwn($\emptyset$, $I \cap J$, $limit$, $msg$)
    end if
end receipt event
```

failed, but just left the system, and the initiator is not yet aware of it. Hence, even with reliable communication channels and no failures, the algorithm might fail to cover significant number of nodes. The latter can be avoided if a the ring and the additional pointers are never stale, as guaranteed by certain systems [19, 25, 29].

In the distributed algorithms field, *reliable broadcast* has been studied extensively [10, 11, 20]. Reliable broadcast ensures two things. First, if the initiator of a broadcast does not fail, all correct nodes eventually receive the message. Second, all correct processes always receive the same set of messages, regardless of any crash failure.[2]

The second requirement of reliable broadcast is quite strong. For example, assume an initiating node starts to broadcast a message and then fails before completing. If some nodes receive and deliver the message, then all other correct nodes must also eventually receive that message and deliver it. Such a strong requirement, however, is justified in many scenarios. Reliable broadcast is used as a building

[2] It also ensures that a received message actually has been sent, such that messages are not "invented" by the algorithm.

block in middleware for building reliable distributed systems, such as Isis [5] and Transis [2]. Reliable broadcast, with the additional constraint that all nodes should deliver messages in the same order, has been shown to be equivalent to the *consensus* problem [10], which is an important theoretical problem with many implications in distributed computing [18].

Our goal is to make broadcast or bulk algorithms, which give stronger guarantees than pure best-effort delivery, but weaker guarantees than reliable broadcast. The motivation for this is that reliable broadcast is quite costly to achieve. The algorithm given by Chandra and Toueg [10] has a message complexity of $O(n^2)$ for n nodes. Furthermore, in some application scenarios, the initiating node is using broadcast to collect information. If the initiating node fails, there is no interest in ensuring correct delivery to all nodes.

5.1 Pseudo Reliable Broadcast

We will modify our simple broadcast algorithm and introduce *pseudo reliable* broadcast. The algorithms will depend on the initiating node not failing. Hence, it has the correctness properties listed in Section 2.1 with two minor modifications. First, the coverage property assumes the initiator does not fail. Second, the designated nodes are all the correct processes. Informally, this means that a broadcast message will reach all processes that remain in the system between the time the algorithm is initiated and terminated, provided that the initiating node does not fail.

The pseudo reliable algorithms will use failure detectors that use timeouts to detect when a node has failed. We assume that the failure detector is strongly complete, which means that it will eventually detect if a node has crashed. The failure detector might, however, not be accurate, which means that it might give false-positives, suspecting that a correct, albeit slow, node has crashed. The only consequence of inaccuracy is increased bandwidth consumption because of the redundant messages being sent. We assume that whenever a failure detector suspects a failure, it will immediately trigger the correction of the routing information, ensuring that pointers to crashed nodes are eventually removed.

There can, however, be a complication in the implementation of the failure detector. The time it takes for a broadcast to terminate depends on the number of nodes in the system, which in turn determines the depth of the broadcast tree. Therefore, using a timeout when waiting for an acknowledgment from a child is problematic, as the parent does not know the depth of its subtree, and can therefore not determine the time to wait before triggering a timeout. We therefore assume the implementation of the failure detector is independent of the broadcast algorithm, as is the case with most failure detectors. This can be achieved by having the failure detector periodically sending a message to its children and awaiting an acknowledgment.

Pseudo-Reliable Broadcast

We first describe a simple way to provide pseudo-reliability and thereafter suggest some improvements.

Every broadcast has a globally unique random identifier associated with it, which is included in every message. Nodes keep track of previously seen identifiers and filter any message which has an identifier previously seen. Hence, redundant messages are filtered.

The algorithm works like the broadcast algorithm with feedback. Hence, the pseudo-reliable algorithm makes use of an *Ack* set, containing the children of the node. In addition to keeping the identities of the children in the *Ack* set, the algorithm keeps track of the interval delegated to each child.

The algorithm is made resilient to failures by using the bulk operation to resume after a failure. Whenever a node suspects that one of its children has failed, it uses the *Ack* set to determine the interval delegated to the failed node. Thereafter, the bulk algorithm is used to cover all nodes in that interval. The bulk algorithm will retry to cover all nodes in that interval. If the suspicion is incorrect, then the receiver will ignore the previously seen message. The algorithm terminates whenever the initiator's *Ack* set is empty. To ensure that the algorithm terminates, it is required that no node inaccurately suspects its children for a long enough time period, such that all *Ack* sets become empty. This requirement is ensured by an eventually perfect failure detector. Note, however, that this failure detector provides a stronger guarantee, as it ensures that, eventually, there will be no inaccurate suspicions by any node.

Improving Pseudo-Reliable Broadcast

The pseudo-reliable broadcast algorithm can be improved to reduce bandwidth consumption. The basic idea is to try to avoid the re-sending of redundant messages after failures. We motivate this by an example. Assume all the children of a node q, except one, are done covering their delegated intervals. Hence, the *Ack* set of q contains a single child. If q fails, q's parent p will detect that and reassign the interval delegated to node q to a new node r. Hence, the new node r will retry to cover all of q's children, rather than the remaining one. Even though p's children will filter those redundant messages, r will still consume resources to send those messages.

To avoid redundant messages, nodes could periodically send an update of their current *Ack* set to their parent. Hence, nodes periodically report their current *Ack* set to their parent, which upon receipt of the updated *Ack* set updates its child's delegated interval to the received *Ack* set. If the previous example was using the suggested improvement, node q would send its updated *Ack* set, containing its last remaining child, to its parent p. When q later crashed, node p would only delegate an interval containing the remaining child of p to the new node r.

6 Application: Efficient Overlay Multicast

In this section we briefly describe how the algorithms presented in this chapter can be used to provide efficient overlay multicast. Overlay multicast can also be perceived as *topic-based publish/subscribe* [17]. In short, a topic-based publish/subscribe system consists of two actors, *subscribers* and *publishers*. A subscriber can subscribe to different topics of interest. Publishers can publish information about an event and a notification of the event will be sent to all subscribers of that particular topic.

The motivation for doing multicast in an overlay network is that the Internet, de facto, does not provide global multicast capability. The reason for this is that many of the routers that form the back-bone of the Internet have multicast turned off, or do not support it. Several overlay networks have been built, such as Multicast Backbone (MBONE) [16], but these lack the self-managing properties that structured overlay networks possess.

6.1 Basic Design

The basic idea is to store information about all multicast groups in a structured overlay network, which we refer to as Γ. Every multicast group is represented by an instance of an overlay network. To multicast a message to a particular group, the message is broadcast to all nodes in that overlay network, using the pseudo-reliable broadcast algorithm described earlier. This approach is similar to [31], except our system avoids sending redundant messages, and reaches all n nodes in a multicast group in $O(\log n)$ time steps, instead of $O(n^{\frac{1}{d}})$. Figure 5 shows an example of a system with two groups.

6.2 Group Management

Group information is stored in the Γ using the group name as a key and group information as a value. Group information consists of contact information for a random subset of the members of the group and additional meta-data about the group. The random subset is kept up to date by the node responsible for the item containing the group information. The responsible node periodically contacts the first alive node that it knows in the group information, and asks it for its routing table. It then updates its random subset by keeping a constant number of those nodes, giving preference to those just received. The responsible node deletes any group information when it cannot find any alive node in a group.

To avoid a responsible node receiving too many requests for a popular group in the Γ, group information is automatically replicated in the overlay, for example

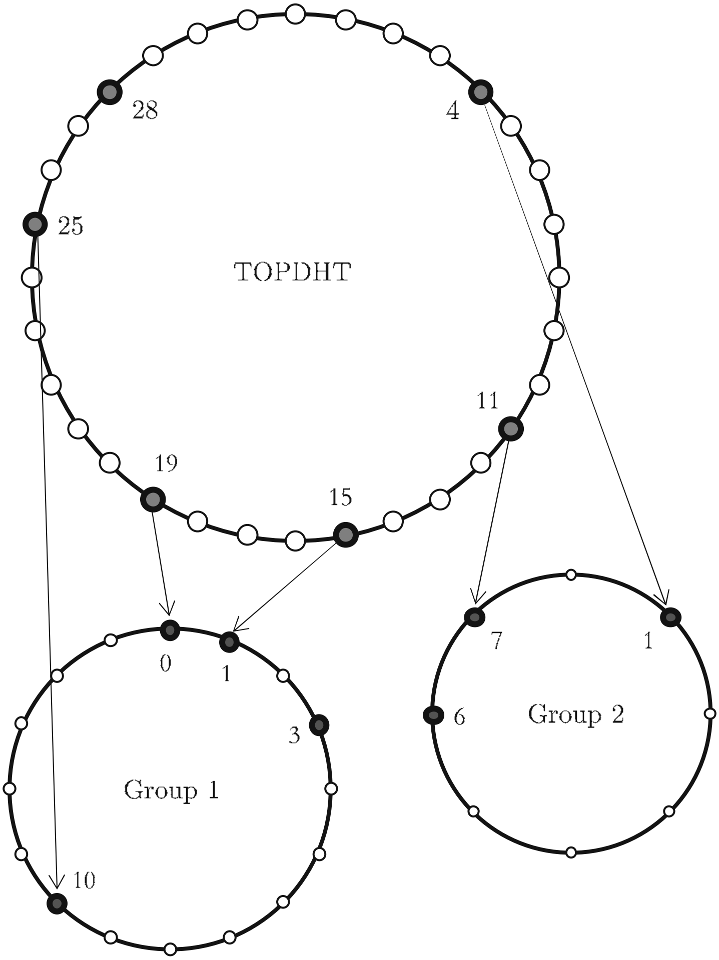


Fig. 5 A Γ containing information about two different multicast groups. The *arrows* represent information about random group members in each group

using successor-list replication [9]. Furthermore, group information can be cached along the lookup path to further relieve nodes in Γ from hot-spots.

Joining and Leaving Multicast Groups

To join a multicast group, a member of the multicast group is obtained by making a lookup for the groups name in the Γ. After obtaining a random subset of the group members, the first alive node in that subset is contacted to join that group. The last node in a multicast group takes special action by deleting the group information in the Γ.

Group Creation

An atomic create operation is provided by the nodes in the Γ to avoid conflict resolution when several nodes try to create a group with the same name. Hence, to create a group, a request is sent to the responsible node, which checks to see if such a group already exists, in which case it reports back with a failure. Otherwise, it creates a group having the creating node as the only node in its group information. Hence, the first node that reaches the responsible node will be successful in creating the desired group.

6.3 IP Multicast Integration

We provide a mechanism to integrate overlay multicast with IP multicast to make efficient use of the network resources.

The integration of overlay and IP multicast is done by modifying the node behavior when nodes join a multicast group and when they multicast to a multicast group.

Joining a multicast groups works as follows. Every multicast group is associated with a multicast address, which is stored in the Γ together with its group information. The joining node first queries the Γ to find out about existing members of the structured overlay network and the multicast address to be used for that group. Thereafter, a joining node attempts to discover other nodes in the desired group to which it can directly IP multicast. This is done by sending a discovery message to the group's multicast address. If another node receives this message, it responds with its contact information. To avoid an explosion of concurrent responses, similar techniques as used by the Internet Group Management Protocol (IGMP) can be deployed [14]. After establishing contact with an existing node, the joining node joins the desired multicast group using the *same* identifier in the overlay as the existing node. Hence, in each multicast group, nodes that can directly communicate with IP multicast have the same identifier. Therefore, a routing table entry pointing to the successor of some identifier, can have many *candidate* successors. We therefore extend the routing tables to contain up to a constant number of candidates per routing entry.

Broadcasting to a multicast group goes in two steps. The first step is to use the broadcast algorithm to reach one node for every IP multicast island in the multicast group. This is done by using the previously described broadcast algorithm with a minor modification. Since each routing pointer might contain more than one candidate, the algorithm has to pick one candidate using some desirable metric. The second step of the broadcast is to let every node that receives the broadcast message to IP multicast the message.

The advantage of letting candidate nodes share the same identifier is that no single node need to represent the whole multicast group. Hence, no single node

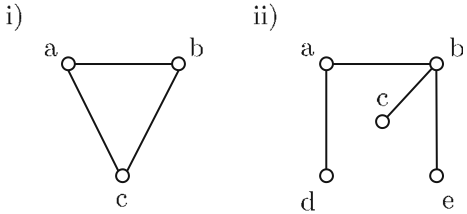


Fig. 6 (i) shows a graph with a cycle (ii) shows a graph without any cycles

will be a single-point of failure, nor will there exist a single bottleneck for that group.

7 Related Work

The nodes in a structured overlay network form a *graph* containing a set of *vertices* and a set of *edges* between the vertices. Each node in the structured overlay network represents a vertex in the graph, and each pointer from a node *a* to another node *b* in the overlay represents an edge between those vertices. General purpose algorithms for broadcasting to all nodes in such a graph have existed for many decades in the field of distributed algorithms [36, Chapter 4] [28, pg 496ff] [4, Chapter 2]. There is, however, a fundamental difference between those algorithms and the ones we present in this chapter.

Graphs may contain *cycles* (see Fig. 6). A cycle starts at a vertex and visits other vertices by traversing edges, and ends in the start vertex, without ever visiting any vertex more than once, except for the start vertex, which is visited twice.

The possibility of cycles complicates broadcasting in graphs. Generally, broadcast algorithms for graphs prevent messages from traveling in cycles indefinitely. These algorithms have two disadvantages compared to the overlay broadcast algorithms we will present. First, they bear an additional message complexity as some messages will be redundant. For example, if the graph consists of three nodes $\{a,b,c\}$, with edges between (a,b), (b,c), and (c,a), only two messages would suffice to broadcast from any node to all other nodes. General broadcast algorithms for graphs would, however, use four messages to cover those nodes, because each node would send the message to all neighbors, except the one it got the message from. For example, if *a* initiated the broadcast, *a* would send to *b* and *c*, *b* would send to *c*, and *c* would send to *b*. This is a consequence of not having global information about the whole graph at each node. Second, such algorithms use extra state at each node to avoid messages wandering in cycles indefinitely. In the previous example, each node would keep track of the received messages. Thus, node *b* would drop the message from node *c*, and node *c* would drop the message from node *b*, as both nodes already have received the message from another neighbor earlier.

Structured overlay networks contain cycles, the algorithms we present in this chapter make use of the internal structure of structured overlay networks to avoid the redundant messages and the extra state associated with avoiding cycles. This is achieved even though no node has global information about the graph.

There are several attempts to provide multicast on top of structured overlay networks. Our approach has several advantages compared to other structured overlay multicast solutions. First, only nodes involved in a multicast group receive and forward messages sent to that group, which is not the case in other systems, such as Scribe [8] and Diminished Chord [22]. Second, the multicast algorithms ensure that no redundant messages are ever sent, which is not the case with many other approaches [23], such as CAN-Multicast [31]. Finally, the system integrates with the IP multicast provided by the Internet, such that the overlay is just used to reach IP-multicast enabled islands, and thereafter IP multicast is used to efficiently reach all nodes in such an island.

Some application level multicast systems are built for the sole purpose of multicasting, rather than being built on top of an existing structured overlay, as in our case. In End-system Multicast or Narada [13], a mesh is built among all nodes participating in the multicast, requiring all nodes to know about each other. In our system, no such mesh is necessary. In other systems, such as Scattercast [12], centralized proxies or rendezvous are used as special infrastructure nodes. Our solution makes no such assumptions.

A final note is that the algorithms provided in this chapter can be used on top of structured overlay networks that have a low constant diameter, such as a one-hop DHT [24, 30]. Though our algorithms work in such cases, they become trivial as a single broadcaster will send a message to every other end node in the system. Such an approach is therefore not scalable. However, if only a fraction of the actual pointers at each node are used, the algorithms safely become scalable and the broadcaster again only directly communicates with a fraction of all the nodes.

References

1. L. O. Alima, S. El-Ansary, P. Brand, and S. Haridi. DKS(N, k, f): A Family of Low Communication, Scalable and Fault-Tolerant Infrastructures for P2P Applications. In *Proceedings of the 3rd International Workshop on Global and Peer-To-Peer Computing on Large Scale Distributed Systems (CCGRID'03)*, pages 344–350, Tokyo, Japan, May 2003. IEEE Computer Society.
2. Y. Amir, D. Dolev, S. Kramer, and D. Malki. Transis: A communication subsystem for high availability. In *Proceedings of 22nd International Symposium on Fault Tolerant Computing (FTCS'92)*, pages 76–84, Boston, MA, USA, 1992. IEEE Computer Society.
3. M. Amnefelt and J. Svenningsson. Keso – A Scalable, Reliable and Secure Read/Write Peer-to-peer File System. Master's thesis, KTH/Royal Institute of Technology, Stockholm, Sweden, 2004.

4. H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced topics*. Wiley series on parallel and distributed computing. John Wiley & Sons, second edition, 2004.
5. K. P. Birman and T. A. Joseph. Reliable communication in the presence of failures. *ACM Transactions on Computer Systems (TOCS)*, 5(1):47–76, 1987.
6. M. Castro, M. Costa, and A. Rowstron. Should we build Gnutella on a structured overlay? *SIGCOMM Computing Communication Review*, 34(1):131–136, 2004.
7. M. Castro, M. Costa, and A. Rowstron. Debunking Some Myths About Structured and Unstructured Overlays. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'05)*, Boston, MA, USA, May 2005. USENIX.
8. M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralised application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC) (Special issue on Network Support for Multicast Communications)*, pages 1489–1499, 2002.
9. J. Cates. Robust and Efficient Data Management for a Distributed Hash Table. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, May 2003.
10. T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, 1996.
11. J. Chang and N. F. Maxemchuk. Reliable Broadcast Protocols. *ACM Transactions on Computer Systems (TOCS)*, 2(3):251–273, 1984.
12. Y. Chawathe. Scattercast: an adaptable broadcast distribution framework. *Multimedia Systems*, 9(1):104–118, 2003.
13. Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *International Conference on Measurements and Modeling of Computer Systems (SIGMETRICS'00)*, pages 1–12, 2000.
14. S.E. Deering. Host extensions for IP multicasting. RFC 1054, May 1988. Obsoleted by RFC 1112.
15. S. El-Ansary, L. O. Alima, P. Brand, and S. Haridi. Efficient Broadcast in Structured P2P Netwoks. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, volume 2735 of *Lecture Notes in Computer Science (LNCS)*, pages 304–314, Berkeley, CA, USA, 2003. Springer-Verlag.
16. H. Eriksson. MBONE: the multicast backbone. *Communications of the ACM*, 37(8):54–60, 1994.
17. P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.
18. M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.
19. A. Ghodsi. *Distributed k-ary System: Algorithms for Distributed Hash Tables*. PhD dissertation, KTH – Royal Institute of Technology, Stockholm, Sweden, October 2006.
20. V. Hadzilacos and S. Toueg. A Modular Approach to Fault-Tolerant Broadcasts and Related Problems. Technical Report TR94-1425, Cornell University, 1994.
21. N. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS'03)*, Seattle, WA, USA, March 2003. USENIX.
22. D. R. Karger and M. Ruhl. Diminished Chord: A Protocol for Heterogeneous Subgroup Formation in Peer-to-Peer Networks. In *Proceedings of the 3rd Interational Workshop on Peer-to-Peer Systems (IPTPS'04)*, volume 3279 of *Lecture Notes in Computer Science (LNCS)*, pages 288–297. Springer-Verlag, 2004.
23. B. Koldehofe. *Distributed Algorithms and Educational Simulation/Visualisation in Collaborative Environments*. PhD thesis, Chalmers University of Technology, Gothenburg, Sweden, 2005.
24. B. Leong, B. Liskov, and E. Demaine. EpiChord: Parallelizing the Chord Lookup Algorithm with Reactive Routing State Management. In *12th International Conference on Networks (ICON'04)*, Singapore, November 2004. IEEE Computer Society.

25. X. Li, J. Misra, and C. G. Plaxton. Concurrent maintenance of rings. *Distributed Computing*, 19(2):126–148, 2006.
26. D. Liben-Nowell, H. Balakrishnan, and D. R. Karger. Analysis of the Evolution of Peer-to-Peer Systems. In *Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC'02)*, pages 233–242, New York, NY, USA, 2002. ACM Press.
27. D. Liben-Nowell, H. Balakrishnan, and D. R. Karger. Observations on the Dynamic Evolution of Peer-to-Peer Networks. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS'02)*, volume 2429 of *Lecture Notes in Computer Science (LNCS)*. Springer-Verlag, 2002.
28. N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.
29. N. A. Lynch, D. Malkhi, and D. Ratajczak. Atomic Data Access in Distributed Hash Tables. In *Proceedings of the First Interational Workshop on Peer-to-Peer Systems (IPTPS'02)*, Lecture Notes in Computer Science (LNCS), pages 295–305, London, UK, 2002. Springer-Verlag.
30. L. R. Monnerat and C. L. Amorim. D1ht: a distributed one hop hash table. In *20th International Parallel and Distributed Processing Symposium (IPDPS'06)*, 2006.
31. S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level Multicast using Content-Addressable Networks. In *Third International Workshop on Networked Group Communication (NGC'01)*, volume 2233 of *Lecture Notes in Computer Science (LNCS)*, pages 14–29. Springer-Verlag, 2001.
32. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 2nd ACM/IFIP International Conference on Middleware (MIDDLEWARE'01)*, volume 2218 of *Lecture Notes in Computer Science (LNCS)*, pages 329–350, Heidelberg, Germany, November 2001. Springer-Verlag.
33. B. Stefansson, A. Thodis, A. Ghodsi, and S. Haridi. MyriadStore. Technical Report TR-2006-09, Swedish Institute of Computer Science (SICS), May 2006.
34. I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the ACM SIGCOMM 2001 Symposium on Communication, Architecture, and Protocols*, pages 149–160, San Deigo, CA, August 2001. ACM Press.
35. I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking (TON)*, 11(1):17–32, 2003.
36. G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, second edition, 2000.
37. B. Y. Zhao, L. Huang, S. C. Rhea, J. Stribling, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A Global-scale Overlay for Rapid Service Deployment. *IEEE Journal on Selected Areas in Communications (JSAC)*, 22(1):41–53, January 2004.

Part VIII
Multimedia Content Delivery

Peer-to-Peer Content Distribution and Over-The-Top TV: An Analysis of Value Networks

Jorn De Boever and Dirk De Grooff

Abstract The convergence of Internet and TV, i.e., the *Over-The-Top TV* (OTT TV) paradigm, created opportunities for P2P content distribution as these systems reduce bandwidth expenses for media companies. This resulted in the arrival of legal, commercial P2P systems which increases the importance of studying economic aspects of these business operations. This chapter examines the value networks of three cases (Kontiki, Zattoo and bittorrent) in order to compare how different actors position and distinguish themselves from competitors by creating value in different ways. The value networks of legal systems have different compositions depending on their market orientation – Business-to-Business (B2B) and/or Business-to-Consumer (B2C). In addition, legal systems differ from illegal systems as legal companies are not inclined to grant control to users, whereas users have most control in value networks of illegal, self-organizing file sharing communities. In conclusion, the OTT TV paradigm made P2P technology a partner for the media industry rather than an enemy. However, we argue that the lack of control granted to users will remain a seed-bed for the success of illegal P2P file sharing communities.

Jorn De Boever
Centre for UX Research / IBBT, K.U.Leuven, Parkstraat 45 Bus 3605 - 3000, Leuven, Belgium,
e-mail: `jorn.deboever@soc.kuleuven.be`

Dirk De Grooff
Centre for UX Research / IBBT, K.U.Leuven, Parkstraat 45 Bus 3605 - 3000, Leuven, Belgium,
e-mail: `dirk.degrooff@soc.kuleuven.be`

Portions reprinted, with permission, from (De Boever, J. (2008). *Value Networks of Peer-to-Peer TV: An Analysis of Actors and their Roles.* Proceedings of the Third International Conference on Internet and Web Applications and Services, Athens, Greece). ©2008 IEEE.

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_34, © Springer Science+Business Media, LLC 2010

1 Over-The-Top TV and P2P Systems

During the past decades, the broadcasting industry has evolved in tremendous ways, driven by new challenges and opportunities caused by regulatory decisions and technological evolutions. These evolutions have been characterized by three main models, which strongly resemble the three architectures for interaction that Lechner and Hummel [15] identified: the mass communication (e.g., unidirectional client/server model, linear analogue TV), the mass customization (e.g., non linear interactive model, interactive television, interactive client/server models), and the multilateral community model (e.g., bittorrent clients, communities, Gnutella clients). We will give a brief overview of these three interaction models applied to the TV industry and we will dilate upon in which of these paradigms P2P (Peer-to-Peer) technology fits in.

1.1 Evolutions of TV

First, TV arose after the Second World War as it entered more and more living rooms to become a commodity that filled in much of the leisure time of households. In those days, television was an analogue, linear medium that provided limited or no interactivity. The offer of programs was confined because of the restricted amount of players. In other words, television was a one-way, passive, lean-back medium and the audience had to take what was available. This model is called the *unidirectional mass communication model* [15].

Next, in the eighties-nineties, more players entered the market in countries where governments decided to stop the monopoly position of public broadcasting companies and allow (limited) competition. The viewers benefited from these regulatory decisions as the program offer expanded providing the end users with more alternatives. As new players entered the market, another evolution was on its way, the interactive non linear digital TV paradigm or the *interactive mass customization model* [15], which promised to bring a new TV experience in which users would become more involved in the programs by several kinds of interactive functionalities. TV would be more personalized and users would no longer be bound by fixed program guides as VoD (Video-on-Demand) was introduced. Other interactive functionalities were developed to engage users in programs such as "push-the-red-button" functionality. The digitalization and introduction of interactivity led to new challenges e.g., on the domain of advertisements as the audience became more fragmented and as customized ads were possible. Another challenge emerged as new competitors entered the market, e.g., many ISPs (Internet Service Provider) started offering content themselves.

Finally, the Internet created a third model, i.e., the *multilateral community model* [15], which introduced new challenges to the industry as they were facing issues such as social networks, user-generated content, new players, and innovative advertising models. This paradigm meant a tremendous shift in the power relations between the commercial industry and the end users as the Internet allowed users to gain control of what, where and how they consumed content. Many users started participating in all kinds of independent, self-correcting, self-organizing, decentralized communities to build software (e.g., open source), create user-generated content (e.g., video images of 9/11, Wikipedia) and to exchange media files (e.g., decentralized P2P file sharing) [5]. In addition, the end users gained a strong voice as new gatekeepers who decide which content has sufficient quality and which not, and as such filtering out uninteresting content. There is no question that this changed balance of powers had a disrupting impact on the business models of the media industry that was forced to reflect on its role in the value network.

The question presents itself: where are peer-to-peer (P2P) systems situated within these interaction models? Surprisingly, P2P technology is utilized in all three mentioned interaction models. As the media industry wants to preserve control of their content, they will implement the unidirectional mass communication model, or the interactive mass customization model, whereas the end users that are frustrated with the restrictions regarding the use of the content will organize themselves in decentralized P2P networks (i.e., the multilateral community model).

1.2 Over-The-Top TV

As previously mentioned, the TV landscape has already handled the shift from the analogue, linear broadcast generation to the interactive digital broadcast generation. However, a new generation shift is on the move, which stimulates content providers and broadcasters to deliver video or TV over the traditional Internet. The broadcasting industry is forced to distribute their content over the Internet as their audience watches less TV and spends more time surfing the Internet. EIAA Mediascope Europe [9] found that 40% of their sample watched less TV in favor of the Internet. Youngsters (16–24 years old) in Europe even spend more time on the Internet than watching TV. To preserve advertising revenues, broadcasters have to enter the slippery market of traditional Internet content distribution. This recent trend of convergence between the traditional Internet and TV, called *Over-The-Top* (OTT) TV/video, challenges the business models of the involved players active in the industry, and explores the boundaries of the *best effort* Internet. The term OTT TV has probably become more popular as the research company Forrester utilized this term in a report. OTT TV does not mean that the actors abandon former generations. Rather, it implies that the actors in the TV industry get involved in multi-channel distribution as the audience becomes more fragmented.

Introducing video/TV on the Internet brings along many challenges as video content requires a lot of bandwidth capacity, which also causes an increased traffic load on the infrastructure of ISPs (Internet Service Providers). The IDC [14] estimated that, between 2006 and 2010, the volume of information on the Internet will be multiplied by six. Probably, one of the driving forces of this increasing volume of data on the Internet will be caused by video, which put a lot of pressure on diminishing costs of distributing bandwidth consuming content. Distributing high quality video and TV to mass audiences over the Internet requires massive amounts of bandwidth, resulting in more costs. Accordingly, broadcasters, and other companies that have to distribute bandwidth consuming content over the Internet, are looking for new models that are characterized by low marginal costs of content distribution. Norton [19] compared several Internet content distribution models and came to the conclusion that P2P systems were the least expensive for the distribution of video content. Liebau et al. [17] compared the costs of two cases of P2P content distribution – BBC's integrated Media Player and Octoshape's P2P streaming of the Eurovision Song Contest – with the costs of client/server content distribution. These authors concluded that P2P content distribution can attain cost-savings up to 90% in comparison with server-based content delivery.

In summary, the OTT TV paradigm involves issues on the level of: video quality (High Definition or not?), bandwidth expenses, Quality of Service (best effort model), multi-channel distribution, new competitors, user-generated content, piracy, new consumption patterns, advertising, social networks, personalization of the offer, asymmetric network architectures, new value chains, etc. In this digital ecosystem, where TV and Internet are converging, P2P technology might become an important partner of the entertainment industry rather than an enemy as P2P systems have been put forward as an interesting method to distribute video to mass audiences in a scalable and cost-efficient manner. Consequently, P2P technology might team up with the media industry in the OTT TV/video generation for content delivery, which has already resulted in the foundation of several commercial companies that offer P2P content distribution.

In this context, the question arises how the different players – in this case the actor offering/owning the content, the P2P provider and the end user – will position themselves strategically in their OTT TV value network? However OTT TV/video has implications on several levels (e.g., copyrights, net neutrality, user behavior), the aim of the present chapter is to address an important aspect of business models, namely value networks. Value networks describe the architecture of which actors execute which roles and how this results in value creation. In order to grasp how players can generate added value, it is important to get a grip on value networks.

After framing our work within the existing literature and theories, we will analyze three cases – Kontiki, Zattoo and bittorrent clients – by means of an actor-roles analysis so as to understand these value networks. These cases were chosen because of their different orientations which make them interesting to compare: Kontiki offers costumer-related services; Zattoo comprises linear TV over the Internet; and the bittorrent clients are file sharing applications. This analysis allows us to grasp how several actors position and distinguish themselves from competing companies, what motivates actors to execute certain roles and to identify who creates value. Next, we will discuss the ambiguous role of the ISPs in the value chain. On the one hand, ISPs welcomed the arrival of P2P file sharing networks – such as Napster and KaZaA – as these file sharing applications drove the customers to broadband subscriptions. On the other hand, ISPs are worried about the arrival of recent legal, commercial P2P systems as these systems cause an increased traffic load on their infrastructure, while their revenues for content distribution drop as the content providers partially outsource content delivery to the end users.

2 Theoretical Framework

Before expanding on business modeling theory, we will provide a review of the most relevant economics literature about P2P systems. As we will demonstrate in this literature review, most of the rare academic papers and articles – that address the economic aspects of P2P systems – focus on one or a few elements of business modeling.

2.1 P2P Systems and Business Modeling

Some authors analyzed the strengths and obstacles of P2P systems to distribute content in a commercial and legal way [7, 24]. MacInnes and Hwang [18] examined several P2P systems on several aspects (e.g., revenue sources, technology, security, etc.) to determine which of the investigated P2P system had the most promising opportunities. From the selected applications (Napster, KaZaA, Kontiki, SETI@home and Groove Virtual Office) the authors concluded that Kontiki and Groove had the most commercial potential because they did well for instance on the level of revenues and benefits offered to the end users.

Other academics designed new services that integrated P2P-technology which refers to the domain of value proposition. On the one hand, Wierzbicki and Goworek [31] designed a business model for a P2P Content Delivery Network (CDN) where users would have to pay to access content. After buying the content, the users would have the opportunity to resell their goods. On the other hand, Gehrke and Schumann [11] proposed a transaction model based on P2P technology to trade physical goods. These researchers tried to launch new services that differ from the already existing services on the market.

Although the business modeling approach, proposed by Ballon [4], emphasizes that the financial model is not limited to the revenue model, the revenue models are a popular topic in the literature which inspired some academics. Hummel, Muhle and Schoder [13] explored several direct and indirect revenue models. A direct revenue model means that users have to pay to view or download content via e.g., pay-per-view, subscriptions, micropayments, etc. In the case of indirect revenue models, the end users can consume the content for free as the content is paid by third parties via advertisements, bundling with other products or subsidies. According to these authors, a viable revenue model should at least fulfill the requirements of differentiated charging and allocation effectiveness. Differentiated charging implies that the revenues should be proportionate to the amount of use. Allocation effectiveness relates to the fact that the rightful owners should be remunerated. Several

P2P systems have failed on the issue of allocation effectiveness, leading to several lawsuits throughout the world.

Finally, some articles explored the use of Digital Rights Management (DRM) techniques to maintain control of when, how and where users are able to consume the content [8, 10]. Content owners want to preserve control of the access, frequency, time and location of consumption. This does not only relate to preventing piracy, but also to provide different versions of digital goods to consumers. The problem of DRM is that these techniques mainly suit the needs of the content owners whereas they often conflict with the users' needs.

2.2 Business Modeling Theory

Before determining the definition of value networks, it is important to grasp the meaning of the concept business model. Many discussions and criticisms have been expressed in the academic society regarding the interpretation of the term business model, leading to a wide array of definitions [20, 22, 28]. Today, there is still no general agreement on its interpretation, leading some of the critics to doubt the use of this term. For instance, Porter [23] complained that many business modeling researchers analyzed the Internet as an isolated economy with no reference to the physical markets, which made him conclude that "The business model approach to management becomes an invitation for faulty thinking and self-delusion" (p. 73).

The vague conceptualization of business modeling points to the fact that researchers have failed to build on existing work [21]. However, several attempts have been made trying to stop this jumble of definitions and interpretations by proposing integrated frameworks [4, 21]. From our point of view, a business model should be considered as a conceptual instrument to gain insight into a complex reality so as to make business operations in terms of value creation intelligible. For a profound discussion on business modeling, we refer to the aforementioned authors as this topic is not the main focus of this chapter.

We have adopted the approach of Ballon [3, 4]. Based on the existing literature, this author has given the initial impetus to make business modeling operational for design and evaluation. This theory stresses the importance of control and value creation to deduce a framework for designing and evaluating business models [4]. The two levels that operate within the control section are the value network (i.e., the architecture of actors, roles and the relationships between them), and the functional architecture (i.e., the architecture of the technical components). The two levels that function within the domain of value creation are the financial model (i.e., the financial streams, revenues, costs, profits) and value proposition (i.e., the outline and positioning of the product or service). These four main pillars constitute a business model. The reasoning behind this framework is that a business model might disclose certain opportunities if there is a fit between all the components of a business model, including a fit with the entire ecosystem in which a company operates. Ballon [3] formulated the following definition of the concept business model:

> A description of how a company or a set of companies intend to create and capture value with a product or service. A business model defines the architecture of the product or service, the roles and relations of the company, its customers, partners and suppliers, and the physical, virtual and financial flows between them. (p. 7)

This definition is broader and more comprehensive than other definitions. For instance, this definition explicitly mentions that the unit of analysis does not only encompass a single company, but several contextual aspects in the ecosystem, including all involved players and stakeholders, must be integrated in the analysis or design of business models.

After having introduced the main concepts related to business modeling theory, in the rest of this chapter, we will focus on one of the four pillars of business modeling, namely value networks of P2P providers. We analyze the value networks as we want to comprehend how different P2P providers position themselves in the market.

2.3 Value Networks

As we already mentioned, our theoretical framework consists of the work of Ballon [3, 4]. We focus on value networks because the architecture of value networks reveals which actors perform what roles, resulting in the determination of who generates what kind of value. Value networks can depict several forms which allow actors to position and to distinguish themselves from competitors [6]. Examining value networks makes it possible to interpret what motivates several stakeholders to execute certain roles [25]. For our analysis, it is important to provide clear descriptions of the concepts so as to exclude misunderstandings. Ballon [4] defined the following concepts:

- Value networks: "The architecture of actors and roles in the (future) marketplace."
- Role: "A role is a distinct value adding activity within the value network that potentially can exist as a (commercial) entity in the marketplace, with its own cost and revenue balance."
- Actor: "An actor is a (commercial) entity active in the marketplace, integrating one or more roles."
- Relationship: "A relationship is the expression of an interaction between roles or actors."

In the definitions of "role" and "actor", it is questionable whether the term "commercial entity" is appropriate as this would exclude users from examination [6]. In our analysis, we have identified the following roles: content production, content aggregation, platform content aggregation, platform provision, content distribution, network provision, customer ownership, marketing and consumption. These roles have been identified based on the work of several authors [15, 16, 25, 29]. In the analysis, we will verify whether the execution of the different roles matches the interests of the involved actors. Regarding this topic, we will pay specific attention to the roles of network operators, which will be treated in a separate section. We focus on the digital distribution of video content over the Internet by means of P2P systems. We will first define the different roles that we have identified.

Content production or provision. The content, which ends up with users, has been produced by persons or organizations with creative ideas. This role might for instance be performed by production studios, broadcasters and increasingly by end users. Content might be published on platforms for several reasons: to increase the scope of the audience (e.g., broadcasters), for advertising purposes (e.g., music industry), to increase visibility (e.g., user-generated content), etc.

Content aggregation. Once content has been produced, an editorial filtering process is being carried out in which one decides which content is appropriate and sufficiently qualitative. Examples of content aggregators are broadcasters and website owners that gather content for their services. Moreover, because of the massive amounts of content that are being offered on some platforms, the bundling of advertisements and other services belongs to the content aggregation role as well. Sometimes, end users are involved in the filtering process – for instance by mechanisms such as ratings and "most viewed" lists.

Platform content aggregation. Van Audenhove, Delaere, Ballon and Bossuyt [29] made a distinction between "content aggregation" and "platform content aggregation". Whereas content aggregation mainly relates to the editorial process (i.e., filtering, editing and branding), platform content aggregation focuses primarily on assembling content providers and content aggregators on the same platform. Several content aggregators utilize several channels to publish their content, and sometimes they publish their content on the same platform as other content aggregators.

Platform provision. This comprises the technical infrastructure to offer the platform. Platform provision serves as the technical linkage between the content and the end users. In many cases, content aggregators and platform content aggregators will make use of the services of external companies for the technical support of a platform. In addition, the roles of platform content aggregation and platform provision will sometimes be executed by the same actor. These companies will try to obtain content deals with content aggregators and content providers so as to be able to offer as much useful content as possible on their platform which might be interesting for an advertising model.

Content distribution. When exploring P2P system cases, content distribution should be considered as a separate role. It is characteristic of P2P systems to involve the end users in distributing the content. In most cases, the actor, responsible for platform provision, will provide the end users with P2P software so as to exploit the capacity that is residing on the end users' equipment. Involving users in this role implies that these users defray the costs content distribution.

Network operators. The network operators are primarily the ISPs that offer the infrastructure for data transport. On the one hand, they offer users access to the content that is available on the Internet. On the other hand, they equip content providers/aggregators with the means to deliver the content to the consumers. ISPs play an important role in the distribution of video content because this content causes an important load on their network infrastructure [12]. In addition, ISPs are no longer solely actors that offer a network infrastructure, but they play the roles of (platform) content aggregators as well. We will dilate on this role in a separate section.

Customer ownership. This role concerns which actor possesses important information regarding their customers and allows the actor to maintain close relationships with their clients [4]. The significance of customer ownership results from purposes of e.g., branding, profiling, reputation management, consumer confidence, customer satisfaction, etc. We have to notice that it is sometimes hard to designate this role to certain actors. A distinction can be made between direct and intermediate customer ownership [4]. It is direct when the actor producing or publishing the content has the role of customer ownership, and it is intermediate when the actor positions itself between the provider/publisher and the customer. On some platforms, there is plenty of content available that lacks a specific or familiar brand, which makes it unlikely that customer ownership resides with the content providers. In these cases, the role of customer ownership is carried out by the aggregators.

Marketing. In this chapter, marketing particularly relates to the use of advertisements to support the content and/or the platform. In other words, this does not relate to the process of creating advertisements which implies that the marketing companies are not taken into consideration. Marketing can be linked to the content, e.g., commercials between two programs. In such cases, this role resides with the content provider. In addition, commercials are often displayed at the platform level, so it resides with the platform provider or platform content aggregator. For advertising, it is important to be able to reach the target audience of the product or service of the announcement.

Consumption. The eventual object of content providers is for their goods or services to be consumed by the end users so as to be able to increase revenues from advertising, increase availability and visibility of content.

3 Case Studies

In this section, we will examine the value networks of content delivery via P2P networks by means of three case studies. The cases under investigation are the following P2P providers: Kontiki, Zattoo and bittorrent clients.

3.1 Case 1: Kontiki

Even though VeriSign decided to acquire Kontiki in 2006, for an amount of about $62 million, they announced that they would sell this service in 2008 [1]. Kontiki offers a service of hybrid P2P content distribution that targets media and other companies for business communication and media content. A few of the realizations of this company include the BBC iPlayer [26] and Hi-Q Video of AOL [27]. Table 1 presents the results of the analysis of the value network of Kontiki. In this table, the role of platform content aggregation is represented between brackets because we did not have an example in which this role was executed. Whether this role will be carried out or not depends on the requirements of the demanding companies.

Table 1 Value network of Kontiki

	Actors		
Roles	Broadcasters	Kontiki	End Users
Content production	✓	-	-
Content aggregation	✓	-	-
Platform content aggregation	(✓)	-	-
Platform provision	-	✓	-
Content distribution	-	✓	✓
Customer ownership	✓	-	-
Marketing	✓	-	-
Consumption	-	-	✓

Depending on who asks Kontiki for peer-assisted delivery solutions, the content is being generated by broadcasters, studios, companies and even by end users (content production). These content providers often use the Internet to expand their reach and availability. Broadcasters filter and bundle content in their editorial office (content aggregation). In other words, the broadcasters decide what is (not) available on the platform and under which conditions. For some of the content providers, such as broadcasters, distributing content on the Internet is mostly an additional service, whereas for others – e.g., Open Media Network – online content provision is their core business. The end users are rather passive in the sense that they are not involved in the content aggregation process.

At the moment, there are no obvious examples of platform content aggregation. From the examples we have seen, there is no actor playing the role of platform content aggregator. This of course depends on the requirements the demanding party formulates. If there is a party who demands an application that aggregates several content aggregators, then Kontiki will develop such a platform that enables all these parties to distribute their content via this P2P platform. However, there are mostly single content aggregators who demand an exclusive distribution platform that operates fully under their control. The major advantage of this approach, for content aggregators, is that they are able to design the user interface themselves so as to highlight their brand. The application might be a stand alone application, but it might be integrated in websites and portals as well.

Kontiki operates almost exclusively in the domain of platform provision. It concentrates on the technical services of the content distribution platform: content delivery, security, intelligent network use, centralized control, monitoring, etc. It provides the necessary facilities for content providers or aggregators to distribute content to the target audience. This offers opportunities for the content aggregators to bring their brand into prominence. In this sense, Kontiki should be regarded as an intermediate actor that offers technical services to content providers and aggregators. The content distribution model utilizes a hybrid P2P technology which means that the idle resources – that reside on the end users' personal computers – are exploited to provide a kind of distributed cache system.

In this model, it is obvious that customer ownership is deposited with the content aggregators or providers. For instance, in the case of the iPlayer, it is evident that BBC establishes the most important relations with end customers. Probably, most users do not even know that Kontiki offers some of the technical functionality. In this sense, Kontiki has no interest in Business-to-Consumer (B2C) customer ownership as key information about the end users has no value for the P2P technology provider for e.g., its brand or reputation. Kontiki does not have direct relations with the end users, because this P2P provider should be considered in a Business-to-Business (B2B) environment where they offer services to other companies.

Kontiki offers its clients the possibility to integrate an advertising or payment model. The revenues from these advertisements or payments end up with the content providers/aggregators. In this case, the role of Kontiki is limited to technical support as well. How the advertising or payment model is integrated depends on the requirements and needs of the client content aggregator/provider.

The consumers in a B2C perspective consume the content from broadcasters. That is why we stated that the role of customer ownership is carried out by content aggregators/providers. The end users are more than customers because they are involved in content distribution as well. Even though Kontiki utilizes the resources of the end users for content distribution, this company does not have a direct relationship with the end users as they operate exclusively in a B2B environment. Kontiki is an actor in a B2B market because they limit themselves to technical services to other companies. This enterprise focuses on a secure content distribution model tailored to the needs of their clients. In this value network, the assets are highly concentrated on the part of the broadcaster, which makes that Kontiki should be regarded as a supporting partner to which the technical components are outsourced. Further, we have to remark that the role of the user might expand in the value network. As we already mentioned, Kontiki offers customer-related services. If there is a content aggregator who decides to distribute user-generated content as well, then Kontiki will allow this. The only roles that are fixed in this case are the roles of the P2P provider, Kontiki. The other roles are not determinate as it depends on the requirements of the demanding content provider/aggregator. This results from the fact that Kontiki is only an actor in a B2B environment and not in the B2C market.

3.2 Case 2: Zattoo

Zattoo was founded in 2005 and can be regarded as an actor that tries to translate the linear broadcast model on the Internet by live streaming of TV channels. This enterprise offers a platform where broadcasters can stream their channels live on the Internet. The main operations of Zattoo include the gathering of several TV channels on one platform (platform content aggregation and platform provision) and the delivery of content via their P2P streaming platform (content distribution). VoD services are not available via this platform. Despite the fact that the distribution of TV content on the Internet is increasing, the core business of the broadcasters remains offering "traditional" television. As more TV viewers spend an increasing amount of time on the Internet, Zattoo tries to offer a way for broadcasters to regain this "lost audience". In this sense, the Zattoo platform should be regarded as a supplement for broadcasters involved in multi-channel distribution, and not as a replacing content distribution model.

Table 2 Value network of Zattoo

	Actors		
Roles	Broadcasters	Zattoo	end users
Content production	✓	-	-
Content aggregation	✓	✓	-
Platform content aggregation	-	✓	-
Platform provision	-	✓	-
Content distribution	-	✓	✓
Customer ownership	✓	✓	-
Marketing	✓	✓	-
Consumption	-	-	✓

The content available on the platform has been generated by production studios or by broadcasters themselves (content production). Next, all the content is being filtered and bundled in an editorial process and advertisements are added in the programming (content aggregation). Zattoo is active as well on the level of content aggregation as they insert advertisements into their platform. When users switch channels, there is a buffer time of 5–7 seconds, which is utilized by Zattoo to display advertisements.

Table 2 indicates that Zattoo is active on the level of platform content aggregation and platform provision as they make several TV channels available on their platform. The technical provision for the reproduction of content is leveraged as well by Zattoo by means of a media player. Zattoo partly delivers the content via their P2P live streaming infrastructure. They offer a streaming platform that is more scalable and cost efficient than the traditional unicast model. In a nutshell, the primary roles of Zattoo are collecting, distributing and presenting content. This value network is quiet similar to the model of traditional linear TV, where cable companies and telecom operators gather content aggregators, supply a platform and take care of the distribution.

Besides the use of the facilities of the broadcasters and Zattoo for the distribution of content, the idle resources available on the peers' equipment are involved in the content distribution as well because it comprises a P2P system. The users of Zattoo have to download software to be able to view the content, which is also

P2P software. This implies that the available bandwidth of the users is utilized to distribute the content which results in decreased costs for broadcasters to distribute video content.

In this case study, it is not immediately obvious who of the actors maintains the most important relationship with the end users (customer ownership). Zattoo takes care of the aggregation of different TV channels, but the audience watches the TV channels and not Zattoo. We already reasoned that the Zattoo model strongly resembles the model of traditional linear TV, which means that this company is trying to bring the traditional TV experience to the Internet. However, it seems that the most prominent relationship with the customers resides with the broadcasters. This is a consequence of the fact that users are solely interested in the content and not in which technology distributes it. The technology should only be regarded as an enabling technology with which users can satisfy their basic needs for certain types of content. The broadcasters need to have direct relationships with its customers for e.g., branding and reputation management. Notwithstanding the fact that the broadcasters have the direct customer relationship with its audience, we argue that Zattoo has an indirect relationship with the audience as well. They have an interest in maintaining good customer relations with the end users as this company depends to some extent on an advertising model which requires them to catch sufficient “eyeballs”. The fact that Zattoo is partially depending on an advertising model, which forces them to be active in a B2C environment, has implications as well in that it obliges this P2P provider to be active in the B2B market as well since most of the eyeballs (B2C) will be captured by offering popular content (B2B). In other words, Zattoo has to attract as many popular TV channels as possible to entice an audience so as to keep their advertisement model viable.

The role of marketing is executed by two actors, namely the broadcasters and the P2P provider. The content aggregators, i.e., the broadcasters, offer their traditional commercials within and between their television programs. In this way, the users of Zattoo view the same commercials that are available on regular TV. Additionally, as already mentioned, Zattoo in its role of platform content aggregator offers ads on its platform by means of banners, text-based ads and video commercials. Zattoo utilizes the buffer period, to switch from one TV channel to another, to integrate advertisements. In conclusion, users are confronted with advertisements on the level of the content aggregators (broadcasters) as on the level of the platform content aggregator (P2P technology provider).

In contrast to the case of Kontiki, which is situated entirely in the B2B domain, Zattoo is active in the B2B and the B2C market. Even the B2B relationship between Zattoo, the broadcasters and the advertising industry is not completely straightforward. On the one hand, some of the small broadcasters are clients of Zattoo as they want to utilize this platform to distribute content and they have to remunerate Zattoo for these services. On the other hand, we can say that Zattoo is most likely a client of the popular broadcasters as they need these popular TV channels to feed their advertising model. For this reason, Zattoo actively tries to attract popular TV channels. In summary, besides the broadcasters, Zattoo wants to maintain a relationship with the end users as well to support their advertising model (B2C).

In conclusion, Zattoo is the central service provider for the broadcasters in this story, which gives Zattoo most control in the B2B value network. By copying the linear TV model on the Internet by means of live streaming, their service addresses the fear of broadcasters that on demand content makes it more difficult to control how the audience consumes the content. Another strength of live streaming consists of the fact that premium content is available as all popular series, movies, sports events that are on the traditional TV are simultaneously available on Zattoo as well. In comparison with the "Zattoo model", which is a live streaming model, on demand streaming platforms – such as Joost and Babelgum – suffer from the fact that broadcasters and other content aggregators will not provide most of their primetime content or blockbusters because most of these programs have specific licenses which restrict when and where this content might be viewed. This is caused by content producers who want to retain control of their content for repurposing and versioning strategies. Table 2 displays the execution of different roles in the value network by the three main actors (i.e., broadcasters, P2P technology provider and consumers). From a B2C perspective, Zattoo should be considered as an intermediary actor between the content provider and the end customers. As a consequence, most control in the B2C value network resides with the broadcasters. The roles of the end users are limited to passive consumption and distribution of content.

3.3 Case 3: Bittorrent Clients

Before beginning this analysis, we have to highlight that bittorrent will be written in lower-case letters to avoid confusion with the BitTorrent company. Bram Cohen, developed the bittorrent protocol and introduced a new era in the file sharing history as this technology was very suitable for sharing large files such as video, games and software. The fact that these bittorrent clients are file sharing systems implies that these can be considered as on-demand services. We focus on systems such as XTorrent and Bitcomet and related torrent sites e.g., The Pirate Bay and Mininova. These systems and websites are primarily utilized to share and download content in an illegal way. Torrent sites are websites where torrent files are made available. These torrent files contain metadata about the different parts of a file and about the tracker that coordinates the file exchange among computers. The reason we are interested in this case is because it allows us to demonstrate how the value networks of legal, commercials systems differs from the illegal use of P2P systems. In this case, we focus on the illegal sharing of content via bittorrent clients. This implies that legal downloads – e.g., the experiments of some broadcasters e.g., the Canadian TV program "Canada's Next Great Prime Minister" and the Norwegian series "Nordkalotten 365" – are excluded from our analysis so as to be able to compare the value networks of legal use of P2P technology with the value networks where content is shared illegally. Of course, we have to remark that the technology itself is not illegal; it is the way people use it that is often illegal. Table 3 outlines the value network of bittorrent clients/websites.

Table 3 Value network of bittorrent clients

	Actors		
Roles	Broadcasters	Bittorrent	End Users
Content production	✓	-	-
Content aggregation	-	-	✓
Platform content aggregation	-	✓	-
Platform provision	-	✓	-
Content distribution	-	-	✓
Customer ownership	-	✓	-
Marketing	-	✓	-
Consumption	-	-	✓

The content available via bittorrent clients/sites has been produced by media companies, artists and production studios (content production). These content owners want to maintain control of their content with the result being that they license their productions. However, many users seem to dislike the restrictions of licensed content and want to gain control of the content themselves. Some of these users videotape this content in movie theatres (i.e., CAM versions), copy it from a DVD (i.e., DVDrip) or have screeners that they release on torrent sites. In other words, users decide which content is available on torrent sites (content aggregation). Much of the files have been aggregated by the so-called "release groups" which are persons or a group of persons that publish content on torrent sites. Some of the release groups have gained an important status in the peer community as a trusted source for files. For example, aXXo is one of the most popular and trusted release groups on torrent sites with the result that this group is a real Internet celebrity. In addition, the whole peer community is passively involved in the content aggregation process just by consuming certain content as this has an impact on the filtering process. As users download content or make it available, they influence the ranking of the content on torrent sites which has an impact on the choices of other users. Further, in the former cases (Kontiki and Zattoo), users are offered only one version of the content, whereas in this case users have to choose a version from the search results. Furthermore, there is a lot of fake and low quality content available in those search results which users try to avoid. This makes the role of content aggregator for the peer community even more important as the most downloaded and shared content is perceived as most trustworthy.

Platform content aggregation takes place by meta-search engines – such as Torrentz and TorrentPond – which allow users to search content on different torrent sites simultaneously. In this way, end users do not have to browse through several sites to find their content as this is assembled on one search engine. The platforms (i.e., the sites and the software providers) are often either proprietary or open source (platform provision).

The execution of the role of content distribution is distinct from the former cases as users are entirely responsible for this, with the result that content might become unavailable if users stop uploading. This has major implications on the level of QoS (Quality of Service) because of the transient connectivity of peers. If peers gradually stop seeding certain content, this content will slowly become unavailable to other interested peers. The speed of exchanging files depends on the number of seeders, free riders and on the download/upload capacity of peers. These characteristics of bittorrent systems make it a best effort model with limited or no guarantees on the level of QoS. However, users seem to accept these flaws because these systems allow them to have full control of the content and also because they can download this content for free.

Customer ownership resides mainly with the torrent sites that have an interest in maintaining good relations with the end users as their revenue model consists of an advertising model (Marketing). For this reason, they want to acquire the most popular release groups to publish the most popular content on their sites so as to attract a large audience to support their advertising model. In Section 2.1, we cited a research [13] that stressed the importance of allocation effectiveness in a revenue model which emphasizes the significance that the content owners should be remunerated. The bittorrent case study is an example of bad "allocation effectiveness" as content owners are not involved in the transactions and revenues, which caused many lawsuits concerning copyright infringements.

In summary, the actors creating and owning the content are not remunerated for their work which causes many lawsuits. Whereas there are some organizations providing the platform and the technology, the peer community exercises their influence over most parts of the value network. The cooperating peers – forming a decentralized, self-organizing community that offers the content – decide which content is fake or has bad quality, they distribute the files and finally they consume the content. In other words, the end users control the value network as they break loose from the restrictions imposed by the media industry or as Benkler [5] formulated this trend: "(...) individuals are less susceptible to manipulation by a legally defined class of others – the owners of communications infrastructure and media" (p. 9).

4 The Role of ISPs

We did not elaborate on this role in the case studies as the execution of this role is the same in different cases. It is interesting however to expand on the role ISPs or network operators play as they take care of an important part of content distribution by leveraging the infrastructure through which the content can be delivered to end users. We treat this role in a separate section because ISPs or network operators find themselves in an ambiguous position [6, 7].

In the beginning, the ISPs welcomed the arrival of (illegal) P2P systems – such as Napster in 1999 – because it drove Internet users from dial-up to broadband connections, leading to more revenues for the network providers. After a few years the penetration of broadband connections caused the amount of dial-up connections to fade away. This increasing number of broadband connections resulted in more users sharing and downloading content, leading to more traffic on the ISPs' infrastructures. Haßlinger [12] found that more than 50% of traffic volume on the ISP networks was emanating from P2P systems. In addition, users began exchanging more bandwidth consuming content such as video and software, which increased network traffic originating from P2P systems even more [30]. The saturation of broadband connections combined with the increased traffic load meant a first turning point in the position of the ISPs regarding P2P systems as they feared that their networks would become congested by traffic that is hard to optimize.

Today, the arrival of legal, commercial P2P systems inclines the ISPs' opinion even further to a reluctant attitude towards P2P systems as the content owners and P2P technology providers bypass the ISPs to distribute content with the result that the revenues for ISPs will shrink: "ISPs are those who suffer under the P2P applications, because they typically charge broadband traffic in a flat rate manner while the content providers apply P2P technology to move traffic from their servers into the network" [17]. P2P systems are cost efficient because content owners outsource content distribution to the end users by utilizing the available resources on the edges of the Internet. P2P technology providers utilize this money-saving characteristic mainly to promote their business. The end users, whose resources are utilized, are confronted with an increasing content offer which urges them to exchange even more content. Additionally, P2P technology changes the traffic patterns of the ISP network as this technology stresses the upload link whereas most of ISPs' infrastructure has an asymmetric architecture (e.g., ADSL) that has been oriented to the download link [24]. In other content distribution models, the content providers or owners have to remunerate the ISP or network provider for the load on the ISPs' infrastructure, e.g., bandwidth, caching, etc. In other words, as a service becomes more popular, the actor offering the content will have to remunerate the ISP proportionate to the load their service has caused on the infrastructure. Content providers/owners that utilize P2P systems shift these costs to the end users by outsourcing of the distribution role to these end users. However, many users have "all-you-can-eat" or "fair use policies" subscriptions which allow them to exchange content in an almost unrestricted way. In other words, the ISPs or network operators are confronted with: (1) an increased load on their networks (especially the upload link) and (2) decreasing revenues as content providers outsource the distribution role to the end users. In addition, many of the P2P systems are to a large extent self-organizing systems which causes a lot of message traffic to keep the network operational. All these issues have resulted in a negative attitude of several network providers towards P2P content distribution, causing ISPs to threaten to queue, throttle and even block P2P systems. We have to notice that the ISPs hold a very strong position because they offer the bridge between content and end users. In this way, ISPs might decide which content users can access and which not. In addition, ISPs play an increasing amount of roles as today they provide their own video content and other services, which make these P2P applications competitors for their own services. All issues mentioned in this section relate to debates on net neutrality, *Walled Garden* and consumer lock-in strategies. The BBC iPlayer, for instance, has already been confronted with resistance from ISPs, who fear that the success of such applications might clog their network, which reveals itself in headlines such as: “ISPs to BBC: We will throttle iPlayer unless you pay up” [2].

As was mentioned in Section 2.2 of this chapter, it is important that all components of a business model fit together [4]. We argue that the business model of most P2P technology providers does not fit the revenue model of many ISPs. This might inspire ISPs to change their revenue models from a flat rate or all-you-can-eat model to a model of layered pricing proportionate to the traffic volume that each user generates [24].

Throttling P2P traffic or introducing usage-based pricing seems to be an unsatisfactory solution for network providers as this might result in unfavorable churn rates. From this analysis, it is clear that P2P technology providers or content providers and ISPs will have to cooperate in order to increase performance and to achieve fair usage of network resources. This challenge to optimize network utilization has resulted in the P4P working group (Provider Portal for aPPlications) in which P2P providers and ISPs cooperate for instance to “(1) facilitate network applications, in particular P2P applications, to achieve the best possible application performance under efficient and fair usage of network resources; and (2) allow network providers to achieve efficient and fair usage of their resources to satisfy application requirements, reduce costs, and increase revenue” [32]. Both P2P providers and network operators have an interest in cooperation as they provide services to the same end users. Therefore, the P4P working group proposes a technical framework in which both parties exchange information so as to optimize traffic flows. More information about the P4P framework can be found in [33].

This argumentation seems to indicate that cooperation between content providers, ISPs and P2P providers will be one of the prerequisites for a viable, commercial P2P system.

5 Discussion and Conclusion

The present study examined the value networks of three cases where P2P technology is utilized to distribute content. The results indicate that the compositions of the value networks, i.e., the architecture of the relationship between actors and roles, vary considerably from case to case. Although there are more than three actors involved in the whole process, we analyzed the roles of the broadcaster/content owners, the P2P technology provider and the end users as these are the most prominent players in these value networks. In this discussion section, we will compare the roles of the three cases studied and will try to provide an adequate interpretation of the results.

First, we will compare the roles of the broadcasters across the three selected cases. Interestingly, we found that broadcasters exert most control of the value network in the case of Kontiki. This probably results from the fact that Kontiki is only an actor in the B2B environment as they offer custom-tailored services to broadcasters in terms of technical infrastructure. Therefore, Kontiki has no interest in maintaining customer relations with the end users, which gives most of control of the B2C value network to the broadcasters. The similarity between the roles of the broadcasters of the Kontiki and Zattoo case is notable. In the case of Zattoo, the broadcasters want to control the value network for their customer relations as well. Broadcasters want to rule over the value networks as they attach great importance to having control of: (1) when and where users may consume content, (2) who might access the content, (3) at what speed the content must be distributed, and (4) QoS. It is interesting to note that content owners have almost no role in the case of bittorrent clients. It may be speculated that this is probably so because, in the case of bittorrent clients, there is a self-organizing community of peers that want to have control of the available content and they therefore have to deprive the content owners of their content.

Second, the roles executed by the P2P technology providers differ conspicuously between the three cases. Although we did not focus on revenue models in our analysis, the data allow us to discuss these as well because revenue models influence to some extent the configuration of value networks. The roles of Kontiki are limited to the technical provision of the platform and the delivery of content. As we already mentioned, this most likely emanates from the fact that Kontiki is an actor in the B2B value network where they offer customer-related services, which implies that they let their customers execute the other non-technical roles. Kontiki has no interest in controlling the B2C value network as they receive revenues directly from the content providers who want to use their services. Facing that, the difference with the Zattoo case is very remarkable as this P2P technology provider has much control of the value network as they are involved in for instance content aggregation, customer ownership and marketing. This might be explained by the fact that Zattoo is partially depending on an advertising model which implies they must attract an audience in order to be a tempting platform for advertisers. To attract an audience for advertisements, they have to make sure that popular content is available on their platform. Therefore, Zattoo wants to exert most control of the B2C value network as their revenues are partially depending on ads. In other words, Zattoo is active in the B2B as well as in the B2C market. In the B2B market, some (smaller) players have to pay Zattoo to distribute their content via the Zattoo platform. In the B2C market, the revenues are depending on an advertising model. In the bittorrent case, the software providers and torrent sites only have an average amount of roles. However, they have little control as what is available on the bittorrent network depends on the preferences of the peer community. The users themselves put index information of the content on torrent sites and not the owners of the torrent sites themselves.

Third, we compared the roles of the end users over the selected cases. In all three cases, users were involved in content distribution which implies that users do not only share content but also the costs of content distribution. In the Kontiki and Zattoo case, it is interesting to note that the users are rather passive as they are only involved in content distribution and, of course, they consume the content in the end. Interestingly, users have no choice to upload content. If they download the P2P software and consume the content, then they upload this content as well. Furthermore, the Zattoo and Kontiki users do not have control of the content they have downloaded as they are being restricted where, how and when they are able to consume the content. In the case of Kontiki, we have to remark that whether or not users can have control of the content depends on the requirements of the demanding content owner as Kontiki offers customer-related services. Probably, the limited freedom of end users in these two cases, results from an unresponsive attitude of the broadcasters towards granting more control to the end users. In contrast with the former two cases, the users of bittorrent clients are actively involved in the value network. Moreover, the users or peers have most control of the value network as this is a self-organizing community of peers: the users introduce the content online; decide what is popular and what is not popular. The bittorrent peers have full control of the content once it is downloaded as the content is physically available on their hard drive, which implies that they can decide when, how and where they are going to consume it. Once downloaded, they can choose whether or not to keep the content uploadable on the bittorrent network. The disadvantage of this self-organizing community is that it is an absolute best effort model with no guaranteed QoS, and there is a lot of fake or corrupted content available.

The current findings reinforce the view that content owners are still determined to maintain control of what end users can do with their content. Despite all the utopian voices that have been raised, announcing a new era where the end users are in control, the value networks of the "legal", commercial P2P systems seem to indicate that content owners do not show any inclination to grant more control to end users. This probably results from the requirements of the content owners who elaborate versioning strategies and who want to prevent piracy as much as possible. Overall results of this study imply that end users are deprived of control in the legal, commercial cases, whereas the end users rule over the value networks of illegal file sharing activities.

In addition, it is remarkable that the two "legal" cases – i.e., Kontiki and Zattoo – both have hybrid P2P architectures, whereas the "illegal" case has a decentralized architecture. Again, this might be explained by the importance the media industry attaches to control, which relates to several issues. Indeed, hybrid P2P systems, which have some centralized components, allow the industry to control and monitor traffic more easily in comparison with decentralized P2P networks. The business models force the media industry to be able to monitor user behavior for instance to support their advertising model or to implement a payment model. Moreover, the industry wants to provide services with guaranteed performance levels in terms of availability and speed. Another important issue in legal, commercial content delivery is security as the industry does not want its customers to be confronted with viruses or malicious nodes. These are all issues in decentralized systems that might be solved by implementing hybrid P2P architectures. Therefore, the needs of the media industry – regarding control, security, QoS, business model – cannot be gratified by decentralized self-organizing communities. This explains why legal, commercial P2P systems generally require hybrid architectures.

The hybrid architectures belong to the unidirectional mass communication model (e.g., Zattoo) and the interactive mass customization model (e.g., Kontiki). Indeed both of these interaction architectures are (partially) centralized and controlled by the media industry. On the other hand, the bittorrent clients, which are mainly used for illegal purposes, belong to the multilateral community model as users control the content.

The results of the analysis indicate that revenue models have an important impact on the configuration of value networks. For that reason, a promising line of study would be to explore and to suggest direct and indirect revenue models for P2P systems. A possible revenue model might consist of, for instance, revenue sharing between content owners and P2P providers. In another conceivable revenue model, users might be remunerated for uploading content since users, which participate in uploading, are actually participating in cost sharing as their resources are utilized by other peers. For instance, former In2Movies, which does not exist anymore, allowed users to assemble points by uploading content with which they could buy content with a discount.

As we explained in Section 2.2, value networks cover only one of the four pillars of business modeling. Therefore, further research should be done to investigate the relation of value networks with the three remaining pillars – i.e., the functional architecture, the financial model and the value proposition – in the ecosystem in which P2P providers operate.

Finally, the convergence of the Internet and TV, which resulted in the OTT TV paradigm, made P2P technology a partner for the industry rather than an enemy. This results from the fact that P2P systems demonstrated to be scalable models with low marginal costs in terms of bandwidth consumption. In this way, the OTT TV paradigm led to the arrival of new P2P systems to distribute content in a legal, commercial way. However, the fact that end users are not bound by restrictions in illegal P2P file sharing communities will remain a seed-bed for the success of systems such as bittorrent clients.

Acknowledgement

FLEET – an interdisciplinary research project in FLEmish E-publishing Trends – is an IWT SBO project, with research partners IBBT SMIT, Cemeso, LSTS, MOFI, ICRI, CUO IBBT, MICT, ECDC and TNO. Additionally, we would like to thank Dr. Verónica Donoso for many useful comments on this chapter.

References

1. Ali R (6 May 2008) VeriSign’s Sale of Kontiki: $1 Million plus Stake in New Business. Available via PaidContent.org. `http://www.paidcontent.org/entry/419-verisigns-sale-of-kontiki-1-million-plus-stake-in-new-business/`. Cited 6 Aug 2008

2. Anderson N (13 Aug 2007) ISPs to BBC: We will throttle iPlayer unless you pay up. Available via Ars Technica. http://arstechnica.com/news.ars/post/20070813-isps-to-bbc-we-throttle-iplayer-unless-you-pay-up.html. Cited 22 May 2008
3. Ballon P (2006) Business Modeling for ICT Products and Services: Conceptual Framework and Design Criteria [Report]. In: Limonard S (ed) Integrated Methodological Framework: Investigating Business Models for Broadband Services – the Case of iDTV, TNO, Delft
4. Ballon P (2007) Business Modeling as the Configuration of Control and Value. Paper presented at the 20th Bled eConference, Bled, Slovenia.
5. Benkler Y (2006) The Wealth of Networks: How Social Production Transforms Markets and Freedom. Yale University Press, London
6. De Boever J (2008a) Value Networks of P2P TV: An Analysis of Actors and their Roles. In: Mellouk A, Bi J, Ortiz G, Chiu DKW, Popescu M (eds) Proceedings of The Third International Conference on Internet and Web Applications and Services, IEEE Computer Society, Los Alamitos
7. De Boever J (2008b) Peer-to-Peer Content Distribution: An Analysis of the Internal and External Potentials and Obstacles. In: Cordeiro J, Filipe J, Hammoudi S (eds) Proceedings of the Fourth International Conference on Web Information Systems and Technologies, INSTICC, Portugal
8. Einhorn MA, Rosenblatt B (2005) Peer-to-Peer Networking and Digital Rights Management: How Market Tools Can Solve Copyright Problems. Policy Anal 534:1–22
9. European Interactive Advertising Association (2007) EIAA Mediascope Europe Study. Available via EIAA. http://www.eiaa.net/Ftp/casestudiesppt/EIAA%5FMediascope%5FEurope%5F2007%5\Flaunch%2Epdf. Cited 5 Jun 2008
10. Fetscherin M (2006) Digital Rights Management: What the Consumer Wants. J Digit Asset Management 2(3/4):143–149
11. Gehrke N, Schumann M (2003) Constructing Electronic Marketplaces using Peer-to-Peer Technology. Paper presented at the 36th Hawaii International Conference on System Sciences, Big Island, Hawaii
12. Haßlinger G (2005) ISP Platforms Under a Heavy Peer-to-Peer Workload. In: Steinmetz R, Wehrle K (eds) Peer-to-Peer Systems and Applications, Springer-Verlag, Berlin Heidelberg
13. Hummel T, Muhle S, Schoder D (2005) Business Applications and Revenue Models. In: Steinmetz R, Wehrle K (eds) Peer-to-Peer Systems and Applications, Springer-Verlag, Berlin Heidelberg
14. IDC (2007) The Expanding Digital Universe: A Forecast of Worldwide Information Growth Through 2010. Available via EMC. http://www.emc.com/about/destination/digital_universe/pdf/Expanding_Digital_Universe_\Executive_Summary_022507.pdf. Cited 4 Apr 2007
15. Lechner U, Hummel J (2002) Business Models and System Architectures of Virtual Communities: From a Sociological Phenomenon to Peer-to-Peer Architectures. Int J Electron Comm 6(3):41–53
16. Lechner U, Hummel J, Eikemeier C (2003) Business Model Peer-to-Peer – Is there a future beyond file sharing? Paper presented at the First Conference on Design, Analysis and Simulation of Distributed Systems, Orlando, Florida
17. Liebau NC, Pussep K, Graffi K, Kaune S, Beyer A, Jahn E, Steinmetz R (2007) The Impact of the P2P Paradigm on the New Media Industries. Paper presented at the Americas' Conference on Information Systems, Keystone, Colorado, USA
18. MacInnes I, Hwang J (2003) Business Models for Peer to Peer Initiatives. Paper presented at the 16th Bled Electronic Commerce Conference, Bled, Slovenia
19. Norton WB (2007) Video Internet: The Next Wave of Massive Disruption to the U.S. Peering Ecosystem. Paper presented at the Stanford Networking Seminar, Stanford, USA

20. Osterwalder A, Pigneur Y (2002) An e-Business Model Ontology for Modeling e-Business. Paper presented at the 15th Bled Electronic Commerce Conference – e-Reality: Constructing the e-Economy, Bled, Slovenia.
21. Osterwalder A, Pigneur Y, Tucci CL (2005) Clarifying Business Models: Origins, Present, and Future of the Concept. Commun AIS 16(2005):1–25
22. Picard RG (2000) Changing Business Models of Online Content Services: Their Implications for Multimedia and Other Content Producers. Int J Media Management 2(2):60–68
23. Porter ME (2001) Strategy and the Internet. Harvard Bus Rev 79(3):62–78
24. Rodriguez P, Tan S-M, Gkantsidis C (2006) On the Feasability of Commercial, Legal P2P Content Distribution. ACM SIGCOMM Computer Commun Rev 36(1):75–78
25. Rupp P, Estier T (2003) A Model for a Better Understanding of the Digital Distribution of Music in a Peer-to-Peer Environment. Paper presented at the 36th Hawaii International Conference on System Sciences, Island of Hawaii, (Big Island), Hawaii
26. Slocombe, M (16 May 2005) BBC iMP: Public Trial For 5,000 in September. Available via Digital Lifestyles.
    `http://digital-lifestyles.info/2005/05/16/bbc-imp-public-trial-for-5000-in-september/`. Cited 16 May 2005
27. TimeWarner (14 Nov 2005) AOL.com Is First Major Portal to Deliver DVD Quality Videos (Press Release). Available via TimeWarner.
    `http://www.timewarner.com/corp/newsroom/pr/0,20812,1129718,00.html`. Cited 10 Oct 2007
28. Timmers P (1998) Business Models for Electronic Markets. Electron Markets J 8(2):3–8
29. Van Audenhove L, Delaere S, Ballon P, Van Bossuyt M (2007) Changing Content Industry Structures: The Case of Digital Newspapers on ePaper Mobile Devices. Paper presented at the 11th International Conference on Electronic Publishing, Vienna, Austria
30. Werbach, K (2008) The Implications of Video Peer-to-Peer on Network Usage. In: Noam EM, Pupillo LM (eds) Peer-to-Peer Video: The Economics, Policy, and Culture of Today's New Mass Medium, Springer Science + Business Media, LLC, New York
31. Wierzbicki A, Goworek K (2005) Peer-to-Peer Direct Sales. Paper presented at the Fifth IEEE International Conference on Peer-to-Peer Computing, Konstanz, Germany
32. Xie H, Krishnamurthy A, Silberschatz A, Yang YR (2007) P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers. P4PWG Whitepaper. Available via DCIA.
    `http://www.dcia.info/documents/P4P_Overview.pdf`. Cited 5 Nov 2008
33. Xie H, Yang YR, Krishnamurthy A, Liu Y, Silberschatz, A (2008) P4P: Provider Portal for Applications. ACM SIGCOMM Computer Commun Rev 38(4):351–362

Live Video and IP-TV

Maria Luisa Merani and Daniela Saladino

Abstract This Chapter aims at providing a comprehensive insight into the most recent advances in the field of P2P architectures for video broadcasting, focusing on live video streaming. After introducing a classification of P2P video solutions, the first part of the Chapter provides an overview of the most interesting P2P IP-TV systems currently available over the Internet. It also concentrates on the process of data diffusion within the P2P overlay and complements this view with some measurements that highlight the most salient features of P2P architectures. The second part of the Chapter completes the view, bringing up the modeling efforts to capture the main characteristics and limits of P2P streaming systems, both analytically and numerically. The Chapter is closed by a pristine look at some challenging, open questions, with a specific emphasis on the adoption of network coding in P2P streaming solutions.

1 Live Streaming and IP-TV

Traditionally, video delivery over the Internet relies upon the consolidated client-server paradigm. In this perspective a centralized server has to be accessed by all clients that wish to download the desired video stream, as Fig.1 illustrates.

This simplified view immediately suggests that the server access bandwidth is the most limiting factor against system scalability, at least in terms of network resources. Referring to the case of constant bit rate connections, when the number of concurrently active clients increases and the sum of the bandwidths that their flows

Maria Luisa Merani
Department of Information Engineering, University of Modena and Reggio Emilia, Italy,
e-mail: marialuisa.merani@unimore.it

Daniela Saladino
Department of Information Engineering, University of Modena and Reggio Emilia, Italy,
e-mail: daniela.saladino@unimore.it

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_35, © Springer Science+Business Media, LLC 2010

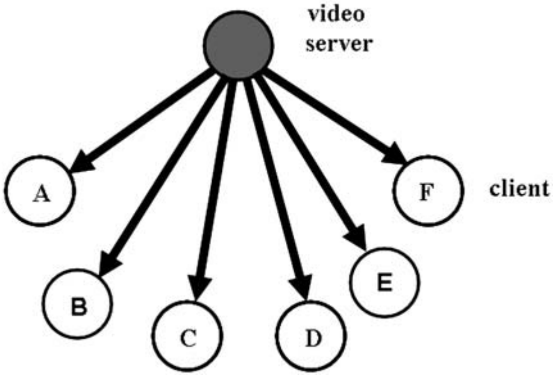


Fig. 1 Conventional client-server architecture

require equals the server access bandwidth, then the video streaming system saturates. No more users can be supported, otherwise congestion will soon appear and markedly penalize the throughput of the video application.

More generally, the client-server approach entails that the video server be the edge of as many unicast connections as the number of clients: one video stream per client is individually and separately taken to destination, consuming bandwidth and network resources, and possibly generating congestion along all traversed paths. Definitely, not a smart solution for several video applications: it suffices to think of the broadcasting of television events, where the same information has to be simultaneously delivered to each subscriber.

Multicasting at the IP-layer, probably the cleanest solution from a conceptual viewpoint, was first proposed to relieve the problem, but IP multicast never took place over the global Internet [32]. The violation of the stateless principle of IP routers, the lack of scalability, the increased difficulty in performing congestion and error control at transport layer on multicast connections were probably the technical factors that mostly limited its inception.

One of the novelty of P2P for video delivery over the Internet resides in moving the multicast approach to the application layer. Another unique point being that it is up to the same end-users of the multicast group to collaborate in the process of swarming the information across the network to other users: a node in the overlay will not only receive the desired video, but will also cooperate to distribute the video to other peers. As we will see in next Section, the way video information propagates across the peer overlay provides the most relevant metric to classify P2P systems for video delivery.

In P2P solutions the video stream is divided into relatively small, equal size chunks, that typically contain a few seconds of the original video. Participating peers make some of their untapped resources available to the system, most notably their upload bandwidth, to pass the video chunks they already own to other peers: this greatly relieves the burden on the original streaming server. Moreover, the infrastructure requirements of the application-layer approach are minimal, and this really makes P2P attractive for video distribution over the Internet, for a variety of heterogeneous services.

To conclude this introductory discussion, observe that P2P had already gained its slice of popularity in file sharing applications, well before it was extended to video delivery. It is however worth underlining that the migration of the P2P approach to this new realm was not painless: video applications exhibit peculiar features, most prominently the real-time constraints that the great majority of them imposes on information delivery; also, video services are and will be bandwidth eager, by far a more challenging characteristic than VoIP, another very common real-time application.

Hence, the focus shifts: whereas efficient indexing and searching techniques are of paramount importance in P2P systems for file sharing, a careful scheduling to minimize delays is required when the P2P overlay handles video, as well as a satisfying resilience to peer churning, i.e., to a swift increase/decrease in the number of peers within the overlay. The ultimate goal is to warrant a satisfying and constant Quality of Experience (QoE) to the peers viewing the video.

2 A Taxonomy of P2P Video Broadcasting Systems

Video applications over the Internet span quite a broadcast range: from videoconferencing, (imposing very strict time constraints), to live video streaming with nearly synchronized users (this is what we will mainly refer to in what follows, interchangeably using the IP-TV and video broadcasting terms), to video on demand (the most delay-tolerant category).

Let us illustrate the main classification of the P2P systems devised for IP-TV, employing the approach to overlay construction as the sorting criterion. P2P solutions are accordingly distinguished in tree-based and mesh-based architectures [18]. In the former the video stream propagates from the source via a tree of peers, so that the video gradually spreads over the entire overlay, from the root of the tree to its last leaves. Recall that the tree is built at the application layer, and relies on the underlying unicast IP connections.

Figure 2 illustrates the idea: in this example, peers A and B receive the video from the source root and then forward it to peers C and D, E and F, respectively; in turn, peer C is in charge of delivering it to leaf peers G and H, peer F to leaf peer I.

This solution is also termed push-based, as the video is pushed along the tree, a topology that naturally meets the multicasting demand. How is the tree formed? When a new peer has to join it, bandwidth and delay are the indicators that typically drive the choice of the parent node: the peer can select the parent depending on the round trip time from it, on the application throughput the parent experiences, on its uploading bandwidth. Avoiding loops is the constraint to respect.

Easy as it appears, this solution unfortunately exhibits several drawbacks: it is prone to outages, as the departure of a non-leaf peer from the overlay deprives all its descendants of the video content; it does not utilize the upload bandwidth of

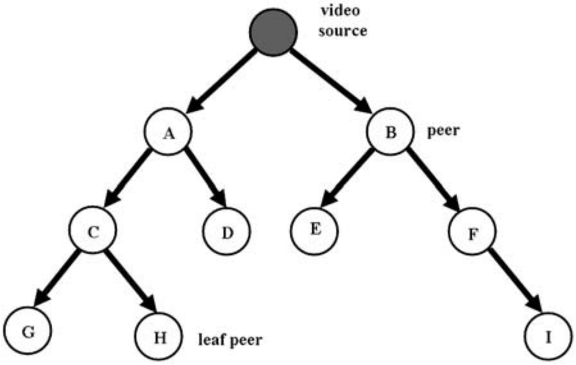


Fig. 2 Tree based P2P overlay

leaf peers, that only receive content, but do not collaborate in distributing the video across the overlay.

Next natural step is therefore to resort to a multi-tree overlay, where peers that are leaves in a tree are not so in a different one, and robustness and better efficiency is achieved via multiple, disjoint trees. Figure 3 reports a simple example of a multi-tree topology, with differently shaded arrows identifying different trees. In the multi-tree topology, the source encodes the video stream into multiple substreams, each substream flowing on a different tree. A peer typically joins more trees, depending on its access link bandwidth, and experiences a quality that depends on the number of substreams it receives. The push mechanism of the single tree topology is retained, as packets belonging to one substream are simply forwarded from the parent node to its children peers along the corresponding tree.

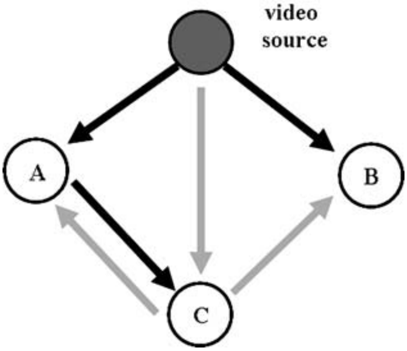


Fig. 3 The multi-tree concept

The main objective to pursue in this articulated topology is to build – and maintain – short and balanced trees. It follows that peers often need to be dynamically reallocated within trees, due to the events of peer joining and departing.

The approach is completely different in mesh-based overlays, also termed pull-based architectures . There is no predefined topology here, no a priori notion of trees for data delivery. Rather, each peer maintains a list of partners and periodically exchanges with them information about the available data. It then "pulls" the desired blocks of video from one of the peers that advertises them, also supplying available data to other partners. Partnerships are updated at a proper rate, to ensure both robustness to failures and efficiency in the data diffusion process.

At first sight, the swarming process of this solution closely resembles popular P2P systems for file sharing like BitTorrent. There is however a significant difference: video has to obey strict time requirements, its blocks need to be delivered without suffering an unbearable delay and jitter; if it were not so, quality would be unfavorably affected. It follows that the scheduling peers adopt to pull the blocks from their parents is significantly different from the ones implemented in P2P file sharing architectures: it has to minimize delay, so as to guarantee that the majority of the downloaded chunks respect the playback deadline.

Some among the currently most popular P2P systems fall within the mesh category, and we have chosen to describe their main features in greater detail in next Section.

What is the best solution between mesh and tree? We anticipate here that both architectures enjoy benefits and drawbacks: the pull-based overlay, the most diffused in commercial systems, is simple to implement and to maintain; it is efficient, as data forwarding is not restricted to specific directions; it is resilient to swift peer dynamics [6]. Yet, it often suffers significant delays at start-up time and when switching channel, as well as non negligible time lags between peers viewing the same video. The time spent by the peer in the process of information exchange, required to know which partner owns which information, is responsible for a large fraction of the service delay the peer experiences [13]. On the contrary, delays are definitely confined in tree-based systems, but the signaling overhead and the complexity in system design and management, to guarantee stable and resilient trees, is not to be underestimated [18].

So far, we have assumed that both tree and mesh-based overlays be organized in a flat, non hierarchical manner. Recently however, an architecture has been proposed, where peers are structured in tiers [26]. The starting point of the proposal is that in a mesh-based overlay not only most of the data blocks propagate via tree structures, but also the majority of them is delivered via an implicit stable backbone [26].

The solutions the authors put forth is therefore a layered architecture, where a first backbone tier of more stable peers with sufficient access bandwidth is organized in a tree structure, and serves more fluctuating nodes, that are placed in a second tier; this second tier can accommodate diverse overlay structures, i.e., either mesh or additional trees.

It therefore appears useful to further distinguish P2P system topologies in flat and layered categories. This classification also turns out convenient when discussing hybrid architectures, that combine the adoption of the P2P paradigm and of content replication servers, strategically placed over the Internet: these systems can well be framed within the layered category, provided the nodes of the tier-1 backbone represent the stable, always available servers, as opposed to the less predictable end-user peers of the second tier.

Last, P2P architectures for IP-TV can be distinguished on the basis of the number of their potential users, classifying P2P systems in small or large size overlays. P2P architectures serving the needs of prosumers (producers and consumers), wishing to broadcast their own video content fall within the first class, and are separated from large P2P systems tailored to the requirements of major TV broadcasters and service content providers. Within the first category, pure P2P overlays, that do not rely upon the presence of content delivery servers, represent an appealing solution; on the contrary, hybrid systems represent the most popular proposal when scalability, as well as reliable service provisioning, are the major constraints.

3 Popular P2P Streaming Systems

There are several P2P systems that are worth being cited, most notably because they experience wide commercial success and rely upon a large basis of users. Among them, we mention PPLive [21], SopCast [25], UUSee [31], GridMedia [9], offering real-time services, and Joost [14] and Babelgum [5], offering Video-on-Demand (VoD) services. All of these systems are proprietary, have received a great success all over the world and their majority has been developed in China.

In what follows we briefly glide over their main features, beginning with PPLive [21]. It is one of the most popular P2P live streaming systems, born in China in 2004. It belongs to the mesh-based family, employs a pull-based protocol for video content transmission and mostly relies on TCP. It offers more than 200 channels, has an average of 400,000 daily users and the bit rate of its video programs ranges from 250 kbit/s to 400 kbit/s. It also offers a few channels at a rate of 800 kbit/s. Channels are encoded with two video formats: Window Media Video (WMV) or Real Video (RMVB) [2]. In PPLive the number of partner nodes of every peer depends on the popularity of the selected channel and on the peer's access type: peers with high bandwidth access (also termed campus peers) have about 40 partners; peers with residential access have a number of partners that ranges from about 10 to 30.

SopCast [25] is another P2P streaming application that provides both VoD and live services, born in China in December 2004. It was able to support more than 100,000 concurrent users only a few months after its introduction. It employs a mesh-based architecture and mostly relies on UDP. In contrast to PPLive, both residential and high bandwidth access peers typically download from 2 to 5 other peers.

UUSee [31] is an additional instance of very large scale P2P streaming solution and was developed in China too. It has several streaming servers around the world, simultaneously offers more than 800 channels, with 100,000 concurrent users, and provides a streaming rate of 400 kbit/s. It belongs to the mesh-based family and employs the pull-based approach. It relies on TCP for data transmission and the number of partners for each user is around 50.

GridMedia [9] is still a Chinese large scale P2P live streaming system (there is no doubt that China gets the lion's share in this field!), implementing a hybrid push-pull protocol. It is able to support more than 224,000 simultaneous users and its streaming rate is 300 kbit/s. The streaming server can support up to 800 connections [30], therefore reaching 240 Mbit/s outgoing bandwidth at server side.

Finally, there are two interesting P2P Video-on-Demand (VoD) streaming solutions, both conceived in Europe: Joost and Babelgum.

Joost [14] is a VoD P2P system for distributing TV content. It was created by Niklas Zennstroem and Janus Friis, the Skype founders, in 2006. It relies on a mesh-based P2P streaming overlay and every peer receives 95% of the video frames from about 25 peers. Joost employs mostly UDP as transport protocol. In particular, UDP is used to transmit data packets and TCP for control messages only. An important feature of Joost is the adoption of a NAT detection mechanism in order to improve system performance: if some peers lie behind the same NAT device, they tend to transmit video content to each other.

Babelgum [5] is a Video-on-Demand P2P streaming system too, conceived by Silvio Scaglia in 2005. It employs TCP for both control and data packets distribution. Every peer receives 95% of the video frames from about 8 peers.

Both systems, Joost and Babelgum, exhibit a hybrid architecture. Their behavior is not purely P2P: rather, they resort to the help of some streaming servers located around the world to distribute the video contents.

4 The Diffusion Process and a Reference System

4.1 Mesh-Based Systems

As promised earlier, we now concentrate on the main characteristics exhibited by an unlayered, mesh-based overlay. The focus is primarily on CoolStreaming, [6, 12, 19], one of the most popular pull-based systems (at least in its original version), generally referenced as the benchmark among the scientific community.

Its developers initially preferred to describe its design as a "data-driven" approach, rather than mesh-based: indeed, no specific overlay structure confines the data flow direction.

The concept behind a mesh system is simple: peers help each other and cooperate in the delivery process, exchanging video chunks. For a generic pull-based architecture it can be affirmed that, when a new peer joins the system because it desires to view the video, the peer first contacts the origin node, where the original video is available. In the simplest case, what happens next is that the origin server redirects the new peer to a tracker, maintaining a membership list: the tracker provides the peer a set of potential partners, that the new node contacts to establish the relationships required to start receiving video chunks. As soon as such relationships are set, the peer starts receiving the buffer maps of its partners, i.e., short control messages that every peer periodically forwards to indicate its available chunks. The new peer will in turn forward its buffer maps, although at the very beginning they will reveal that the node has no content to share with other participants of the overlay. From the buffer maps it receives, the new peer can identify its potential *parents*, i.e., the most appropriate partners from which to start fetching the video chunks via a proper scheduling algorithm.

Note that there is a signification distinction between the terms overlay members, partners and parents [12]: the first term identifies all end-users in the overlay, wishing to view the same video; the second term refers to the peers that exchange information with the reference peer about chunk availability via their buffer maps; the third term indicates what peers (the parents) are actually providing video content to the peer (the child). The mutual relation between overlay members, partners and parents is graphically represented in Fig. 4.

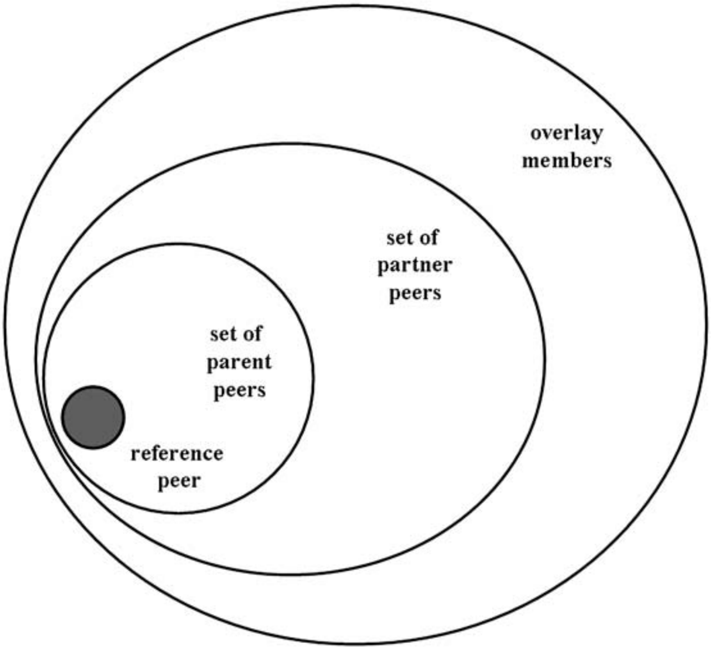


Fig. 4 Overlay members, partners and parents

As for the buffer map, its simplest structure hosts a string of zeros and ones, where the one indicates that the chunk is available at the peer, whereas the zero tells that the peer does not own the chunk; a proper offset allows to identify the first chunk of the sequence that the map refers to. This is exactly the description of the buffer maps in the original CoolStreaming prototype, that employed 120 bit long strings. Given a chunk contains one second of video, we immediately conclude that there are a couple of minutes available in the peer buffer, a typical order of magnitude for most P2P systems.

4.2 Scheduling

After receiving the buffer maps, the peer can request the chunks that it is missing and that other peers advertise. The request scheduling is a delicate issue: video chunks have to meet severe playback deadlines; if they do not, their late arrival translates into a loss, and ultimately in a degraded quality experienced by the end-user viewing the video. This constraint translates in rejecting the classical round robin scheduler, in favor of heuristics that keep the number of late or lost chunks to a minimum, possibly equal to zero.

As an example, we refer to the scheduling algorithm of the first CoolStreaming release [6], that starts by determining all potential suppliers of every chunk. The chunks with fewer suppliers are considered first, and for each chunk, a unique supplier is identified: it is the one with the highest bandwidth and enough available time to transmit the chunk. Once a schedule is specified, an individual bit sequence resembling the buffer map is sent to each supplier, indicating the chunks that the peer intends to pull from it. The video chunks are then delivered to the requesting peer via UDP connections, properly enhanced by the congestion control mechanism TFRC implements [7, 29].

In passing by, we mention that TCP is alternatively adopted by some P2P streaming systems for transporting video data, despite its overhead in terms of connection opening and closing phase, as well as retransmission handling.

4.3 Software Architecture and Overlay Membership

Let us now more thoroughly dissect the software architecture of a peer, to logically frame the different modules that constitute it: following the description in [13], in the peer we distinguish the P2P streaming engine and the media player, as Fig. 5 indicates.

The streaming engine has the responsibility of fetching video chunks from partners; of storing the retrieved chunks in a buffer; of passing the chunks to the media player. It is also in charge of providing the available chunks to those peers requesting them and to manage the buffer maps. Finally, it continuously updates the partnership list. Referring again to the first CoolStreaming, each node periodically establishes new partner relations with randomly chosen partners: the node that provides the lowest average number of chunks per unit time is discarded and replaced by a new, better performing partner.

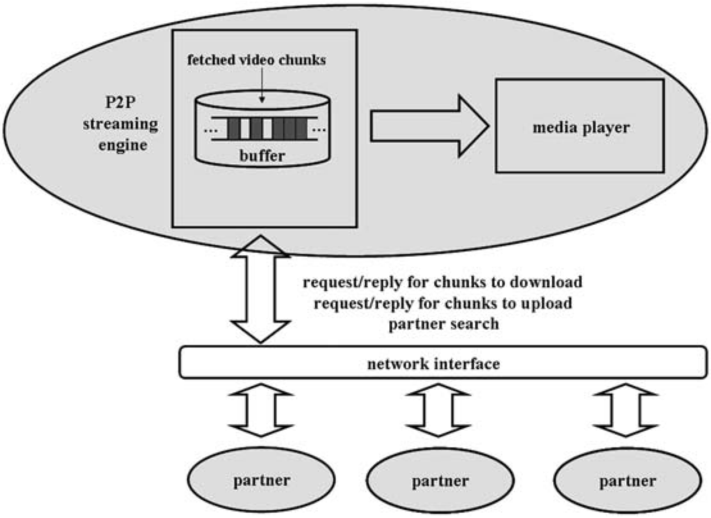


Fig. 5 Software components in a peer

An additional, last point deserves a further refinement in our discussion: the overlay membership management. In small to medium sized overlays, peers retrieve membership information directly from the tracker server, the unique repository of the system view: this is exactly what we have assumed in our first discussion. It is not so for large size P2P streaming systems, such as CoolStreaming: here a new node joining the system contacts the source, that redirects it to another peer, called deputy peer (rather than to the tracker server), randomly selected from its own membership cache. The deputy (i) provides a list of eligible partners, that the new peer contacts to establish its partner relationships, and also (ii) updates its own cache to record the entry of the new peer. This redirection guarantees a more uniform partner selection, as well as a reduced load over the video source. What is interesting to observe is that in each peer of the overlay there has to be not only a partnership cache, but also a membership cache.

Why is this cache present in all peers and how is it managed?

The answer to the first question is important: the ultimate goal is to disseminate among all peers a uniform, although forcedly partial view of the overlay members. This is necessary for the deputy functions we cited above, but also because each peer periodically consults its membership cache to replace partners, either when some of them depart or when some better performing nodes become available. This happens in a decentralized manner, without placing any burden on the origin server.

The answer to the second question is that a gossiping protocol is employed to create and update the membership cache, which in turn triggers the explanation of what we mean by the term gossip-like algorithm. A gossip algorithm, also tagged as epidemic, presents the following characteristics: a peer sends a new message (in our case it periodically announces it exists) to a random subset of other peers; in the next round these peers propagate the message in the same manner, and so do next peers that receive it. Gradually, in a totally distributed manner, the information the peer exits propagates in the overlay, contributing to the construction of the local view of the overlay members at each peer that receives it.

When discussing the weaknesses of mesh-based systems, we already evidenced that excessive initial delays are expected to plague this architecture: indeed, CoolStreaming earlier release had Achille's heel of long startup delays, as well as a high failure rate in joining a channel during a peer churn [12].

The diffusion process it adopted was then radically modified, and we intentionally describe it next, with the intent to understand how CoolStreaming and in general P2P streaming overlays have to evolve to fulfill commercial system requirements.

4.4 New CoolStreaming

The New CoolStreaming adopts a multiple substream solution to better swarm the video among its peers; in turn, this calls for a fundamental change in the architecture of the peer buffer, in its management, as well as in the scheduling scheme; moreover, a new, hybrid push-pull mechanism that the peers adopt to download video chunks is developed. Last but not least, CoolStreaming now employs multiple servers, strategically located. Equivalently, the system does not rely on a pure P2P overlay any longer; rather, it now adopts a hybrid architecture, where the streaming capacity is proportionally amplified with the number of servers, and the content is taken closer to the end-users.

Let us see in detail the multiple substream solution and the novel buffer organization at the peer. As before, the video is divided in blocks of equal size; the novelty relies in splitting the blocks in different substreams, as Fig. 6 reports. Video chunks are grouped according to the following pattern: given that K substreams have to be formed, the j-th substream ($j = 1,2,\dots,K$) is made of the chunks with the following sequence number in the original stream: $j + i \cdot K$, $i = 0,1,2,\dots$.

A peer can subscribe to one or more substreams, fetching the corresponding chunks from multiple parents. Although no specific coding technique is employed, the critical point resides in maintaining the synchronization among different substreams.

The structure of the peer buffer has to be accordingly modified: a synchronization buffer precedes the cache buffer and hosts the chunks received from each substream, sequentially ordered, as the example in Fig. 7 shows. The chunks are then combined in a single stream as soon as chunks with contiguous sequence numbers are received from each substream.

In the example of Fig. 7 the video chunk with sequence number 11 is yet to be received and therefore the combination process halts at this point in the sequence.

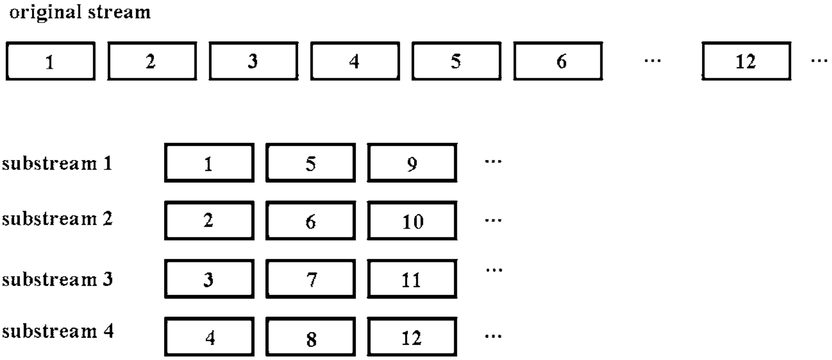


Fig. 6 Decomposition of the original video stream in K substreams ($K = 4$ in the considered example)

As for the buffer maps exchanged by peers in the New CoolStreaming system, these too have undergone a significant rethinking: a buffer map now specifies not only the chunk availability at the peer, but also the peer current requests. In greater detail, the map is composed of two vectors, whose dimension is equal to K, the number of substreams: the elements of the first vector indicate the sequence number of the last chunks the peer received for every substream; the elements of the second vector specify what substreams the peer wants to subscribe to.

We close this shot with the description of the new content delivery mechanism adopted by CoolStreaming. As previously anticipated, peers in the original CoolStreaming prototype had to deliberately fetch – pull – each chunk from other peers. The revisited architecture inherits the pull mechanism only for the first chunk of the requested substream; from then onward, the parent peer will keep continuously forwarding – pushing – all chunks to the requesting node.

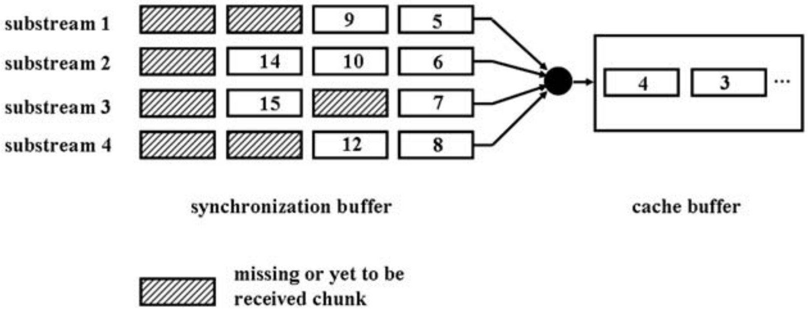


Fig. 7 Logical organization of the peer buffer when $K = 4$ substreams are considered

Periodically, parent peers are updated, to replace nodes that either departed, experienced a failure, or provided insufficient video content. We refer the interested reader to [12], as well as to [4], for details about the parent reselection process.

5 Measurements and Quality Monitoring

In this Section we are going to present a few significant measurements about some large P2P commercial systems, namely, PPLive, SopCast and CoolStreaming. We will also examine a small P2P architecture, featuring a moderate number of users.

Both measurements performed at network edge and at a central facility will be introduced and commented: the former are attained client-side, where the single peer itself can capture the local upload and download traffic and analyze it; the latter are performed server-side. The log-server has a complete view of the system: to cite a few significant records, number of peers connected, session duration, upload/download bandwidth of each peer, peers' IP address/port number.

There are several reasons for measuring and monitoring system parameters: perhaps the most important is to guarantee a satisfying viewing experience to system users. This is an essential feature for wide commercial success of any P2P streaming solution, that has to guarantee video playback continuity, without video freezing and skipping, in order not to discourage the users. If a user is unsatisfied he/she could decide to abandon the system. Therefore, the main goal is to optimize as much as possible the quality perceived client-side, the Quality of Experience (QoE).

Different factors could threaten the QoE: participating nodes heterogeneity, frequent peers churn, delays. In what follows we explain the most significant IP-TV quality metrics, that in turn have a deep impact on QoE: initial start-up delay, video switching delay, video playback continuity.

The *start-up delay* is the time interval between the instant a channel is selected and the instant when the video playback starts on the screen. This is a critical delay, with a straightforward influence on the viewing experience of the user.

If the user decides to watch another video, he/she switches channel and this causes the *video switching delay*, which is definitely than that experienced in traditional television.

As for *video playback continuity*, each video chunk has a playback deadline. Hence, if a certain video chunk is not in the buffer before its playback deadline, two situations can occur. If there are no video chunks in the buffer of the player, the user experiences freezing of the video, that is the playback of last video frame. If there are some video chunks in the buffer, the player plays them back although they might be not continuous: in this circumstance the user experiences skipping of the missed chunk's frames [13].

We now provide a glimpse at some specific attributes of the examined P2P systems, as can be inferred from local measurements. Next, their behavior and partly their performance will be discussed, with the help of the measures available at the central facility.

5.1 Network Edge Measurements

This subsection focuses on measurements performed client-side, displaying some of the features exhibited by PPLive, SopCast, as well as by an instance of a small P2P overlay.

The first measurements illustrate what transport protocol the different P2P streaming systems adopt. Not all streaming architectures employ the same transport protocol: some of them exclusively utilize TCP, others UDP only, and others both. In the latter case, the system typically uses TCP for control traffic and UDP to carry data packets.

In order to know what kind of transport protocol a system relies upon, it is sufficient to check how many TCP and UDP packets are exchanged among the local peer and its partner nodes. The results we obtained for SopCast and PPLive are graphically shown in Figs. 8 and 9, respectively. We can conclude that SopCast employs UDP to transport both video and control packets. In contrast, PPLive employs UDP for data and TCP to carry control traffic. Unlike UDP, TCP guarantees reliable packet delivery and enforces congestion control: this first feature is useful for signaling traffic, but not necessarily for live video traffic, that has strict time restrictions.

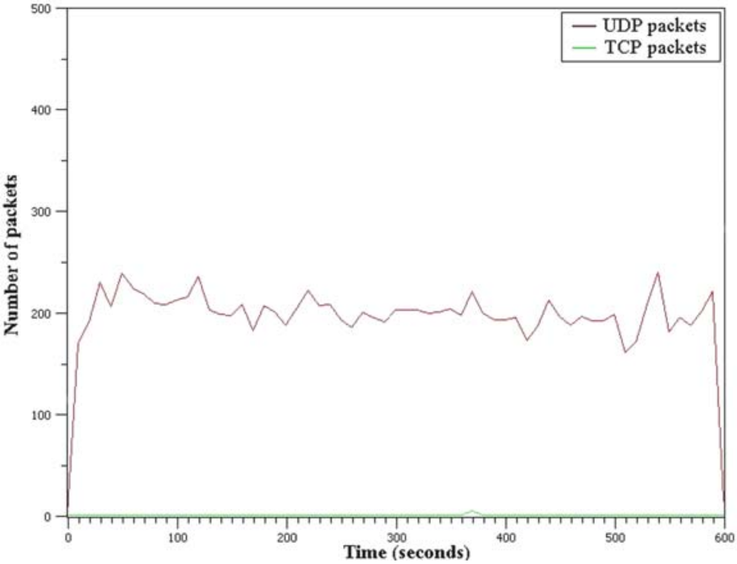


Fig. 8 Number of UDP and TCP packets exchanged in a time window of 600 s by a SopCast peer

We now turn our attention to the P2P application throughput : to locally assess it, we once more resort to traffic captures performed by the peer. The values of the upload and download throughput that the PPLive client achieves are graphically represented in Fig. 10 for an ADSL connection. As expected, the ADSL upstream and

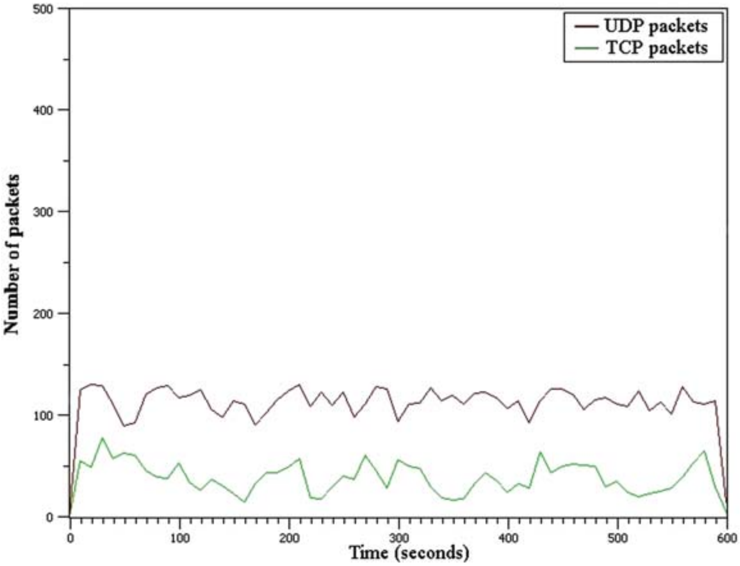


Fig. 9 Number of UDP and TCP packets exchanged in a time window of 600 s by a PPLive peer

downstream asymmetry directly reflects in the throughput values the P2P application exhibits. To be accurate, the traffic exchanged by the local peer and captured to execute this analysis contains not only video chunks but also control messages, useful to manage peer's partnerships, and, in particular, buffer maps. Yet, control packets represent a far smaller percentage with respect to data.

There are no great differences between PPLive and SopCast, whose channels have both a streaming rate of about 400 kbit/s, so we choose not to report the results referring to the latter system.

Next measurements are concerned with the download throughput that the local peer achieves via its best partners, i.e., the peers from which it receives most of the packets.The results obtained for PPLive are shown in Fig.11: this figure reports the total throughput of the local peer, as well as the contributions provided by the best four partners and by the best partner. We can observe that the local peer receives about one third of the total chunks from the best four partners and much less from the best partner. This suggests that in PPLive every peer has to connect to several other peers, among which the chunk requests are equally allocated.

Until now, we have presented some results referring to large systems, featuring a vast group of users. However, there exist P2P live video streaming solutions belonging to the small overlay category, that are equally worth being investigated.

We focus our attention on one of them, providing only a few TV channels: its floor of users is therefore quite moderate, typically of the order of a few hundreds.

The system we have examined is StreamerOne [28]: it is the first Italian P2P live video streaming platform, and currently provides only a few test channels. It is based on a mesh architecture, with no predefined topology. Every peer maintains a list of partners and periodically exchanges with them information about the available data, via the buffer maps. It then pulls the desired video blocks from one of the peers that advertises them.

StreamerOne architecture encompasses the presence of one control server and of various streaming servers, one for each broadcasted channel. When a peer joins the system, it receives from the control server the references to all streaming servers.

Once the peer has selected the desired channel, it receives from the corresponding streaming server a list of 10 peers to exchange video information with. Peers are selected randomly from the set of peers watching the same channel, and the worst contributor is periodically purged from the list and replaced by a new one.

Additionally, every peer receives from the control server a list of peers, not necessarily watching the same channel, with which to exchange further information about the system, such as name, number of users and current quality of all channels.

In the beginning, Streamerone broadcasted television channels with a rate of 160 kbit/s. Recently, its developers switched to an H.264 based transmission, with a 224 kbits/s rate, that guarantees better video quality.

For this system too, we take a look at a few interesting sets of local measures, namely:

- the number of partner nodes from which the peer receives data packets;
- the Cumulative Distribution Function (CDF) of the received packets length.

The first measurement is useful to understand how many peers cooperate with the user to visualize the requested video. Figure 12 shows that in one hour of experiment there are on average 5 other peers providing the local peer the desired data packets.

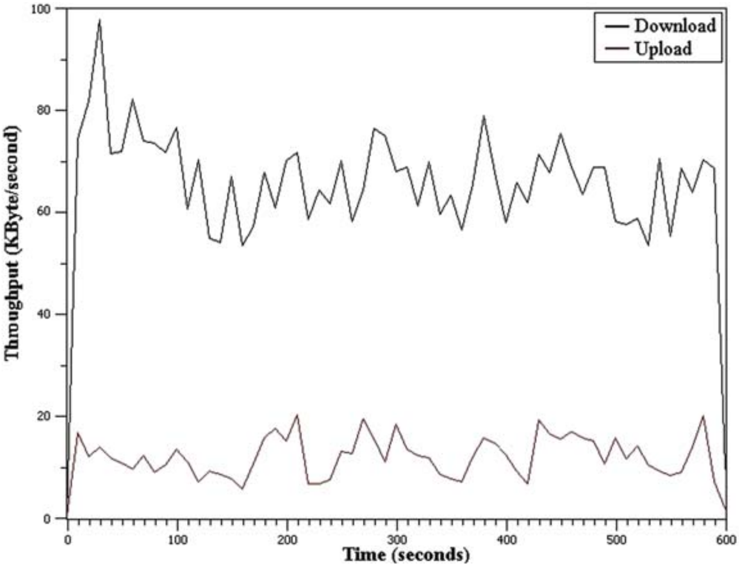


Fig. 10 Download and upload throughput of a PPLive peer

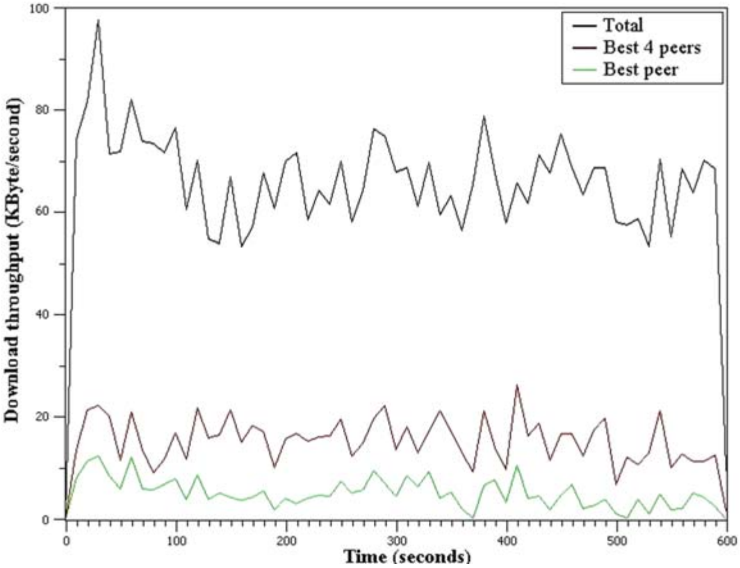


Fig. 11 Total and partial download throughputs of a PPLive peer

At first, the user receives data only from two peers, one of which is the streaming server. Then, as it starts receiving buffer maps, the local peer contacts more partners to request – and obtain – the chunks it needs.

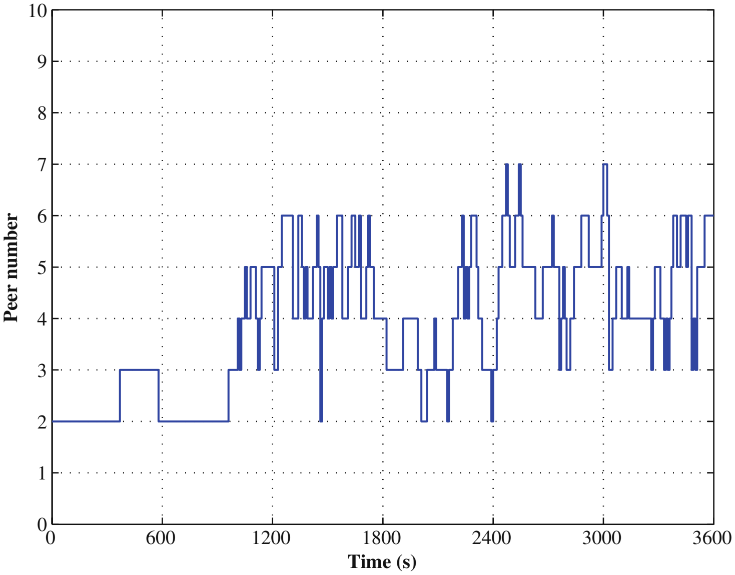


Fig. 12 Evolution of the number of partner peers in a small system

The second experiment aims at providing an indicative characterization of the traffic that the peer receives from and sends to the P2P network. Figure13 graphically shows the Cumulative Distribution Function (CDF) of the packet size, as derived from the local data. We can observe that the CDF follows a bimodal distribution, underlining two kinds of packets, large and small. The first ones convey data, have a size larger than 1500 bytes and sum up to 40% of the total number of packets. The second ones carry control information (conveying, e.g., buffer maps) and have a length that ranges from approximately 60 to 200 bytes. Although derived for the small P2P overlay, it is important to note that the bimodal distribution is typically observed in large systems too.

5.2 System Measurements

In this subsection we will comment some of the measurements that are typically performed server-side. The examined systems will be CoolStreaming, PPLive and the small P2P overlay considered earlier, StreamerOne. The main reference sources will be [4] for CoolStreaming and [2] for PPLive, as log-server traces for these systems are not publicly available.

One of the most common measurements carried out on P2P video streaming systems is the number of peers joining/leaving the system and the number of peers contributing in data streaming transmission evolution during time. Figure14 qualitatively reports a typical behavior for the number of simultaneous CoolStreaming users, as reported in [4], during the peak hours between 18:00 and 23:00 (6:00 and

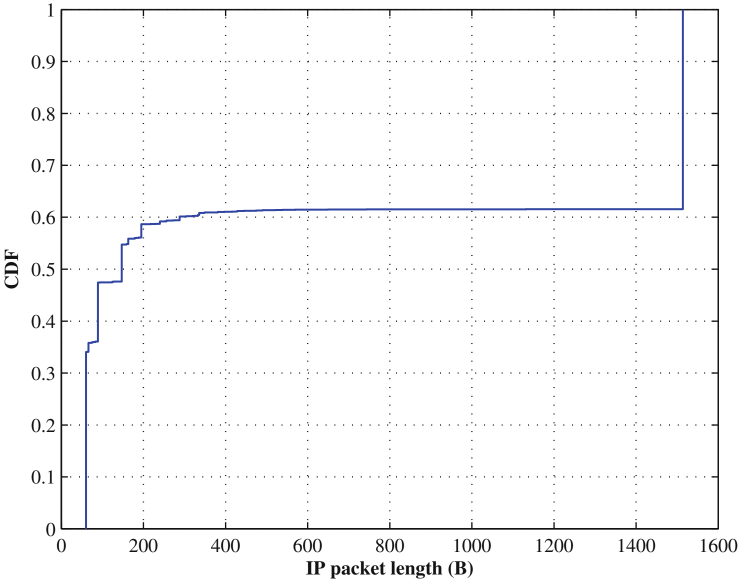


Fig. 13 Packet size CDF in a small system

11:00 p.m.). These results underline the great scalability of the P2P solution, which easily supports more than 40,000 concurrent users.

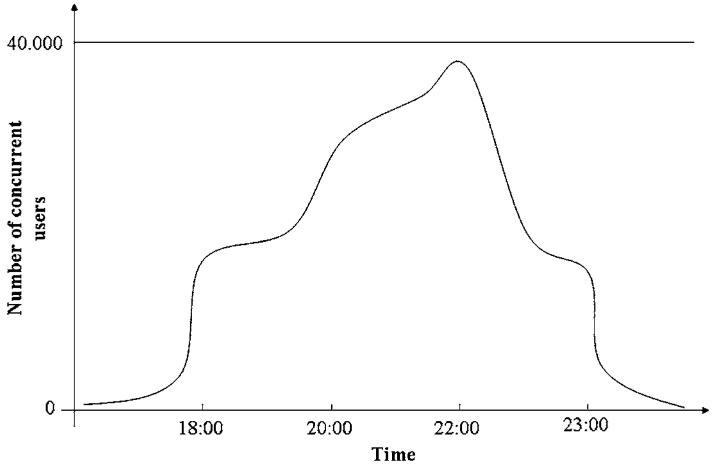


Fig. 14 Evolution of the number of simultaneous users in CoolStreaming

This type of measurement has been performed on PPLive too [2]. The number of peers for one of its popular television channels is qualitatively represented in Fig. 15. By observing the figure, we can say that the examined program reaches a large number of concurrent users, about 2,700. Moreover, the peaks of the users occur at 12 AM and 7 PM, suggesting that people tend to watch IP-TV *outside* office hours [2].

As expected, the number of concurrent users in the system is tightly correlated to the popularity of the program. For this reason, the authors of [2] have performed the same measurements during the Spring Festival Gala on Chinese New Year too, that is the most popular event in China. The results obtained emphasize a sudden increase of the users from 50,000 to 200,000 when the corresponding program starts, and a prompt decrease when the program ends. Again, these observations suggest that the P2P system scales well. An important feature, common to all P2P streaming systems, is the following: when a TV program ends, peers immediately and simultaneously leave the network, so that a batch-departure occurs. This phenomenon is not present in P2P systems for file sharing, where users depart at different instants [2].

A large scale P2P streaming system typically attracts millions of users from all over the world. Therefore, another relevant feature to monitor is the geographical distribution of the users . This measurement requires a comparison between every peer's IP address and a database containing all the associations between ranges of IP addresses and corresponding geographical areas. The results reported for PPLive in [2] are shown in Figs. 16 and 17, that show the user geographical distribution in a generic weekday and in the Spring Festival Gala day on Chinese New Year,

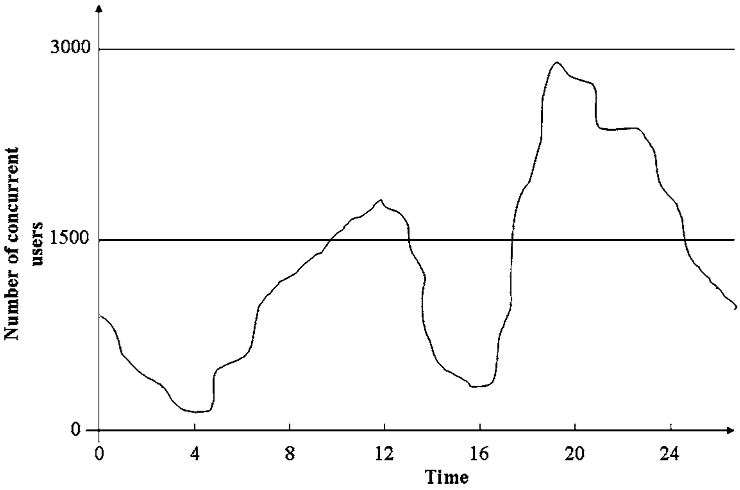


Fig. 15 Evolution of the number of simultaneous users in PPLive in a whole day

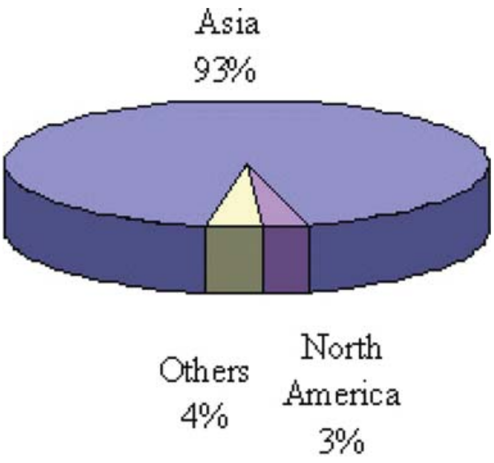


Fig. 16 Users geographical distribution in PPLive in a generic weekday

respectively. In Fig. 16 we can see that the highest percentage of users are from Asia. In Fig. 17 the situation is a bit different: the percentage of users from outside Asia is higher during this event, indicating that PPLive gathers hundreds of thousands of users from all over the world when important events are broadcasted.

Classifying user connections on the basis of their upload capacity is a further, useful distinction to perform. We can distinguish users into private, when their IP addresses are not visible outside their own LAN, and public, when their IP addresses are visible. If some peers have the same IP address, they are typically users behind a NAT device. By checking the user capacity to establish TCP connections a peer can be further categorized as [4]:

- *Direct-connect*: peer with public address, that can establish partnerships to and from other peers;

- *Universal Plug and Play (UPnP)*: peer with private address, that can establish partnerships with other peers and the other peers can establish partnerships with it;
- *NAT*: peer with private address, that the other peers cannot establish partnership with;
- *Firewall*: peer with public address, that the other peers cannot establish partnership with.

The CoolStreaming analysis performed in [1] provides the peers classification shown in Fig.18. We notice that only a small percentage of the peers are UPnP and direct-connect nodes: they are 30% of the total and contribute with more than 80% to the upload bandwidth [1]. On the other hand, there is a significant percentage of users behind NAT devices, having limited uploading capacities. The remaining small percentage represents peers that usually stay in the system for a short period of time. Most of them are NAT/firewall peers [1]. We anticipate here that a similar situation occurs in the small overlay we have examined.

An additional system feature that is worth mentioning is the session duration time, i.e., the time that elapses between the join and the departure of the peer from the system.

The results reported in [4] for the CoolStreaming system are shown in Fig. 19, where the qualitative distribution of the session duration is shown. We observe that, once the users successfully obtain the video stream, they are stable and remain within the system for the entire program duration [4]. But there are also many short sessions, that depend on the startup failures of newly joined nodes. Literature provides different indications for the statistical description of the session duration time: [16] relies upon a lognormal distribution function; on the contrary, [10] adopts a Weibull distribution to describe the lifetime of the viewers. To the authors' knowledge, there are however no precise, quantitative studies that try to fit experimental data to some specific distributions.

Last parameter we comment upon is the start-up delay. Figure 20 plots its Cumulative Distribution Function (CDF) as reported in [4] for the native, pull-based only CoolStreaming scheme, as well as for the new CoolStreaming, based on the hybrid push-pull architecture. It is immediate to conclude that, although the pull-based system is simple and robust, the hybrid solution definitely exhibits a superior performance.

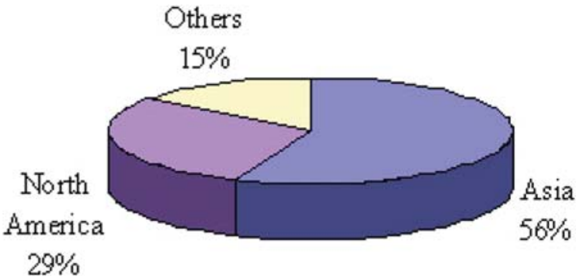


Fig. 17 Users geographical distribution in PPLive during the spring festival gala

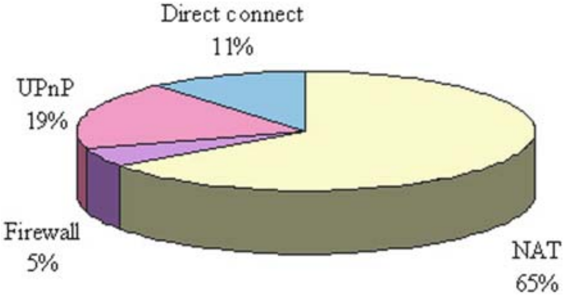


Fig. 18 Peer classification in CoolStreaming

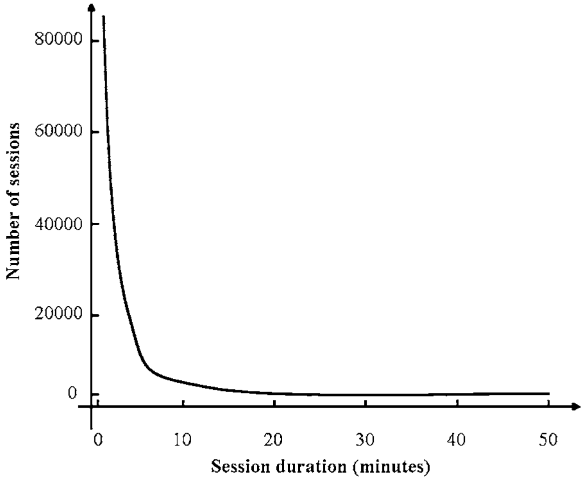


Fig. 19 Distribution of sessions duration in CoolStreaming

Similar conclusions are attained in [30], where both simulations and real traces indicate that the pull-push hybrid approach can significantly reduce play-back delays.

A deeper insight into some of the most significant quality-related metrics is provided in [20], where the performance of a large – undisclosed – live P2P video system is evaluated.

In order to shed some light on how a small system works, we next present and discuss some measurements performed server-side on this kind of architecture. The parameters that have been monitored are:

- the number of users in the weekday and during peak hours;
- the geographical distribution of the users;
- the percentage of free riders upon the total number of peers.

Figure 21 plots the number of StreamerOne users during a soccer match of the 2008 European Championship. The number of concurrent users remains fairly low

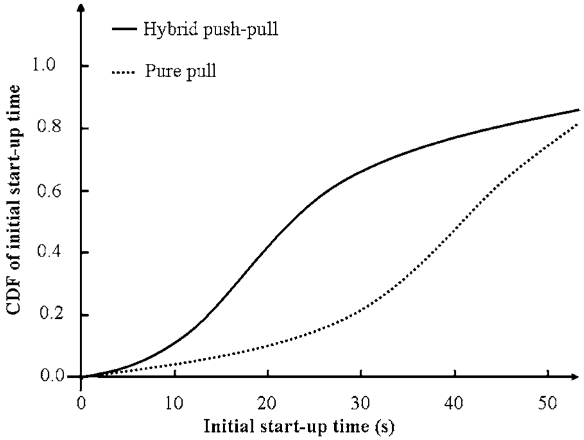


Fig. 20 Start-up delay CDF in CoolStreaming

until about 14:00, it then increases reaching the peak (around 1300 peers) when the match begins. This number progressively decreases at the end of the match. It is immediate to notice that the number of users of this small system is by far lower than in CoolStreaming and PPLive.

Next, Fig. 22 represents the geographical distribution of the users in the small overlay. We observe that most of them are located in Europe, but there is also a non negligible amount of U.S.A. nodes. The remaining percentage represents users from the rest of the world.

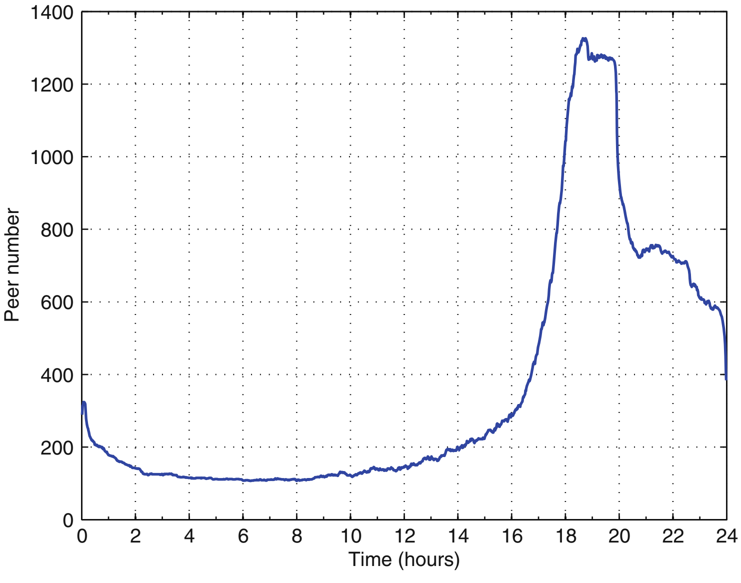


Fig. 21 Evolution of the number of peers during a very popular event in a small system

A last, important feature that we take into consideration is the presence of free riders, without any distinction between nodes lying behind a NAT device or a firewall. These peers do not contribute to the diffusion of the video stream, as they receive chunks from others, without uploading anything. Free riders therefore represent a serious threat to the functionality of any P2P system. As Fig. 23 shows, their number is significant in the small overlay, where their negative impact on performance has to be properly limited. If no countermeasures are taken, the risk is to unbearably degrade the QoE level that "good", collaborative users expect.

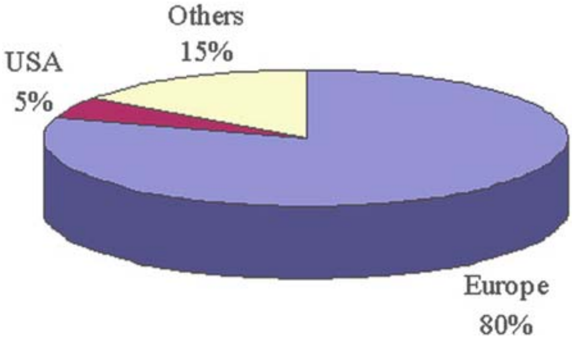


Fig. 22 Geographic distribution of peers in a small P2P system

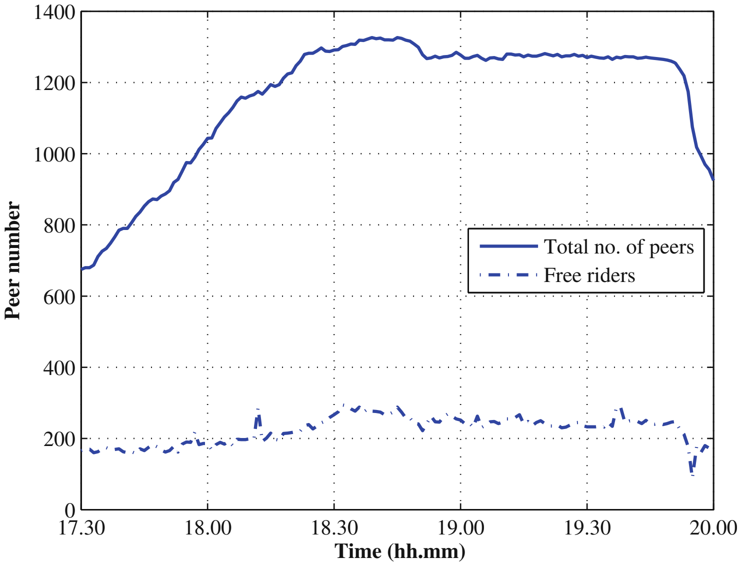


Fig. 23 Total number of users and free riders in a small system

6 Modeling Insights

Now that we have understood how P2P streaming systems work and how their behavior can be monitored, we shift our attention to some of the most significant contributions in the field of modeling that recently appeared in literature. The goal is to delineate the effects on performance of the main system attributes: upload and download capacity of the peers; classes of peers and their cardinality; buffering available at peers and bearable playback delay; free riders and churns that dynamically and unpredictably spoil system equilibrium.

6.1 First Modeling Efforts

To begin with, we develop some simple considerations that help understand the advantages of the P2P solution with respect to the traditional, centralized client-server architecture, no matter whether P2P is employed for file sharing or for video streaming.

Following [3] and more substantially [17], we investigate how the server-side capacity u_s needed to support a rate r to N identical users can be decreased, when each peer makes available a fraction η of its upload bandwidth u to the system.

In this simple, deterministic setting, we observe that

$$u_s = max\left[0, N(r - \eta \cdot u)\right]. \tag{1}$$

The relation between N and u_s is depicted in Fig. 24, for $r = 700$ kbit/s, an upload bandwidth $u = 400$ kbit/s and for different values of η, namely, $\eta = 0.9, 0.5$ and 0. From visual inspection, it is straightforward to conclude that when the P2P system is well designed, a considerable saving is achieved in server capacity : as an example, when $N = 2 \times 10^4$ peers populate the system, u_s can be decreased from 14 Gbit/s to 6.8 Gbit/s for $\eta = 0.9$.

Next, inevitable question is: how can the efficiency η be computed?

To answer, we resort to [17]: the focus of this work is on P2P for file sharing, rather than for video streaming, yet its analysis can be easily adapted to our case as well. In details, this work assumes that the upload bandwidth of a peer will not be utilized only if all of its partners already have the chunks of that peer: it follows that the efficiency η is

$$\eta = P[\text{there is at least one peer that wants a chunk the peer has}]. \tag{2}$$

If K is the number of partner peers and the distribution of chunks between peers is independent from peer to peer, and identical for all peers, then

$$\eta = 1 - P[\text{peer j needs no chunks from peer i}]^K \tag{3}$$

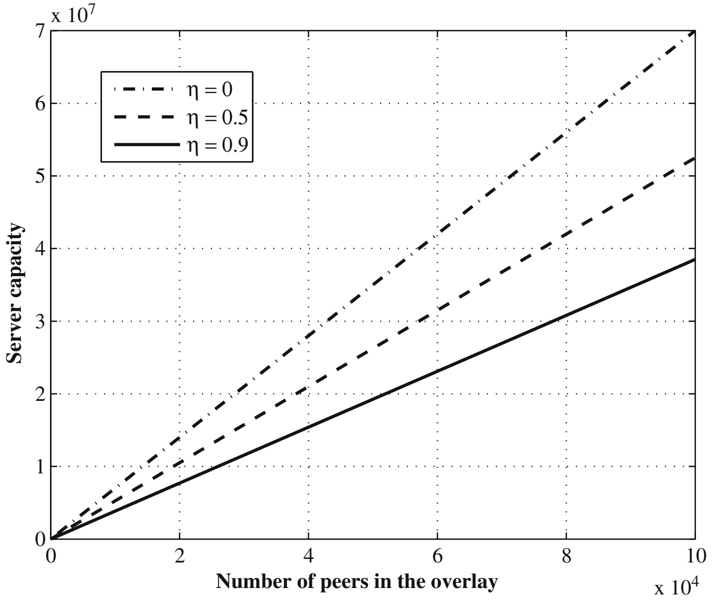


Fig. 24 Server capacity to support N simultaneous viewers in the P2P architecture ($\eta \neq 0$) and in the client server approach ($\eta = 0$)

We next denote by n_i the number of chunks that peer i possesses, out of the M available for sharing in the peer buffer, and assume n_i is uniformly distributed in $[0,\ldots,M-1]$. We can therefore write:

$$P[\text{peer j needs no chunks from peer i}] = P[\text{peer j has all pieces of peer i}] = \\ = \textstyle\sum_{h=1}^{M-1}\frac{1}{M}\cdot\sum_{k=0}^{h}\frac{1}{M}\cdot P[\text{peer j has all chunks of peer i}|n_i = k, n_j = h]\,, \qquad (4)$$

where last expression directly derives from the hypothesis that

$$P[n_i = k] = P[n_j = k] = \frac{1}{M}\,, \quad \forall k \in [0, M-1]\,. \qquad (5)$$

Further elaborating on (4) leads to

$$P[\text{peer j needs no chunks from peer i}] = \\ = \tfrac{1}{M^2}\textstyle\sum_{n_j=1}^{M-1}\sum_{n_i=0}^{n_j}\dfrac{\dbinom{M-n_i}{n_j-n_i}}{\dbinom{M}{n_j}} = \tfrac{1}{M^2}\textstyle\sum_{n_j=1}^{M-1}\dfrac{\dbinom{M+1}{n_j}}{\dbinom{M}{n_j}} = \\ = \tfrac{1}{M^2}\textstyle\sum_{n_j=1}^{M-1}\frac{M+1}{M+1-n_j} = \frac{M+1}{M^2}\sum_{n_j=1}^{M-1}\frac{1}{M+1-n_j} = \frac{M+1}{M^2}\sum_{m=2}^{M}\frac{1}{m} \qquad (6)$$

having set $m = M + 1 - m_j$.

Replacing last outcome in η expression, it can be concluded that

$$\eta = 1 - \left(\frac{M+1}{M^2}\right)^K \cdot \left(\sum_{m=2}^{M} \frac{1}{m}\right)^K . \tag{7}$$

The (K, η) relation, illustrated in Fig. 25 for three different values of the buffer size M, reveals that it is useless to increase the number of partner peers K above $5-10$, as efficiency has already reached unity. It also suggests that the number of available chunks M in the peers' cache plays a significant role when determining η: when M is large, η increases. On the other hand, when M grows, the playback delay that the peer experiences also increases, a phenomenon that needs to be accurately monitored. Interestingly $5-10$ is deemed an adequate partner range for several of the current P2P IP-TV applications.

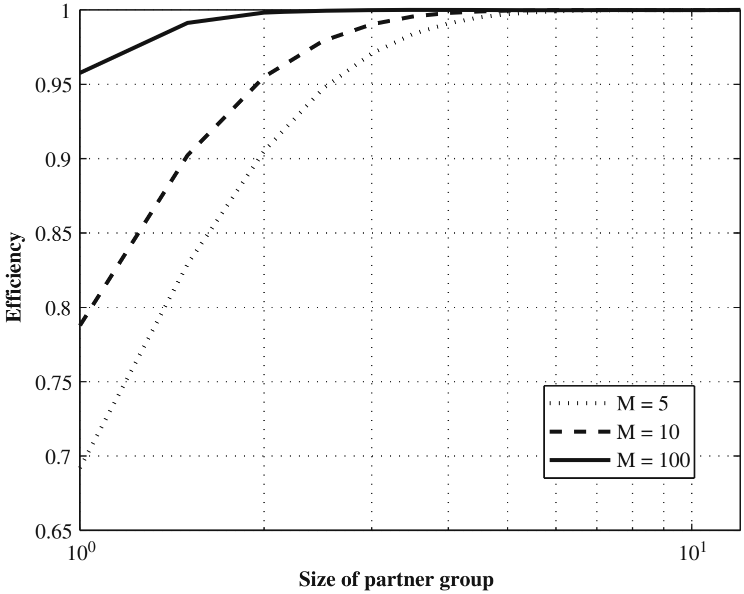


Fig. 25 Efficiency as a function of the parent group size K, for different values of the number M of available chunks in the peers' buffer

6.2 More Recent Contributions

A deeper insight into the fundamental characteristics of P2P streaming systems that rely upon a fully connected mesh is achieved in [27].

To understand its contribution, let us introduce some proper assumptions and notations.

A fluid model is employed to describe the video stream that propagates from the server at rate r, and to describe the bits – as opposed to chunks – that propagate among peers. The system initially examined is bufferless, i.e., bits are not cached in the peer before being played back or before being copied to other peers.

Let N be the number of peers in the system and let u_i denote the upload bandwidth of the i-th peer, $i = 1,2,\ldots,N$. When all peers in the system receive the video at rate r, the system is said to guarantee *universal streaming*.

The first result that is provided states that the maximum achievable streaming rate, r_{max}, is given by

$$r_{max} = min\left\{u_s, \frac{u_s + \sum_{i=1}^{N} u_i}{N}\right\} \tag{8}$$

that implies *all* peers contribute to the swarming process with their *entire* upload bandwidth (equivalently, the efficiency η_i is 1, $\forall\, i$, $i = 1,2,\ldots,N$).

For most real P2P systems, a useful reference is a two class model, where a user can be classified as either a super peer or an ordinary peer: typically, the former has a high-speed access, such as from a campus network, whereas the second utilizes a residential, ADSL access: we assume all peers in either class exhibit the same upload capacity. We denote by N_1 and N_2 the number of super peers and ordinary peers, respectively, and by u_1 and u_2 their upload capacities. A further hypothesis is that $u_2 < r < u_1$.

It immediately follows from (8) specialized to the latter case, that universal streaming can be achieved whenever the following inequality is satisfied:

$$r \leq \phi(N_1,N_2) = min\left\{u_s, \frac{u_s + N_1u_1 + N_2u_2}{N_1 + N_2}\right\} \tag{9}$$

Let us now examine in greater detail the $\phi(N_1,N_2)$ term.

Let peers of both classes join the system following a Poisson process, of rate λ_i, $i = 1,2$, and indicate by μ_i, $i = 1,2$, the rate at which they leave. No hypothesis is made about their sojourn times, that can be arbitrary. Let $N_1(t)$ and $N_2(t)$ indicate the number of peers of the two classes in the system at time t: note that they are two independent Poisson random variables [22].

Focusing on the second term in $\phi(N_1,N_2)$, which is the significant one for all the cases of practical interest, we can compute the probability of achieving universal streaming as

$$P\left[r \leq \frac{u_s + N_1(t)u_1 + N_2(t)u_2}{N_1(t) + N_2(t)}\right] = P\left[N_1(t) \geq cN_2(t) - u_s^{'}\right], \tag{10}$$

where

$$c = \frac{r - u_2}{u_1 - r} \quad \text{and} \quad u_s^{'} = \frac{u_s}{u_1 - r}. \tag{11}$$

Recalling that $N_1(t)$ and $N_2(t)$ are independent Poisson random variables, hence obey the following distribution,

$$f_i(n) = \frac{e^{-\rho_i}\rho_i^n}{n!} \quad \text{with} \quad \rho_i = \frac{\lambda_i}{\mu_i} \quad \text{and} \quad i = 1,2, \tag{12}$$

it is possible to write

$$\begin{aligned} P[\text{universal streaming}] &= \textstyle\sum_{l=0}^{\infty} P[N_1(t) \geq cl - u_s^{'} | N_2(t) = l] \cdot f_2(l) = \\ &= P[N_2(t) \leq \lfloor \tfrac{u_s^{'}}{c} \rfloor] + \textstyle\sum_{l=M+1}^{\infty} P[N_1(t) \geq \lceil cl - u_s^{'} \rceil] \cdot f_2(l) = \\ &= F_2(M) + \textstyle\sum_{l=M+1}^{\infty} \left((1 - F_1(\left\lceil cl - u_s^{'} \right\rceil) + f_1(\left\lceil cl - u_s^{'} \right\rceil) \right) \cdot f_2(l), \end{aligned} \tag{13}$$

where

$$F_i(n) = \sum_{l=0}^{n} f_i(l) \quad \text{and} \quad M = \left\lfloor \frac{u_s^{'}}{c} \right\rfloor . \tag{14}$$

Making use of (13), it is now possible to explore the achievable performance of the P2P system. The work in [27] examines a "small" overlay, with a number of simultaneous peers in the proximity of 75, and a "large" overlay, whose number of simultaneous peers are in the proximity of 7500. The considered rates are $r = 3$ units, $u_1 = 7$ units and $u_2 = 1$ units, to be interpreted, e.g., as 300, 700 and 100 kbit/s, respectively. The average sojourn times are set to $\frac{1}{\mu_1} = \frac{1}{\mu_2} = 30$ minutes for both classes; in the small overlay λ_2 is set to 100 peers per hour, so that ρ_2, the average number of ordinary peers, turns out to be equal to 50; in the large overlay, λ_2 is set to 10000 peers per hour, leading to $\rho_2 = 5000$. In both scenarios λ_1 is then varied: in the proximity of 25 in the small system, near 2500 in the large one.

Having defined the probability of degraded service as

$$P_{degr} = 1 - P[\text{universal streaming}], \tag{15}$$

Figure 26 qualitatively reports P_{degr} as a function of the ratio $\frac{\rho_1}{\rho_2}$, determined in [27] for both a small and a large overlay, when $u_s = 7$: this figure shows that in both systems performance improves when the arrival rate of super peers increases (equivalently, when the $\frac{\rho_1}{\rho_2}$ ratio increases). It can further be observed that the P2P system with a large population performs better, as the large overlay provides universal streaming over a much wider range of system parameters: the physical justification behind this behavior is that the departure of super peers has a confined effect in the large overlay, whereas it can lead to a significant worsening in the small system.

The study also points out that when the system scales towards larger dimensions, the role of the server and its upload rate u_s become more and more marginal.

Once the assumptions of bufferless peers is removed, the analytical, closed-form approach has to be abandoned in favor of numerical solutions obtained via a simulation tool. The most salient outcome derived by the authors of [27] is that the buffer introduction brings a remarkable improvement to system performance. For instance, they observe that the large overlay previously examined, that achieved a

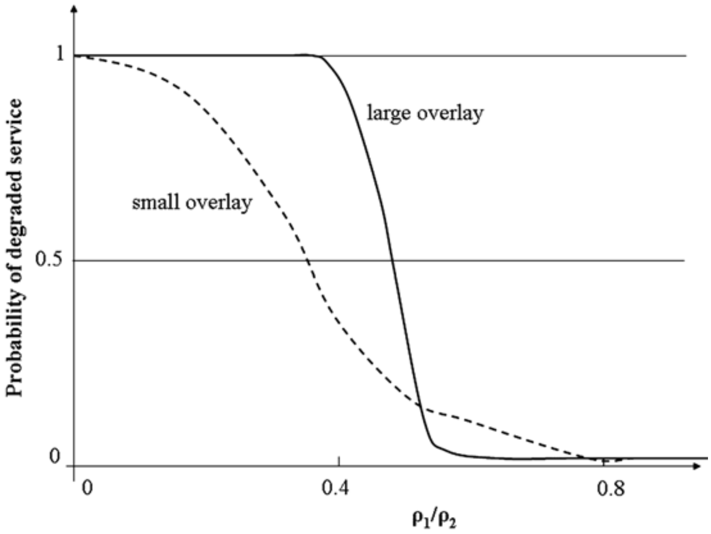


Fig. 26 Probability of failing to achieve universal streaming in a small and in a large overlay, as a function of $\frac{\rho_1}{\rho_2}$, when $\lambda_2 = 100$

probability of degraded service equal to 0.5 at $\frac{\rho_1}{\rho_2} = 0.5$, experiences a P_{degr} of approximately 0.05 when buffering is introduced. Moreover, a cache that can store 30 seconds of the video is already sufficient to exploit almost all the potential buffering gain. Analogous results hold for the small overlay.

Finally, it is observed that buffering can even bring a larger improvement than increasing the infrastructure server bandwidth u_s.

What is the limit of the analysis just described? Mainly the assumption that all N peers are connected to each other and that all $N-1$ peers contribute with their available content to the stream the remaining peer is viewing: definitely not what happens in a real system, as we previously described. Being able of utilizing all the peer upload bandwidth it is not only a matter of random fluctuations in the peer numbers, $N_1(t)$ and $N_2(t)$.

To mention only a few additional key elements, we observe that a successful system behavior heavily depends on the scheduling algorithm employed by peers, on the parents they rely upon, on the type of connections that peers experiment (they can be fully visible or belong to the free rider class, meaning that they can download content but do not contribute to the system with any upload bandwidth).

Yet, the main goal of [27] is to determine the *maximum* achievable streaming rate r_{max} and to tie the evaluation of the probability of degraded service to it: as such, it represents a genuine effort.

There are a few, additional papers that deal with the performance evaluation of P2P streaming systems. Although we do not have room to cite here all their outcomes, yet we believe that they deserve a careful citation: [23] for the theoretical effort in finding heuristics with provable good performance; [30] for the mathematical analysis that helps comprehend why the pull-based approach is so efficient in utilizing peer upload capacity; [8] for a model relying on stochastic graph theory, able to capture the fundamental properties of the mesh-based systems. The interested reader is strongly encouraged to enrich his/her knowledge through these excellent references.

6.3 A Numerical Comparison Between Mesh and Multiple-Trees

If we abandon the analytical world and consider the simulative approach, an interesting comparison between mesh-based and multiple tree-shaped overlays is performed in [16].

In this work, the hypothesis is that both systems employ Multiple Description Coding (MDC). For this technique, a video stream is divided into multiple substreams: each can be independently decoded and the quality of the received video increases with the number of decoded substreams. Every peer can subscribe to a different number of substreams, depending on its download bandwidth. The MDC choice helps in counteracting the heterogeneity in peers' bandwidth, and therefore warrants a better bandwidth usage.

In the multiple tree approach, different substreams propagate via disjoint trees; each peer can join different trees, subject to the constraint that it has to be an internal node in only one tree, and has to appear as a leaf in the remaining trees of the overlay. Moreover, the trees have to be balanced and short, meaning that the number of their internal nodes as well as their depth have to be comparable.

Proper rules are set to handle node arrivals and departures [16]: as an example, a new peer is added as an internal node to the tree with the lowest number of internal nodes; a new internal node is placed as a child for the node with the lowest depth that can accept a new child or has a leaf child; when an internal node departs, the subtree that was rooted in it tries to rejoin the tree as a whole, but if it does not succeed, its constituent nodes independently rejoin the tree.

In the examined mesh-based approach, the content delivery mechanism is very similar to the one BitTorrent adopts. A new peer contacts the bootstrapping node, that replies providing a random subset of peers able to act as parents for the node; connected peers have a parent-child relationship, like in the tree-based architecture. Peers periodically indicate what video chunks they have available to their child peers and in turn request – pull – chunks from their parents.

The packet scheduling aims at profitably utilizing the upload bandwidth that peers make available to the mesh-based system; it also strives to guarantee an adequate quality of the received video allowing for the reception of several substreams; it finally guarantees a timely delivery of the requested video chunks.

The crucial difference between the two approaches, correctly outlined by the authors of this paper, lies in the delivery process of the substreams. Whereas in the multiple-tree overlay all chunks belonging to one substream *statically* follow the same tree to propagate among peers, in the mesh overlay the delivery tree is individually chosen for each chunk and is *dynamically* shaped as the chunk crosses the network. This greatly helps at efficiently employing all the available bandwidth of the peers. In contrast, in the multiple tree overlay there might be occurrences where one of the internal peers does not have sufficient upload bandwidth "to feed" all its child peers.

In terms of similarity between the two architectures, the first perspective that [16] takes is to state that the superposition of multiple trees is essentially equivalent to a mesh. Another similarity is that in both architectures video chunks follow a tree to reach their destination: it is manifest in the multiple-tree overlay, but even in the mesh-based case it is possible to capture the snapshots of the per-chunk trees, i.e., of the delivery paths that each single video chunk follows [26]: these are indeed trees, whose internal nodes are quite stable and whose depth is typically longer compared to tree-based systems, mainly because of their dynamic formation. Additionally, the per-chunk trees exhibit a high degree of correlation, sharing several internal links.

The content delivery mechanism of both push and pull-based architectures is numerically examined in [16] by simulation: initially, 200 homogeneous peers with symmetric upload and download bandwidths are considered, in a system that employs 20 substreams, each coded at a rate $bw_d = 80$ kbit/s. The *access link* bandwidth of each peer is set equal to $K \times deg \times bw_d$, where K, $K \geq 0$, is a tunable parameter and *deg* indicates the degree of each peer, i.e., the number of incoming and outgoing connections the peer can support. When *deg* is the same for all peers, $K \times bw_d$ is the average *per-connection* bandwidth. The physical topology underlying the examined systems, as well as the adopted congestion control protocol, is detailed in [16].

The study determines the percentage of bandwidth utilization over the entire system and the average quality delivered to each peer: the first metric quantifies the effects of the content bottleneck phenomenon, i.e., of the situation where a parent peer does not possess any useful chunk to deliver to a particular child, even though it has some upload bandwidth available, as well as the effects of a low link access bandwidth at the peers; the second performance index is defined as the average number of substreams a peer receives during a session and is strictly related to the quality of the received video.

The system bandwidth utilization is determined as a function of the average per connection bandwidth, $K \times bw_d$: in the mesh-based system its behavior is practically independent of the peers' bandwidth, and it takes on high values, of the order of 0.95, revealing that the mesh always allows to fully leverage the upload bandwidth that peers make available to the system. In the multiple tree-based architecture, when the average per-connection bandwidth is low, system utilization is poor: this is caused by parent peers that cannot satisfyingly support all their downstream connections in the trees; when the per-connection bandwidth is high, system utilization drops again, although for a different reason: the content bottleneck phenomenon appears. In between these two extremes, satisfying values of bandwidth utilization are achieved, that are however slightly lower – of the order of 0.9 – with respect to the alternative system: they are obtained in the proximity of $K = 1$, i.e., when the per-connection bandwidth is slightly higher than the description bandwidth bw_d. Figure 27 illustrates this behavior in a qualitative manner.

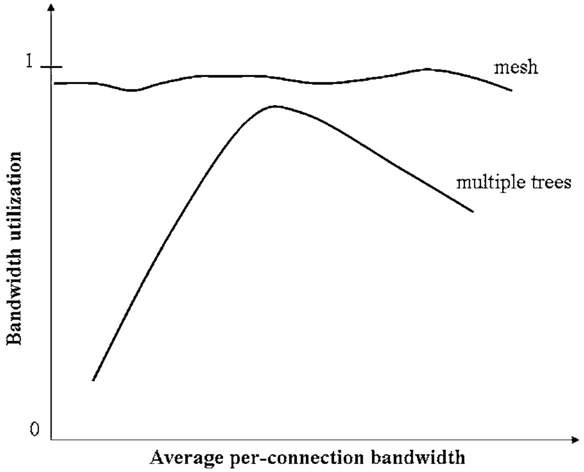


Fig. 27 The qualitative dependence of system bandwidth utilization on the average per-connection bandwidth

Next, the behavior of the average received quality is investigated. As Fig. 28 indicates, quality always increases with the per-connection bandwidth in the mesh system, in a practically linear manner; in the multiple tree approach, increasing the per-connection bandwidth causes the average quality to reach the target value of *deg*, but this limit is not trespassed.

It can be concluded that the mesh-system can fruitfully exploit any value of peer bandwidth, delivering a proportionally higher quality.

When the number of peers that a chunk visits before reaching the destination peer is computed, the simulations reveal that the average path length is longer in the mesh-based approach, as expected.

When the effect of bandwidth heterogeneity among peers is taken into account, both architectures display better utilization and quality.

In contrast, when the effects of the overlay size are examined, the study indicates that the mesh successfully scales, whereas the bandwidth utilization and the quality of the multiple tree approach gradually worsen. This can be explained by the fact that with no changes in the degree, the depth of the trees increases.

As for the ability to cope with churns, this comparative study assumes that the duration of the peer session is lognormally distributed, and that the peer interarrival times obey a Pareto distribution, with properly set parameters [16].

It is observed that the path from source to individual peers is more stable in the mesh, and that for both approaches the ancestor changing rate increases with peer

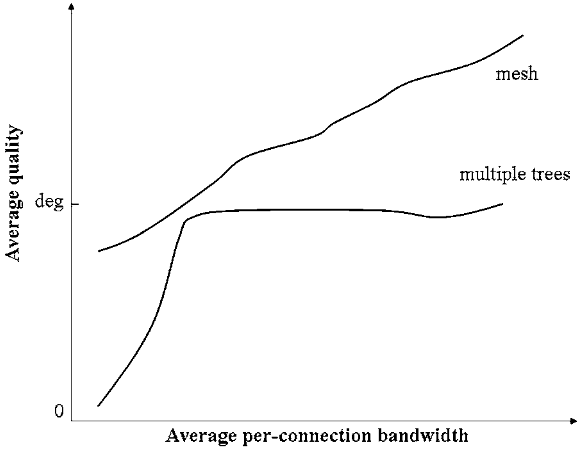


Fig. 28 Average received quality as a function of the per-connection bandwidth

population, due to the fact that the average distance between peers increases with peer population.

In conclusion, this numerical investigation provides useful insights when the main focus is on bandwidth utilization and quality; with respect to these performance metrics, its outcomes indicate that the mesh-based architecture consistently achieves superior performance. However, the study only partially investigates the delays introduced by the mesh architecture, a critical issue, as also [12, 30] have evidenced.

7 Open Issues and Promising Solutions

A wide, large-scale deployment of P2P architectures to distribute live video streaming over the Internet still requires sound answers to several challenging questions: we provide a brief “to-do list” in what follows.

7.1 QoS and QoE

On the QoS/QoE side, the scarce control that P2P systems have on the quality the viewers endure is undoubtedly an issue, mainly in highly dynamic environments. This limit is intrinsic to the structure of the current Internet; nevertheless, it needs to be adequately tackled in the light of commercial, pervasive adoption of P2P-based solutions.

Directly related to the QoE issue, there lies the necessity to decrease the start-up delays, that are often in the order of several tens of seconds: definitely a new, undesired experience for viewers of ordinary TV channels, that on the contrary often and rapidly change channel. As [2] indicates, possible solutions to investigate are network coding and redundant downloading. Another suggested approach requires the employment of dedicated servers, that provide the video at relatively low quality, whereas the P2P overlay guarantees the high quality video [13]. When a peer joins the system, it first – and quickly – receives the low quality stream from the server, then begins to effectively employ the P2P network to obtain the video at a higher quality.

Also, smaller playback lags are needed, to avoid the weird situation where the frames that some peers view are minutes behind other peers. This calls for efficient chunk scheduling techniques and more intelligent peering strategies.

On the measurement side, the remote monitoring of P2P systems is a further, significant chapter: [11] represents a good example of how to track playback continuity, start-up latency and playback lags in a network-wide manner relying upon the harvesting of buffer maps.

However, this is "only" passive monitoring: active countermeasures are still to be devised, to assure and protect a satisfying viewing experience; they should come into play as soon as the monitoring results indicate the need to intervene.

7.2 Network Awareness

On the network side, a crucial point is represented by the impact that P2P traffic has on ISP infrastructures: simple peer selection and random scheduling usually place a significant burden on the underlying network, and peers are highly spread, as the geolocalization analysis shows. Network-aware resource allocation within the overlay should substantially help in relieving the strain.

Finally, users lying behind NAT and firewalls may not be able to contribute to the system with their upload bandwidth: better NAT traversal schemes have to be put into use; how to treat such users, what QoE they should receive is also an interesting dilemma. The work in [6] suggests some simple schemes to improve the overlay construction when peers lie behind a NAT: we report them in Fig. 29. As shown in Fig. 29a, one of the peers could take on the responsibility of broadcasting to all other users behind the NAT; alternatively, the incoming P2P traffic could be confined, yet some redundancy be guaranteed, as Fig. 29b summarizes.

We dot not have room to explore in detail all these interesting questions; rather, we have intentionally decided to concentrate on a specific, promising instance of enhancement, that is, network coding.

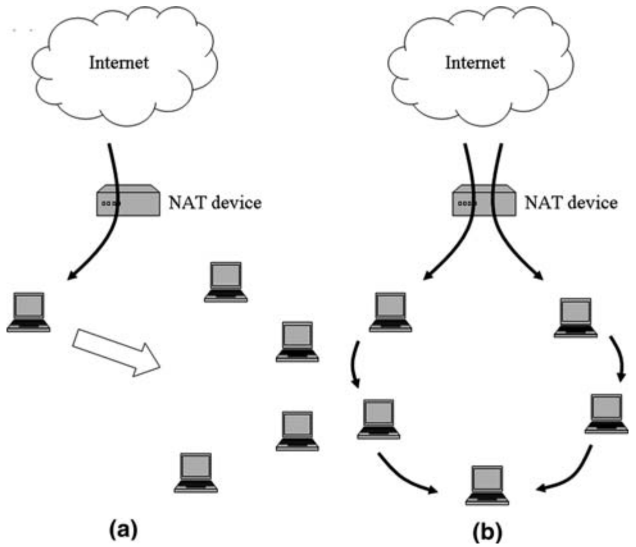


Fig. 29 (**a**) A possible optimization scheme for NAT users; (**b**) An alternative solution

7.3 Network Coding for P2P Streaming

To scout this novel research topic, we refer to [15, 24], notwithstanding that there are several alternative approaches that are equally worth being investigated.

The first step to understand the potential of network coding for P2P video streaming was performed in [15], where a conventional pull-based P2P overlay was directly compared with an analogous scheme enhanced by the adoption of random network coding . In the latter system, each video chunk **b** is further divided into n blocks $\mathbf{b} = [b_1, b_2, \ldots, b_n]$, each block b_i having a fixed size k. When a video chunk has to be transmitted to a peer, the parent, say p, rather than sending the entire chunk, chooses m blocks out of the n within the chunk, $m \leq n$, and a set of random coefficients $c_1, c_2, \ldots, c_m$ in the Galois field $GF(2^8)$, to produce one coded block x of size k:

$$x = \sum_{i=1}^{m} c_i^p \cdot b_i^p \, . \tag{16}$$

As each coded block is a linear combination of m original blocks, it is uniquely identified by the coefficients of the combination. Then, the parent peer p transmits the coded block, together with the coefficients of the linear combination (a overhead not to be underestimated).

As the session proceeds, a peer accumulates coded blocks from its parents and in turn encodes blocks to forward to its child peers. The destination peer d can recover the original video chunk as soon as it has received n linear independent coded blocks $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, taking advantage of the following relation:

$$\mathbf{b} = \mathbf{A}^{-1} \cdot \mathbf{x}^T , \tag{17}$$

where **A** is the matrix of the coding coefficients of x (each row in **A** corresponds to the coefficients of one coded block).

The decoding process becomes faster resorting to Gauss-Jordan elimination: now the peer starts to decode a video chunk as soon as it receives its first coded block. Moreover, if the peer receives a coded block that is linearly dependent on previously received blocks, the elimination process provides an all zeros row; in turn, this event triggers the discarding of the coded block, while the receiver keeps waiting for additional data. In other words, there is no need of an a priori check on the received blocks, to guarantee they are linear independent.

What the authors of the study in [15] observe is that the overlay with random network coding is more resilient against network dynamics and achieves a better bandwidth usage. This has to be ascribed to the finer granularity introduced by the coding mechanism in the streaming process: the system unit is the coded block, rather than the entire video chunk.

Also note that in this overlay a peer missing a video chunk may be served by multiple randomly selected peers that have coded blocks of the same requested chunk.

Last remark becomes crucial for the design of the novel P2P streaming system proposed in [24], and termed R^2.

Here too, random network coding is confined within the single chunk, for the primary reason of reducing the number of blocks that the encoder has to manipulate, therefore limiting its complexity.

However, to enhance the probability of having different multiple sources for the same video chunk, each contributing with its own coded block, this new system employs a random push, rather than the explicit pull requests that are typical of mesh-based systems, to drive the diffusion process.

In greater detail, it is the destination peer, say d, that – very frequently – advertises what its missing chunks are (rather than what its availability is), via its buffer maps. Now any peer p that receives this information and possesses any of these chunks can potentially act as one of the origins for that chunk. Indeed, the algorithm states that, if p possesses the chunk, p randomly decides whether to push out one coded block of the chunk that d is missing.

A video chunk is therefore truly served by multiple originating peers, and this choice allows to take full advantage of the benefits random network coding brings in. While the session proceeds, the receiving peer accumulates coded blocks from different contributing peers into its local buffer; as in [15], it immediately starts the progressive decoding process resorting to Gauss-Jordan elimination.

The reader is referred to [24] for the thorough description of R^2 and of all its characteristics and constituent algorithms.

The idea behind the R^2 proposal is depicted in Fig. 30, that helps to understand how it departs from a traditional pull-based overlay, whose concept is shown in Fig. 31.

As intuition also suggests, the newly proposed architecture allows the adoption of much larger video chunks than pull-based overlays. This can be qualitatively

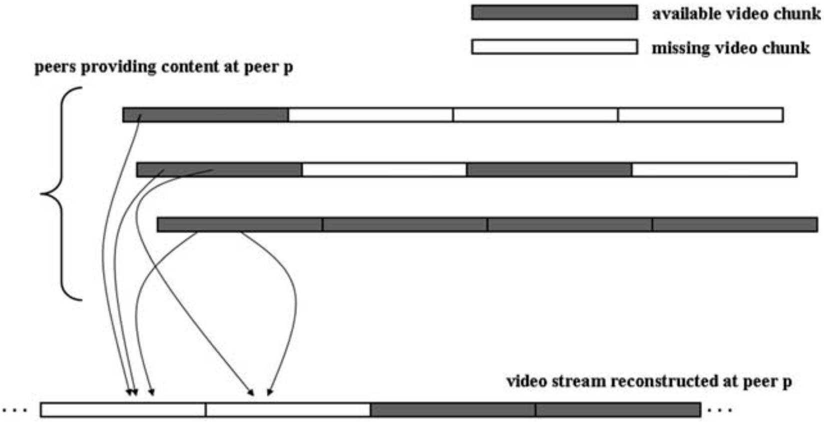

available video chunk
missing video chunk
peers providing content at peer p
video stream reconstructed at peer p

Fig. 30 The concept behind R^2 and its adoption of network coding

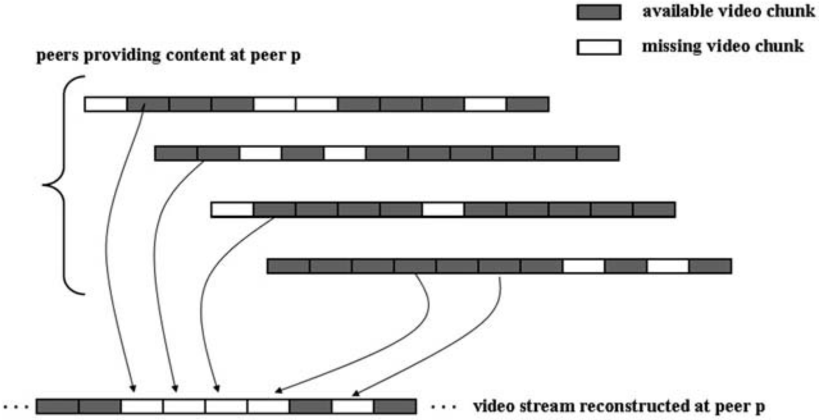

available video chunk
missing video chunk
peers providing content at peer p
video stream reconstructed at peer p

Fig. 31 The concept behind a traditional pull-based mesh system

explained observing that in a conventional system a missing video chunk is served by one partner peer at a time. In contrast, a missing video chunk in R^2 is typically sent by multiple originating peers, and each of them selects via a proper algorithm (i) which chunk to send; (ii) what coding coefficients to use. Equivalently, peers collaborate, randomly and without any explicit knowledge of each other, to the reconstruction of each chunk.

Several additional assumptions are introduced in [24] and contribute to the satisfying performance of the system: to cite a few, playback is synchronized, so as to maximize the overlap of playback buffers in the peers; peers receive buffer maps in an extremely timely manner; when composing coded blocks, peers attribute higher priority to video chunks close to the – common – playback deadline. Under such hypotheses, that are quite cumbersome indeed, the study comes to the conclusion that R^2, when properly tuned, outperforms conventional P2P solutions: it significantly reduces the number of playback skips in a streaming session, therefore achieving better playback quality; it quickly fills buffers at the peers at start-up time and it maintains them adequately filled during the entire session, despite of peers' departures and arrivals.

On the negative side, the new system has to be carefully tuned in terms of chunk and block size; it does imply complex choices, such as the synchronization of peers; it mandates buffer maps to be almost continuously exchanged and control messages to constitute a non negligible fraction of the entire traffic. Nevertheless, it truly represents the first step toward a deeper understanding of the potential of random network coding in P2P overlays.

8 Summary

The aim of this Chapter is to provide the reader an up-to-date, fresh picture on P2P technology as applied to live streaming systems.

After introducing the motivations behind the P2P approach for video delivery, a taxonomy of P2P video broadcasting architectures is presented.

A representative round-up of some popular streaming systems comes next, followed by a more detailed characterization of the software architecture of mesh-based solutions.

Measurements performed both at network edge and server side offer a tangible description of some large and small overlay features.

The modeling section outlines a systematic view of the analytical and simulative efforts in capturing the essential features and limit behavior of P2P streaming solutions, as currently witnessed by scientific literature.

A look at some challenging open questions that, when solved, will open the way to a massive introduction of the P2P technology in streaming systems, completes the different views gathered by the chapter.

References

1. Xie, S., Keung, G.Y., Li, B.: A measurement of a large-scale peer-to-peer live video streaming system. Packet Video 2007, November 2007, pp. 153–162.
2. Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K.W.: A measurement study of a large-scale P2P IPTV system. IEEE Transactions on Multimedia, Vol. 9, Issue 8, December 2007, pp. 1672–1687.
3. Tewari, S., Kleinrock, L.: Analytical model for BitTorrent-based live video streaming. Proc. of IEEE CCNC 2007, Consumer Communications and Networking Conference, January 2007, pp. 976–980.
4. Li, B., Xie, S., Keung, G.Y., Liu, J., Stoica, I., Zhang, H., Zhang, X.: An empirical study of the Coolstreaming+ system. IEEE Journal on Selected Areas in Communications, Vol. 25, Issue 9, December 2007, pp. 1627–1639.
5. Babelgum, "Babelgum Homepage" http://www.babelgum.com/

6. Zhang, X., Liu, J., Li, B., Yum, T.S.P.: Coolstreaming/DONet: a data-driven overlay network for efficient live media streaming. Proc. of IEEE Infocom 2005, 13–17 March 2005, Vol.3, pp. 2102–2111.
7. Floyd, S., Handley, M., Padhye, J., Wiedmer, J.: Equation based congestion control for unicast applications. Proc. of ACM/SIGCOMM 2000, May 2000, Vol. 1, pp. 1–14.
8. Carra, D., Lo Cigno, R., Biersack, E. W.: Graph Based Analysis of Mesh Overlay Streaming Systems. IEEE Journal on Selected Areas in Communications, Vol. 25, Issue 9, December 2007, pp. 1667–1677.
9. GridMedia, "GridMedia Homepage," http://www.gridmedia.com/
10. Zhang, M., Sun, L., Yang, S.: iGridMedia: providing delay-guaranteed peer-to-peer live streaming service on Internet, Proc. of IEEE Globecom 2008, to appear.
11. Hei, X., Liu, Y., Ross, K.W.: Inferring network-wide quality in P2P live streaming systems, IEEE Journal on Selected Areas in Communications, Vol. 25, Issue 9, December 2007, pp. 1640–1654.
12. Li, B., Xie, S., Qu, Y., Keung, G.Y.; Lin, C., Liu, J., Zhang, X.: Inside the New Coolstreaming: principles, measurements and performance implications. Proc. of IEEE Infocom 2008, 13-18 April 2008, pp.1031–1039.
13. Hei, X., Liu, Y., Ross, K.W.: IPTV over P2P streaming networks: the mesh-pull approach, IEEE Communications Magazine, Vol. 46, Issue 2, February 2008, pp.86–92.
14. Joost, "Joost Homepage," http://www.joost.com/
15. Wang, M., Li, B.: Lava: a reality check of network coding in peer-to-peer live streaming, Proc. of IEEE Infocom 2007, May 2007, pp. 1082–1090.
16. Magharei, N., Rejaie, R., Guo, Y.: Mesh or multiple-tree: a comparative study of live p2p streaming approaches. Proc. of IEEE Infocom 2007, May 2007, pp. 1424–1432.
17. Qiu, D., Srikant, R.: Modeling and performance analysis of BitTorrent-like peer-to-peer networks. Proc. of ACM SigComm, pp. 367–378.
18. Liu, J., Rao, S.G., Li, B., Zhang, H.: Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast. Proceedings of the IEEE, Vol.96, Issue 1, January 2008, pp. 11–24.
19. Li, B., Yin, Y.: Peer-to-peer live video streaming on the internet: issues, existing approaches, and challenges. IEEE Communications Magazine, Vol. 45, Issue 6, June 2007, pp. 94–99.
20. Agarwal, S., Singh, J.P., Mavlankar, A., Baccichet, P., Girod, B.: Performance and quality-of-service analysis of a live P2P video multicast session on the Internet, Proc. of the International Workshop on Quality of Service, Enschede, The Netherlands, 2008.
21. PPLive, "PPLive Homepage," http://www.pplive.com/
22. Kleinrock, L.: Queueing systems. Volume I: Theory. John Wiley & Sons, New York, 1975.
23. Massoulie, L., Twigg, A., Gkanttsidis, C., Rodriguez, P.: Randomized decentralized broadcasting algorithms, Proc. of IEEE Infocom 2007, May 2007, pp. 1073–1081.
24. Wang, M., Li, B.: R^2: random push with random network coding in live peer-to-peer streaming, IEEE Journal on Selected Areas in Communications, Vol. 25, Issue 9, December 2007, pp. 1655–1666.
25. SopCast, "SopCast Homepage," http://www.sopcast.com/
26. Wang, F., Liu, J., Xiong, Y.: Stable peers: existence, importance, and application in peer-to-peer live video streaming. Proc. of IEEE Infocom 2008, 13–18 April 2008, pp. 1364–1372.
27. Kumar, R., Liu, Y., Ross, K.: Stochastic fluid theory for P2P streaming systems. Proc. of IEEE Infocom 2007, May 2007, pp. 919–927.
28. StreamerOne, "StreamerOne Homepage," http://www.streamerone.it/
29. Handley, M., Floyd, S., Padhye, J., Wiedmer, J.: TCP Friendly Rate Control (TFRC): protocol specification, RFC 3448, January 2003, ftp://ftp.rfc-editor.org/in-notes/rfc3448.txt, proposed standard.
30. Zhang, M., Zhang, Q., Sun, L., Yang, S.: Understanding the power of pull-based streaming protocol: can we do better? IEEE Journal on Selected Areas in Communications, Vol. 25, Issue 9, December 2007, pp. 1678–1694.
31. UUSee, "UUSee Homepage," http://www.uusee.com/
32. Sentinelli, A., Marfia, G., Gerla, M., Kleinrock, L., Tewari, S.: Will IPTV ride the peer-to-peer stream? IEEE Communications Magazine, Vol.45, Issue 6, June 2007, pp. 86–92.

Providing VoD Streaming Using P2P Networks

Juan Pedro Muñoz-Gea, Josemaria Malgosa-Sanahuja, Pilar Manzanares-Lopez, and Juan Carlos Sanchez-Aarnoutse

Abstract Overlays and P2P systems, initially developed to support IP multicast and file-sharing, have moved beyond that functionality. They are also proving to be key technologies for the delivery of video streaming. Recently, there have been a number of successful deployments for “live” P2P streaming. However, the question remains open whether similar P2P technologies can be used to provide VoD (Video-On-Demand) services. A P2P VoD service is more challenging to design than a P2P live streaming system because the system should allow users arriving at arbitrary times to watch (arbitrary parts of) the video.

In this work, the requirements to supply P2P VoD services and the different design decisions that have to be adopted are surveyed. The open problems are also presented.

Juan Pedro Muñoz-Gea
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain,
e-mail: `juanp.gea.sanchez@upct.es`

Josemaria Malgosa-Sanahuja
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain, e-mail: `josem.malgosa@upct.es`

Pilar Manzanares-Lopez
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain,
e-mail: `pilar.manzanares@upct.es`

Juan Carlos Sanchez-Aarnoutse
Department of Information Technologies and Communications, Polytechnic University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Spain, e-mail: `juanc.sanchez@upct.es`

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_36, © Springer Science+Business Media, LLC 2010

1 Introduction

Overlays and P2P (peer-to-peer) systems, initially developed to support IP multicast and file-sharing, have moved beyond that functionality. With the increasing bandwidth capacity provided by the Internet, they are also proving to be key technologies for the delivery of video streaming.

The basic solution for streaming video over the Internet is the client-server model: a client connects with a server and the video is streamed to the client from the server. One variation of the client-server model is the Content Delivery Network (CDN) model: the server first pushes the video to a set of content delivery servers, and the client, in order to download the video, is directed to a nearby content delivery server. The advantages of the CDN model respect to the client-server model are the following: it shortens the users' startup delays, reduces the traffic on the network and serves more users. However, the major challenge for server based video streaming solutions is its scalability, because the bandwidth provision must grow proportionally with the client population. This makes the server based video streaming solutions expensive [14].

P2P networking has emerged as a new paradigm to build distributed network applications. The basic design philosophy of P2P is to encourage users to act as both clients and servers, namely as peers. In a P2P network, a peer not only downloads data from the network, but also uploads the downloaded data to other users in the network. Recently, there have been a number of successful deployments for "live" P2P streaming. Coolstreaming [24], which was built in March 2004, has been recognized as one of the earliest large-scale P2P live streaming systems. It was constructed using a new concept called *data-driven* model, which is somewhat similar to the technique used in BitTorrent. Since then, there have been several commercial systems that have attracted a large number of viewers, such as PPLive [16] and SopCast [18].

On the other hand, Video-on-Demand (VoD) has been identified as the key feature to attract consumers to IPTV (Internet Protocol Television) service. VoD service allows users to watch any point of video at any time. Compared with live streaming, VoD offers more flexibility and convenience to users and truly reaches the goal of *watch whatever you want whenever you want*. The question has remained open whether P2P technology can be used to provide VoD services. It is more challenging to design a P2P VoD service than a P2P live streaming system because even though a large number of users may be watching the same video, they are asynchronous to one another (different users may watch different portions of the same video at any given moment).

A fundamental component of any P2P VoD system is its overlay network structure, related to the organization of participating peers into an overlay structure. Based on this component, P2P VoD systems can be broadly classified into *tree-based* and *mesh-based* categories. Tree-based systems have well organized overlay structures and they typically distribute video by actively pushing data from a peer to its children peers. One major drawback of tree-based streaming systems is their vulnerability to peer departure, which will temporarily disrupt video delivery to all peers in the subtree rooted at the departed peer. On the other hand, in a mesh-based P2P streaming system, peers are not confined to a static topology. Instead, peer relationships are established/terminated based on the content and bandwidth availability on peers. This kind of system tries to achieve fast file downloading by swarming. With this method, a file is divided into small size segments, and the server disperses them to different users. The users download from their neighbors the segments that they currently do not have. Since multiple neighbors are maintained at any given moment, mesh-based video streaming systems are highly robust to peer failures. Therefore, this chapter is going to concentrate on mesh-based approaches.

An important component of any mesh-based P2P VoD system is the content forwarding policy, that is, the way the multimedia content is stored and transmitted to each participating peer through the overlay. There are two main categories for this component: the *buffer-forwarding* and the *storage-forwarding* approaches. In the buffer-forwarding approach each client caches a limited number of segments around its current “play offset”. In the storage-forwarding approach each peer stores a great number of segments at its local storage (such as hard disk).

Another key aspect in mesh-based P2P VoD systems is the design of mechanisms to find a partner with necessary segments. Due to scalability concerns these mechanisms usually use distributed searching structures, like the distributed hash table (DHT) and the skip-list. In the buffer-forwarding approaches, peers are sorted in the distributed searching structures by the play offset. On the other hand, in the storage-forwarding approaches, peers are sorted in the distributed searching structures by the segments they store. In both cases, when a client wants to play a segment, it first looks for the supplying peers of that segment and then sends requests to those peers for the service.

The collaboration with a partner for content delivery (*data scheduling*) is another very important and challenging problem in mesh-based P2P VoD systems. The scheduling scheme in mesh-based P2P VoD systems has two main components: how to send requests to adequate neighbors, and how to evaluate the requests received from many peers. For the buffer-forwarding approaches, many heuristics have been proposed to address this issue, such as round robin or smallest-delay. Some recently proposed schemes use network coding to improve the throughput. On the other hand, for the storage-forwarding approaches several modifications of the BitTorrent segment selection algorithm have been proposed. In addition, the use of rateless coding, such as LT or Raptor codes, is a new alternative to address this issue.

In this work the requirements to supply P2P VoD services and the different design decisions that have to be adopted are surveyed. Several specific solutions are classified and analyzed in order to give an overview of the state of the art until now. The open problems are also presented.

2 P2P VoD Services Overview

In a general VoD streaming service, videos have to be delivered to users with low delay and VCR-like operation support (e.g. pause, fast-forward, fast-rewind). VCR operations change the location of the viewing point, the video playback direction and the video playback speed, and they introduce extra complexity and overhead into the VoD system. In a P2P VoD service these operations raise a challenging issue, because the neighboring peers may not have the data at the new playback point, or sufficient uplink capacity to support faster playback rate. In addition, peers have heterogeneous capacities in terms of network bandwidth, memory size, etc., which make it difficult to determine where and when to fetch the expected data segments from multiple suppliers within playback deadlines. On the other hand, unlike a traditional VoD service (with dedicated server or CDN), in a P2P VoD service peers dynamically join and leave the system (this phenomenon is denoted as *peer churn*). Peer churn introduces dynamics and uncertainty into the P2P network and it can degrade the user's viewing quality.

Two other very important characteristics of general VoD streaming services, which also have to be taken into account for P2P VoD services, are the following: First, the whole video must be delivered to a new peer that joins the VoD service. Therefore, VoD systems must find an efficient way to provide the latecomers with the initial missing part of the video. Second, a client will stop watching a VoD stream when its QoS degrades. This last observation reveals the importance of a robust failure recovery protocol in a VoD streaming system.

Based on the above discussion, the following dominant problems for P2P VoD streaming are considered:

- *Effective handling of clients asynchronous requests*: The system is expected to deliver the video in full-length to every peer.
- *Robust failure recovery*: The failure recovery protocol should quickly reorganize the network with the available peers.
- *Small control overhead*: The control overhead must be kept small to make the system scalable.

3 Overlay Network Structure for P2P VoD

As it was previously said, a key aspect of any P2P VoD system is the organization of participating peers into an overlay structure. Based on this component P2P VoD streaming systems can be broadly classified into two categories: *tree-based* and *mesh-based*. Both structures were also presented in the previous chapter. There are other systems which combine the conventional client-server architecture with the P2P *mesh-based* approach. This structure is called *hybrid network*.

3.1 Tree-Based Network Structure

Tree-based P2P systems were originally designed to be used as multicast structures at the application layer without network layer support. However, in this kind of system the users are synchronous, and in a VoD service the users are asynchronous. Therefore, the use of a tree-based P2P system to provide VoD services is not direct. In the last years, several mechanisms have been proposed to solve this problem. For example, in [6] the authors designed a tree-based P2P VoD system inspired by the patching scheme [11]. Users that arrive within a predefined time interval *T* constitute a session, and together with the server they form a tree-based P2P system. The server starts streaming the entire video over the tree, and when a new client joins the session, it joins the tree and retrieves the stream from it. In addition, the new client must obtain a *patch*, the initial portion of the video that it has missed (from the start of the session to the time it joined the base tree). The patch is available at the server as well as other users who have already cached the patch. Figure 1 illustrates a snapshot of the solution above when a new user arrives at time 40. Each user is marked with its arrival time to the system. It shows two sessions, starting at time 20 and 31, respectively, with time interval *T* equal to 10. User *E* has to create a new session because its arrival time to the system (31) is out of the time interval *T* of the user *A* (20 + 10 = 30). One major drawback of tree-based streaming systems is their vulnerability to peer departure, which will temporarily disrupt video delivery to all peers in the subtree rooted at the departed peer.

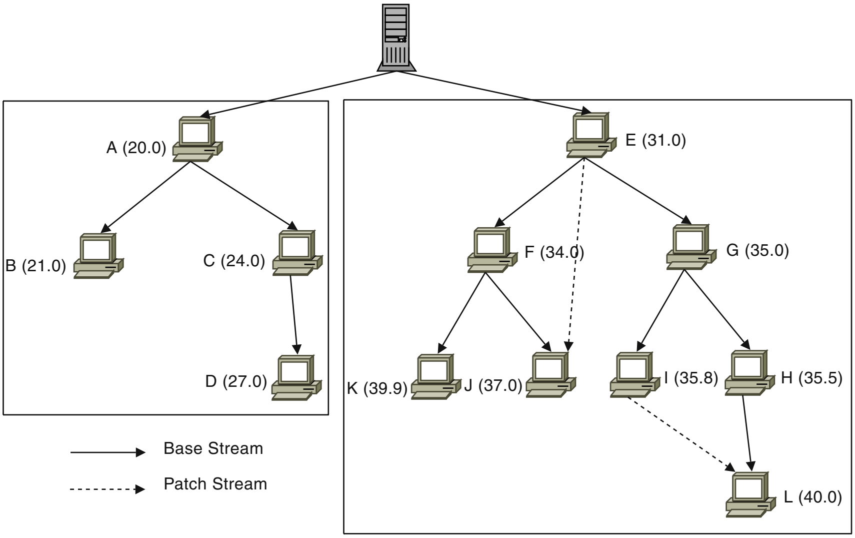


Fig. 1 A snapshot of the patching scheme with $T = 10$ at time 40

3.2 Mesh-Based Network Structure

Peers in mesh-based P2P systems are not confined to a static topology, they dynamically connect to a subset of random peers in the system. Figure 2 represents an overview of a mesh-based P2P VoD architecture. This kind of system tries to achieve fast file downloading by swarming. With this method, a file is divided into small size segments and the server disperses them to different users. The users download from their neighboring peers the segments that they currently do not have, and they have to be received before their playback time. To fully utilize users upload bandwidth and achieve the highest downloading throughput possible, the segments at different users have to be different from each other so that there is always something to exchange. This is the so-called *diversity requirement* in mesh-based P2P system. On the other hand, since multiple neighbors are maintained at any given moment, mesh-based systems are highly robust to peer failures. Therefore, this chapter is going to concentrate on mesh-based approaches. Within this kind of networks, several suppliers search and data scheduling mechanisms for both storage-forwarding and buffer-forwarding approaches are going to be introduced.

3.3 Hybrid Network Structure

Hybrid network structures try to incorporate the P2P *mesh-based* downloading methodology into the conventional client-server VoD service to have the best of both solutions. There exist several hybrid solutions. Some of them, like BASS [3] and PONDER [5], use a P2P network to download the majority of the content, and the server provides the missing data when the playback time is reached. In PONDER, when the playback time approaches, the client prepares a *missing data vector* $V_{missing}$, which is a bit map indicating missing segments. The missing data vector

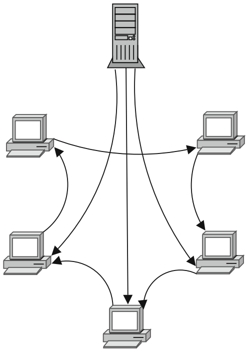


Fig. 2 An overview of P2P mesh-based architecture. Each peer stores some segments for serving others

is sent to the server together with the deadline for these segments, and the server streams them. In addition, in PONDER, the server also streams the first segments to a new client in order to reduce the startup delay.

Other hybrid systems, like PPB [10] and SPB [9], use the P2P network to download the first segments and the rest of the segments are retrieved from a multicast or broadcast transmission performed by a server. Specifically, in PPB (P2P Batching) the server starts streaming the hole video in a multicast channel. When a new client joins the system, it downloads the leading portion of the video from a P2P network with the nodes that previously arrived at the system. In SPB (Skyscraper-based Peer-to-Peer Broadcasting) the video is partitioned into X segments. While the first segment is delivered by P2P manner, the rest of $X-1$ segments are placed on the individual broadcast channels and transmitted periodically. When a new client is admitted to the system, it first joins the P2P network for the requested video. Once the first segment for the requested video has been received from the supplying peers in the P2P network, the client downloads the other video segments from the broadcast channels.

4 Forwarding Approaches

The content forwarding policy is another important component of any P2P VoD system, that is, the way the multimedia content is stored and transmitted to each participating peer through the overlay. There are two main categories for this component: the *buffer-forwarding* and the *storage-forwarding* approaches. There also exist *hybrid-forwarding* approaches which integrate both the buffer-forwarding and the storage-forwarding. In addition, several *pre-fetching* techniques are sometimes used to manage the traffic congestion or dynamic departures of peers.

4.1 Buffer-Forwarding

In the buffer-forwarding approach, each client caches a limited number of segments around its current play offset, and it exchanges its cached segments with partners who have close play offset and can thus provide the expected data with high probability. The users need to search for such partners when joining or taking a VCR operation. Since the contents buffered in one peer are continuously changing, an additional structure is needed for the partner search. In this structure the participating peers organize themselves based on the playback progress. After locating the partners, the user collaborates with them to schedule the data transmission and exchange the content. Figure 3 depicts the general framework of a typical buffer-forwarding P2P VoD system. It is noticed that the users form two types of overlay networks: the *index overlay*, for suppliers search, and the *data overlay*, for media transmission.

This approach has two main drawbacks: First, a peer only redistributes the video, that it is currently watching, to the peers that are watching approximately the same video scene, and it does not redistribute the content that it has stored in its storage. Second, the upload bandwidth of the peers that have not content in their buffers cannot be utilized.

4.2 Storage-Forwarding

In the storage-forwarding approach each peer stores a great number of random video segments at its local storage (such as hard disk). This approach can be used when peers have a large storage capacity. For example, in VMesh [23] each peer keeps a list of peers who have the previous and the next video segments. Following the list, it can quickly find the peers with the next requested segments. Furthermore, a peer also keeps a list of peers storing the same segment for load balancing purpose. If a node is loaded, it redirects some of its requests to other peers on its list. The main drawback of this kind of systems is the need to have a big storage capacity in order to achieve a good performance. In addition, the upload bandwidth of the peers has to be high enough to support several connections.

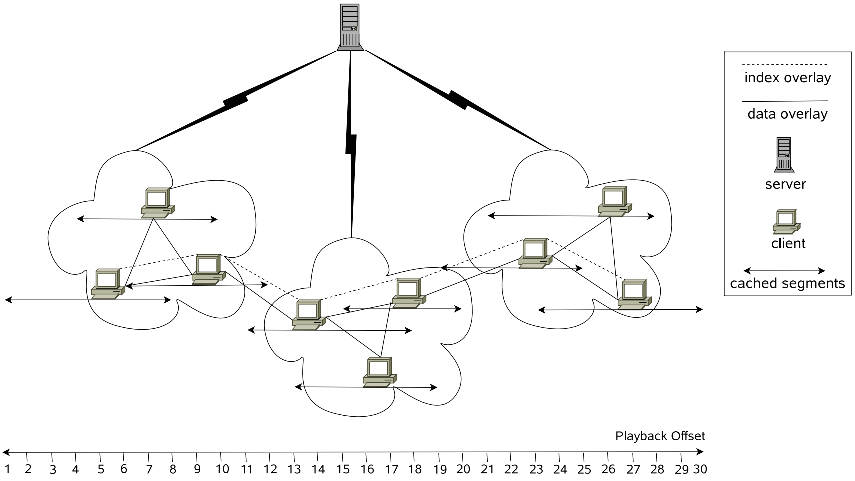


Fig. 3 General framework of a typical buffer-forwarding P2P VoD system

4.3 Hybrid-Forwarding

A hybrid-forwarding P2P VoD architecture integrates both the buffer-forwarding and the storage-forwarding approaches. The peers watching the same video form an overlay with the buffer-forwarding approach, and each peer replicates one or multiple segments of the video it has watched before in its storage.

An interesting hybrid-forwarding P2P VoD architecture is proposed in [8]. In the example shown in Fig. 4, peers 3, 7, 130, 118, 104 and 1 form video-1 buffer-forwarding overlay, and peers 134, 34, 115, 43, 22 and 1 form video-2 buffer-forwarding overlay. Some peers watching video 2, such as peer 115, can forward their stored segments to the peers watching video 1 (e.g. peer 7). The idle peers (e.g. peer 27, 12, and 75) are encouraged to contribute their stored segments to serve other peers. In the hybrid-forwarding architecture, the peers may have both buffer-forwarding and storage-forwarding outgoing links (e.g. peer 115). These two kinds of outgoing links participate in different video sessions and compete for the limited upload bandwidth. Furthermore, when the buffer-forwarding parents of peer 7 (e.g. peers 130 and 118) jump to other positions, peer 7 can still have a stream supply from its storage-forwarding parent (e.g. peer 115).

4.4 Pre-Fetching

The idea of pre-fetching is that nodes download future segments, though they are not immediately useful. This mechanism increases the playback continuity providing easy access to segments when they are needed and minimizing the overhead of segments propagation from remote locations. It is specially recommended when managing the traffic congestion or dynamic departures of peers. A representative example is proposed in [12]. This system consists of two independent modules: the pre-fetching module, which downloads the full file with lower priority, and the sliding window, which downloads the immediately needed segments with higher priority.

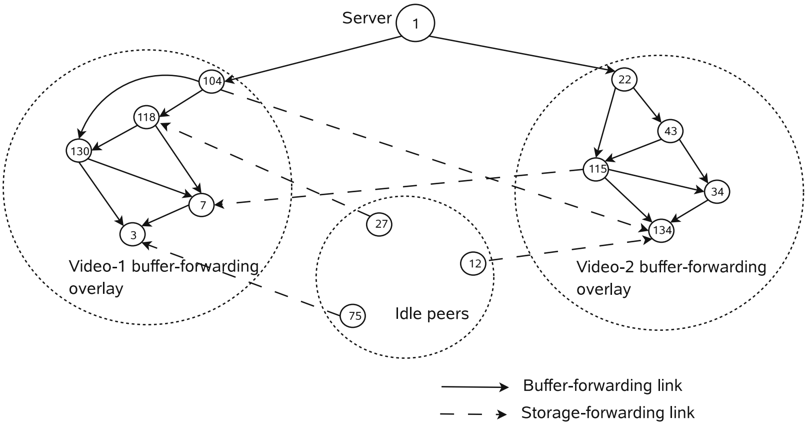


Fig. 4 A hybrid-forwarding P2P VoD system

However, pre-fetching mechanisms can also be used with other objectives. For example, in BulletMedia [21] the pre-fetching mechanism tries to ensure that all segments are replicated at least *k* times, where a typical value of *k* is 4. When a peer has spare incoming bandwidth, it determines which segments it should pre-fetch in order to ensure good diversity across the overlay. To achieve this, the peer examines its local cache and determines the set of segments it is not replicating currently. Then it selects segments at random from this set, and performs a lookup in a DHT. The DHT response includes a count of the number of peers replicating the segments. If the number of replicas is below a pre-defined threshold then the peer begins to retrieve the segments.

In [7] the prefetching mechanism is used to reduce the seeking distance. In VoD applications with frequent seeks, the next position the peer will access may not be the following segment, it will probably be any other segment in the whole video. The proposal in this work is that each peer performs pre-fetching basing on the segment access probability, which is estimated from the seeking statistics in the previous sessions. In order to represent the seeking statistics, the authors employ Flajolet-Martin (FM) [4] sketches.

5 Suppliers Search

A key aspect in mesh-based P2P VoD systems is the design of mechanisms to find partners with needed segments. It is very challenging to develop an efficient suppliers search mechanism among a large population of collaborative peers. Due to scalability concerns, the search mechanism should consist of a distributed structure with sub-linear search efficiency. Today, some distributed structures have been proposed with logarithmic search efficiency, for example, the DHT (distributed hash table) structure.

5.1 Searching in Buffer-Forwarding

In the buffer-forwarding approaches, peers are sorted in the distributed search structures by the play offset. Given the play offset as the sorting key, these structures support searching mechanisms in a distributed manner. However, the maintenance cost of these search structures is not trivial, since all peers need to be registered in the structure. Some buffer overlapping may occur among the different nodes, and removing the nodes whose buffer range is fully covered by other nodes does not reduce the total buffer coverage. The BAS (buffer-assisted search) structure [2] tries to maintain as few peers as possible for better search efficiency. In other words, it wants to minimize the search structure size without sacrificing the search effectiveness. This problem can be formulated as the Minimum Buffer Cover (MBC), which is a variation of the well-known NP-hard problem Minimum Set Cover.

However, there are other systems that use more novel management schemes. For example, in RINDY [1], a video is divided into a series of timeslots which encapsulate the content of one second, and the buffer window can contain at most w timeslots. The distance between any two peers is calculated from their current playing positions. For instance, the distance from peer j to peer i is $d_{ji} = cur_j - cur_i$. If d_{ji} is negative, then the playing position of peer j is behind of that of peer i and peer j is called a *back-neighbor* of peer i; otherwise peer j is called a *front-neighbor* of peer i. The neighbors are organized into a series of concentric, non-overlapping logical rings according to their relative distances (see Fig. 5). The radius of the i-th ring is $w \cdot 2^i$ (the ith ring covers the area between the two circles with radii $w \cdot 2^{i-1}$ and $w \cdot 2^i$). A peer's innermost ring, whose ring number is zero, is responsible for collecting some neighbors with close playing positions (this ring is called *gossip-ring*). Outer rings are mainly used to improve the speed of lookup operations (these rings are called *skip-rings*). For the *gossip-ring*, each peer keeps track of at most m neighbors, called *near-neighbors*; for each skip-ring, a peer tries to maintain at most k *front far-neighbors* and k *back far-neighbors*.

In [22] the use of a skip list is proposed to implement the suppliers search mechanism of a VoD streaming system. A skip list is a probabilistic data structure with efficiency comparable to a *binary search tree*, in which every element associates a key with a specific value. It is built in layers, where the bottom layer is a linked list ordered according to the key of every element, and each higher layer acts as a fast route for the lists below. An element in layer i appears in layer $i+1$ with some fixed probability p (two commonly-used values for p are $1/2$ or $1/4$).

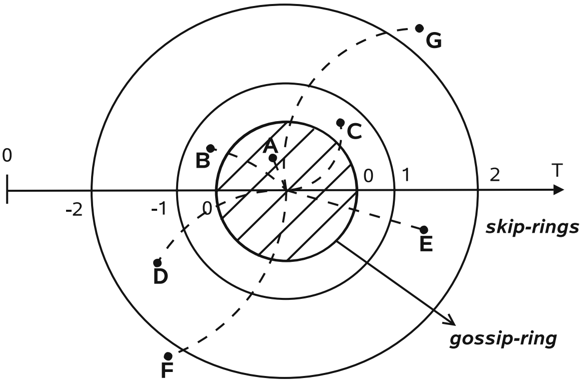


Fig. 5 Neighbor distribution over rings for peer P

A search for the value associated with a specific key begins at the head element in the top list, and proceeds horizontally traversing forward pointers that do not overshoot the element containing the searched key. When no more progress can be made at the current level of forward pointers, the search moves down to the next level. When no more progress can be made at level 1, the element that contains the desired key is in front of the current element. The pseudo-code of the search algorithm appears in Table 1, and Fig. 6 represents an example of this algorithm. In this example, the algorithm searches for the value associated with key 12. The mapping between a skip list and a VoD streaming overlay is straightforward: the playback offset of a client serves as its key in the skip list, and this key is updated over time according to the playback progress.

Table 1 Search in skip lists

```
Search(list, keySearch)
    x = list->header

    FOR i = list->level DOWNTO 1 DO
        WHILE x->forward[i]->key < keySearch DO
            x = x->forward[i]

    x = x->forward[1]
    IF x->key = keySearch THEN RETURN x->value
        ELSE RETURN failure
```

5.2 Searching in Storage-Forwarding

In the storage-forwarding approach, depending on the capacity of its local storage, each peer stores a number of segments randomly chosen from the *N* segments of the video. In this way, there are multiple copies of each video segment in the network. When a client wants to play a segment, it first looks for the supplying peers of that segment and then sends requests to those peers for the service. If there is no supplying peer, the requesting peer requests the media server for the target segment as the last resort. In this section the searching mechanisms of two representative storage-forwarding proposals are presented.

In Vmesh [23], each storage peer should register its stored segments in a DHT network in order to allow other peers to locate them. A new client searches for its segment of interest in the DHT network and starts playing the video when the re-

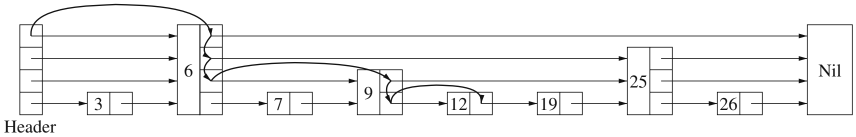


Fig. 6 Example of searching in skip lists

quested data arrives. Then, it continues to request the next segment when the current one is nearly finished. In order to shorten the segment location latency, the peers also form an overlay mesh among themselves. This overlay mesh is represented in Fig. 7. In this figure every circle represents a peer and the number inside represents the ID of the segment it holds. Each peer keeps a list of pointers (i.e., the IP addresses) pointing to some peers which store the next video segment and the previous video segment. By using the pointer list, clients could request for the locations of the next required segment. Besides, for the load balancing issue, each peer also keeps a list of pointers to some peers which are storing the same segment. If a node is loaded, it redirects some of its requests to other peers on its list.

BulletMedia [21] also uses a DHT to store information about content location within the peers of the mesh overlay. However, the DHT does not store information at a segment granularity, instead, sets of contiguous segments are deterministically grouped into chunks (e.g., 100 segments in a chunk), and each one is assigned a unique key (*chunkId*). Each peer monitors its content cache and when all associated segments with a particular *chunkId* are present, the peer inserts an entry into the DHT. When a peer performs a seek operation, it determines the *chunkId* of the block and then queries the DHT.

6 Data Scheduling

Data scheduling is an additional, important and challenging problem in mesh-based P2P VoD systems. The scheduling scheme in mesh-based P2P VoD systems has two main components: how to send requests to adequate neighbors, and how to evaluate the requests received from many peers.

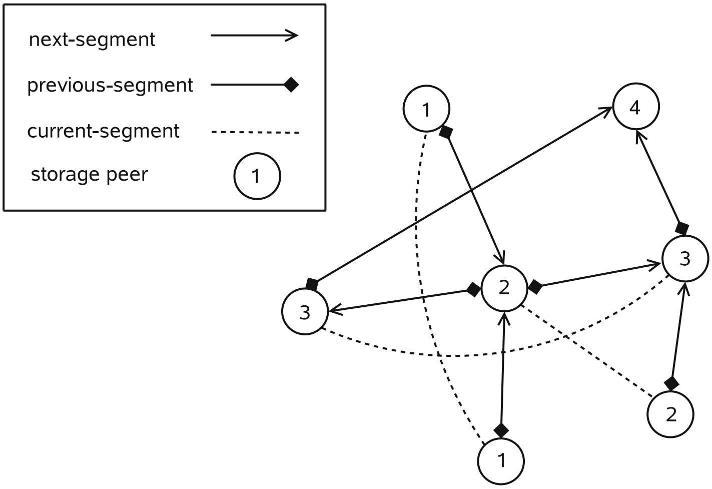


Fig. 7 In VMesh, storage peers hold several video segments and at the same time, they keep a list of peers who possess the next / previous / current segments for random seeking and load balancing purposes

6.1 Scheduling in Buffer-Forwarding

Many heuristics have been proposed to address the issue of buffer-forwarding approaches, such as round robin or smallest-delay. For example, in OCTOPUS [13] the system schedules packets based on partners' bandwidths. Each peer assigns an initial scheduling bandwidth to every partner ($W(x) = Sbps/N$), where $Sbps$ means the streaming rate of media file and N the number of partners and it is adjusted in data transferring according to the success ratio of scheduling. The system calculates the success ratio of scheduling for the past periods in function of the scheduling results from every partner. If success ratio is 1, it is obvious that the predefined scheduling bandwidth $W(x)$ is less than the actual bandwidth, and the scheduling bandwidth values can be enlarged to make full use of the bandwidth resource of partners. But if the success ratio is low, it is obvious that the scheduling bandwidth is too large and the service capacity of the partner is not so strong. The value of scheduling bandwidth may have to be decreased.

However, the previous schemes suffer from inefficient use of network resources in large and heterogeneous networks. Some recently proposed schemes use network coding (NC) to improve the system throughput. Wang and Liu [22] proposed directly applying the linear NC into VoD streaming. The original data segments are denoted as $c_1, c_2, ..., c_w$, where w is called *coding window (CW)*. A random linear coding on $\{c_i\}$ is a vector $f_i = \sum \beta_i \times c_i$, for $i \in (1...w)$, where the coefficient vector β_i is randomly generated in a finite field F_q of size q. f_i is referred to as *combined* data segments, which also spans a range of w. If all f_i are linearly independent, once a node has a subset of f_i that spans range w, it can recover all the w original segments by solving a set of linear equations.

However, it is difficult to apply network coding to VoD systems because some segments may miss the playback deadline before being decoded, which causes severe performance degradation. In conventional network coding the CW is always the same as the number of expected segments. The deadline-aware network coding (DNC) [2] scheme tries to use a CW that is as large as possible for better coding efficiency while controlling the CW size so that no segment misses its play deadline. DNC estimates the CW to be the maximum number of segments that can be retrieved from the partners before the most urgent deadline. Then the problem is to find an assignment for retrieving segment j from partner i, so that the number of received segments is maximized before the most urgent deadline. The authors transformed the DNC problem into the Max-Flow problem.

6.2 Scheduling in Storage-Forwarding

For the storage-forwarding approaches several modifications of the BitTorrent segment selection algorithm have been proposed. On the other hand, when individual peers are under common control (this is the case of residential home gateways or set-top boxes-STBs-under the control of a content provider), the use of rateless coding such as LT or Raptor codes is a new alternative to address this issue.

BitTorrent regards all segments as equally important, and it employs Tit-for-Tat (TFT) strategy in determining to which neighbors a peer serves content, (that is, a peer favors a neighbor who transmits data in return). However, in a VoD system the highest priority is given to the segments close to their playback time. BiToS [20] supports the VoD service by modifying the segment selection algorithm of BitTorrent. A peer in BiToS has three components, as shown in Fig. 8: The *received buffer* stores all the segments that have been received, the *high priority set* contains the segments close to their playback time which have not been downloaded yet, and the *remaining piece set* contains the segments that have not been downloaded. A segment in the *high priority set* is downloaded with probability p while one in the *remaining pieces set* is downloaded with probability $1-p$. By setting the value of p greater than 0.5, the blocks in the *high priority set* are given preference over the ones in the *remaining pieces set* to be downloaded earlier.

In [19] the peers are set-top boxes (STBs), which are under the control of a content provider. Each video is chopped into windows of contiguous data of size W, and it is pushed to a population of STBs (supposing that there are M STBs). The authors propose two data placement schemes: full-striping and code-based. The full striping scheme stripes each window in M blocks, each of size W/M, and each one is pushed to only one box. A full window is reconstructed at a particular box by concurrently downloading $M-1$ distinct blocks for the window from the other $M-1$ boxes. Code-based placement tries to reduce the maximum number of simultaneous connections that a box can serve to y (for some $y < M-1$). In order to achieve this, it applies LT coding [15]. It divides each window into k source symbols, and generates Ck coded symbols (C is called the expansion ratio). A viewer can reconstruct a window of a video by concurrently downloading any Cky/M distinct symbols from an arbitrary set of y boxes out of $(M-1)$ boxes.

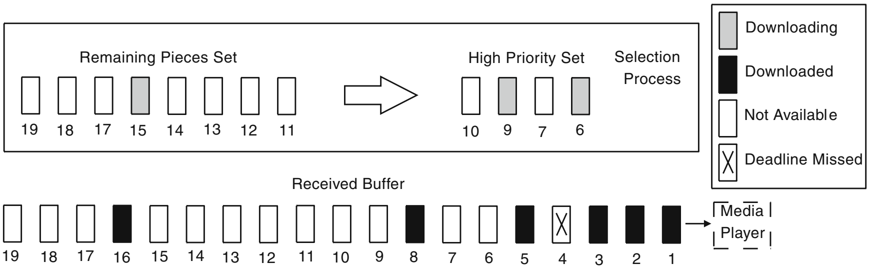


Fig. 8 BiToS peer structure

7 Technical Challenges

In this section some of the technical challenges to be overcome are discussed. It is commonly assumed that users want to collaborate in a P2P network however, in reality, this is not always the case. There could be many users that refuse to contribute bandwidth. Designing incentive mechanism that enable users to cooperate is absolutely necessary in these systems in order to achieve a good performance. It is well-known that the P2P file downloading protocol BitTorrent uses Tit-for-Tat (TFT) as incentive to induce peers to help each other. In P2P streaming, this does not work since many peers cannot contribute uploading bandwidth greater than or equal to the playback rate. This is related with the asymmetric nature of nodes, that is, nodes behind ADSL and cable can receive several hundreds of kilobits per second but can fundamentally only donate less.

On the other hand, NATs impose fundamental restrictions connectivity of nodes on an overlay and may prohibit direct communication with one another. In the last years several protocols have been proposed for dealing with NATs. For example, STUN (Session Traversal Utilities for NAT) [17] provides a means for an endpoint to determine the IP address and port allocated by a NAT that corresponds to its private IP address and port. It also provides a way for an endpoint to keep a NAT binding alive.

8 Summary

It is more challenging to design a P2P VoD service than a P2P live streaming system because although a large number of users may be watching the same video, they are asynchronous to one other. In this chapter the dominant problems for P2P VoD streaming have been presented.

An important component of any P2P VoD system is the organization of participating peers into an overlay structure. Based on this component these systems have been classified into two categories: tree-based and mesh-based. Hybrid networks, which combine the conventional client-server architecture with the P2P mesh-based architecture, have been also presented.

Another important component of any P2P VoD system is the content forwarding policy, that is, the way the multimedia content is stored and transmitted to each participating peer through the overlay. There are two main categories for this component: the buffer-forwarding and the storage-forwarding approaches. There also exist hybrid-forwarding approaches which integrate both the buffer-forwarding and the storage-forwarding.

In mesh-based P2P VoD systems, suppliers search and data scheduling are identified as the two main mechanisms. Several mechanisms for both storage-forwarding and buffer-forwarding approaches have been introduced. Finally, several technical challenges to be overcome have been presented.

9 Future Research Directions

Most P2P streaming systems are not *ISP-Friendly* because data exchanges among nodes are driven by content availabilities. That is, nodes connect to multiple random peers, local and remote, and exchange data between different networks. These video exchanges significantly increase the traffic volume on links within and between ISPs. How ISPs should manage the P2P video streaming traffic deserves further investigation to maintain the stability of their network infrastructures.

On the other hand, several ISPs have started to provide IPTV services. It will be beneficial for ISPs to integrate P2P technology into their IPTV systems to significantly reduce their server and network infrastructure cost. Many interesting research problems need to be addressed to develop an integrated IPTV solution for network providers and IPTV users.

Acknowledgement

This research has been supported by project grant TEC2007-67966-C03-01/TCM (CON-PARTE-1) and it is also developed in the framework of "Programa de Ayudas a Grupos de Excelencia de la Región de Murcia, de la Fundación Séneca, Agencia de Ciencia y Tecnología de la RM (Plan Regional de Ciencia y Tecnología 2007/2010)". Juan Pedro Muñoz-Gea also thanks the Spanish MEC for a FPU (AP2006-01567) pre-doctoral fellowship.

References

1. Cheng, B., Jin, H., Liao, X.: Supporting vcr functions in p2p vod services using ring-assisted overlays. In: Proceedings of the IEEE International Conference on Communications 2007 (ICC 2007), pp. 1698–1703 (2007)
2. Chi, H., Zhang, Q., Jia, J., Shen, X.: Efficient search and scheduling in p2p-based media-on-demand streaming service. IEEE Journal on Selected Areas in Communications **25**(1), 119–130 (2007)
3. Dana, C., Li, D., Harrison, D., Chuah, C.N.: Bass: Bittorrent assisted streaming system for video-on-demand. In: 2005 IEEE 7th Workshop on Multimedia Signal Processing, pp. 1–4 (2005)
4. Flajolet, P., Martin, G.N.: Probabilistic counting algorithms for data base applications. Journal of Computer and System Sciences **31**(2), 182–209 (1985)
5. Guo, Y., Mathur, S., Ramaswany, K., Yu, S., Patel, B.: Ponder: Performance aware p2p video-on-demand service. In: Proceedings of the IEEE Global Telecommunications Conference, 2007. GLOBECOM'07, pp. 225–230 (2007)
6. Guo, Y., Suh, K., Kurose, J., Towsley, D.: P2Cast: peer-to-peer patching scheme for VoD service. In: Proceedings of the 12th International Conference on World Wide Web, pp. 301–309 (2003)

7. He, Y., Guobin, S., Xiong, Y., Guan, L.: Probabilistic prefetching scheme for p2p vod applications with frequent seeks. In: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS 2008), pp. 2054–2057 (2008)
8. He, Y., Lee, I., Guan, L.: Distributed throughput maximization in hybrid-forwarding p2p vod applications. In: Proceedings of the 2008 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 2165–2168 (2008)
9. Ho, K.M., Lo, K.T., Feng, J.: Cooperative transmission strategy for video-on-demand system. In: Proceedings of the International Conference on Information Networking, 2008. ICOIN 2008, pp. 1–5 (2008)
10. Ho, K.M., Poon, W.F., Lo, K.T.: Peer-to-peer batching policy for video-on-demand system. In: Proceedings of First International Conference on Communications and Networking in China, 2006. ChinaCom'06, pp. 1–6 (2006)
11. Hua, K., Cai, Y., Sheu, S.: Patching: A multicast technique for true video-on-demand services. In: Proceedings of the 12th ACM International Conference on Multimedia, pp. 191–200 (1998)
12. Janardhan, V., Schulzrinne, H.: Peer assisted vod for set-top box based ip network. In: P2P-TV '07: Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV, pp. 335–339 (2007)
13. Liao, X., Jin, H.: Octopus: A hybrid scheduling strategy for p2p vod services. In: GCC '07: Proceedings of the Sixth International Conference on Grid and Cooperative Computing, pp. 26–33 (2007)
14. Liu, Y., Guo, Y., Liang, C.: A survey on peer-to-peer video streaming systems. Peer-to-Peer Networking and Applications **1**(1), 18–28 (2008)
15. Luby, M.: Lt codes. In: FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science, p. 271 (2002)
16. Pplive. http://www.pplive.com/ (2005)
17. Rosenberg, J., Mahy, R., Matthews, P., Wing, D.: Session traversal utilities for nat (stun). RFC 5389 (Proposed Standard) (2008). URL `http://www.ietf.org/rfc/rfc5389.txt`
18. Sopcast. http://www.sopcast.org/ (2005)
19. Suh, K., Diot, C., Kurose, J., Massoulié, L., Neumann, C., Towsley, D.F., Varvello, M.: Push-to-peer video-on-demand system: Design and evaluation. IEEE Journal on Selected Areas in Communications **25**(9), 1706–1716 (2007)
20. Vlavianos, A., Iliofotou, M., Faloutsos, M.: Bitos: Enhancing bittorrent for supporting streaming applications. In: Proceedings of INFOCOM 2006. 25th IEEE International Conference on Computer Communications, pp. 1–6 (2006)
21. Vratonjić, N., Gupta, P., Knežević, N., Kostić, D., Rowstron, A.: Enabling dvd-like features in p2p video-on-demand systems. In: P2P-TV '07: Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV, pp. 329–334 (2007)
22. Wang, D., Liu, J.: A dynamic skip list-based overlay for on-demand media streaming with vcr interactions. IEEE Transactions on Parallel Distributed Systems **19**(4), 503–514 (2008)
23. Yiu, W.P.K., Jin, X., Chan, S.H.G.: Vmesh: Distributed segment storage for peer-to-peer interactive video streaming. IEEE Journal on Selected Areas in Communications **25**(9), 1717–1731 (2007)
24. Zhang, X., Liu, J., Li, B., Yum, T.S.: Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming. In: Proceedings of IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies., pp. 2102–2111 (2005)

Part IX
Mobile P2P

Peer-to-Peer Overlay in Mobile Ad-hoc Networks

Marcel C. Castro, Andreas J. Kassler, Carla-Fabiana Chiasserini, Claudio Casetti, and Ibrahim Korpeoglu

Abstract Wireless multi-hop networks such as mobile ad-hoc (MANET) or wireless mesh networks (WMN) have attracted big research efforts during the last years as they have huge potential in several areas such as military communications, fast infrastructure replacement during emergency operations, extension of hotspots or as an alternative communication system. Due to various reasons, such as characteristics of wireless links, multi-hop forwarding operation, and mobility of nodes, performance of traditional peer-to-peer applications is rather low in such networks. In this book chapter, we provide a comprehensive and in-depth survey on recent research on various approaches to provide peer-to-peer services in wireless multi-hop networks. The causes and problems for low performance of traditional approaches are discussed. Various representative alternative approaches to couple interactions

Marcel C. Castro
Department of Computer Science, Karlstads University, Universitetsgatan 2, SE-651 88, Karlstad, Sweden, e-mail: `Marcel.Cavalcanti@kau.se`

Andreas J. Kassler
Department of Computer Science, Karlstads University, Universitetsgatan 2, SE-651 88, Karlstad, Sweden, e-mail: `Andreas.Kassler@kau.se`

Carla-Fabiana Chiasserini
Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy, e-mail: `chiasserini@polito.it`

Claudio Casetti
Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy, e-mail: `casetti@polito.it`

Ibrahim Korpeoglu
Department of Computer Engineering, Bilkent University, 06800 Ankara, Turkey,
e-mail: `korpe@cs.bilkent.edu.tr`

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_37, © Springer Science+Business Media, LLC 2010

between the peer-to-peer overlay and the network layer are examined and compared. Some open questions are discussed to stimulate further research in this area.

1 Introduction

A mobile ad-hoc network (MANET) is a collection of autonomous mobile nodes that communicate using wireless links without support from any pre-existing infrastructure network. In such a multi-hop network, nodes operate as both end hosts and routers, forwarding packets wirelessly towards other mobile nodes that may not be within the direct transmission range of each other. MANETs are formed with the key motivation that users can benefit from collaborations with each other. Wireless mesh networks (WMN) are comprised of a wireless mesh backbone formed of quasi-stationary wireless mesh routers which wirelessly relay packets generated by (mobile) mesh clients, that connect to the wireless mesh routers like to normal access points. WMNs are emerging as an attractive infrastructure for next generation wireless access networks and they share many properties with MANETs such as multi-hop forwarding. While MANETs typically operate standalone and more autonomous, Internet access for MANETs and WMNs is desirable. Multi-hop networks such as MANETs or WMNs have been considered to support future ubiquitous and pervasive computing scenarios, and therefore will be intrinsic part of the future Internet.

Recently, applications based on the Peer-to-Peer (P2P) communication paradigm are increasing in popularity. Examples are popular file-sharing applications (e.g., Kazaa [41], Gnutella [56]), upcoming P2PSIP solutions for Voice over IP, or P2P video streaming that use P2P techniques to form an overlay on top of existing networks. P2P computing refers to technology that enables two or more peers to collaborate spontaneously in a network of equals (peers) by using appropriate information and communication systems without the necessity for central coordination. In that sense, P2P networks are overlay networks typically operated on infrastructured (wired) networks, such as the Internet. However, the P2P overlay network is dynamic, where peers come and go (i.e., leave and join the group) for sharing files and data through direct exchange. Such peer-to-peer communication paradigm will be very important in wireless multi-hop networks as centralized servers might not be available or located in the Internet. Therefore, P2P will be an interesting alternative for decentralizing services or making its own local resources available in the multi-hop network to serve local user communities.

P2P overlay networks in the Internet and mobile ad-hoc networks share many key characteristics such as self-organization and decentralization due to the common nature of their distributed components [33]. They also share a high degree of dynamicity as nodes can join and leave the network at any given time. These common characteristics lead to further similarities between the two types of networks: both have a frequently changing topology caused by nodes joining and leaving dynamically. Also in a MANET terminals are mobile and communication follows a hop-by-hop connection establishment.

The common characteristics shared by P2P overlays and MANETs also dictate that both networks are faced with the same fundamental challenge, that is, to provide connectivity in a decentralized and dynamic environment. Thus, there exists a synergy between these two types of networks in terms of the design goals and principles of their routing protocols and applications built on top: both P2P and MANET routing protocols and applications have to deal with dynamic network topologies due to membership changes or mobility.

In addition, P2P overlays over the Internet rely on the IP routing infrastructure, which is resource rich especially in terms of bandwidth availability. Mobile ad-hoc networks, instead, are rather limited in bandwidth, and a high maintenance traffic, as it is used currently in structured overlay networks, will lead to scalability problems when legacy P2P services are used "as-is" in multi-hop environments. Thus, one of the main issues is how to efficiently provide the same kind of P2P services implemented in legacy wired networks in multi-hop networks, and how to enable efficient overlay services and applications on the resource constrained wireless multi-hop networks.

The common characteristics, challenges, and design goals between P2P overlays and mobile ad-hoc networks point to new research directions in wireless networking, that is, to exploit the synergies between P2P overlays and multi-hop networks such as MANETs. There are several examples where knowledge on interactions between P2P and MANET can either help to realize more efficient P2P networks and services on top of multi-hop networks or will lead to the design of better and more scalable routing protocols [8, 24, 53, 70]. Understanding such interactions will also help to clarify, what support from routing layer shall be required for scalable operation of P2P on top of heterogeneous mobile networks.

The remaining part of this chapter is then organized as follows. In Section 2, we give a brief overview on structured and unstructured overlay networks. We introduce wireless multi-hop networks and highlights key properties of wireless operation and multi-hop forwarding. The challenges encountered while deploying P2P services in mobile ad-hoc networks are detailed in Section 3. Section 4 provides a detailed survey of related approaches including work on both unstructured (e.g., flooding based protocols, unstructured key lookup, and proactive search routing) and structured (e.g., topology dependent and topology independent) P2P overlays for MANETs. Recent studies, such as ORION [39], MPP [26], P2PSI [31], ZP2P [38], VRR [8], SSR [24], CrossROAD [18], MADPastry [70], MeshChord [7], and Hashline [61] will be introduced. Additionally, the respective advantages and disadvantages are evaluated. Section 5 introduces important P2P application scenarios for MANETs, such as decentralized name service (e.g., MAPNaS [71] and P2PNS [2]), overlay-based multicast (e.g., XScribe [19]), and multimedia services (e.g., P2PSIP [20]). Finally, Section 6 concludes the chapter.

2 Overview on Peer-to-Peer and Ad-Hoc Networks

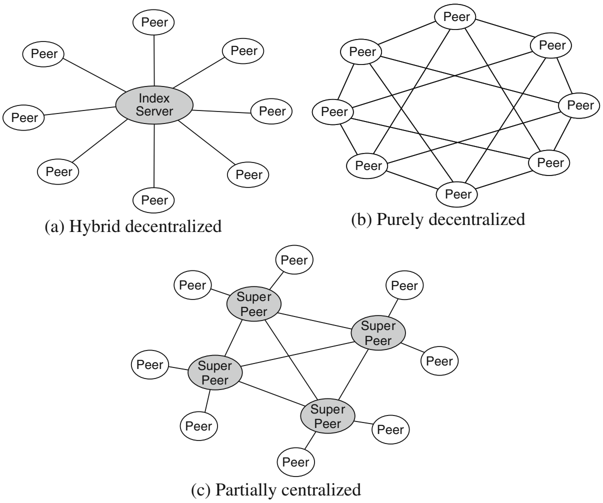
Wireless multi-hop networks feature several peculiar aspects which significantly differentiate them from other wireless systems and pose serious technical challenges. In this section, we highlight the main characteristics of these systems and discuss some of their most challenging issues, i.e., wireless multi-hop communication, mobility, and traffic routing in multi-hop networks.

2.1 Peer-to-Peer Overlay Networks

We begin however with a brief overview on peer-to-peer networks. There are numerous peer-to-peer overlay networks proposed with very different architectures and protocols. The architectures for P2P overlays can be categorized into two main classes: unstructured P2P overlays and structured P2P overlays.

Unstructured overlays do not impose a rigid relation between the overlay topology and where resources or their indices are stored. This has a number of advantages like; easy implementation and simplicity, supporting dynamic environments and keyword search (instead of exact match queries). But the major drawback of such overlay is scalability problem. Search operation for a resource may take a long time and consume network resources extensively, since most of the time there is no relation between the name of resources and their locations. Depending on the degree of centralization, unstructured P2P overlays are usually classified into three sub-categories: 1) hybrid decentralized overlays such as Napster, Publius, and Bittorent [45, 52] (Figs. 1a, 2) purely decentralized overlays such as initial version of Gnutella and Free Haven [56, 57] (Figs. 1b, 3) partially centralized overlays such as Gnutella version 0.6, Fasttrack/Kazaa, Morpheus, Overnet/eDonkey2000 [30, 41, 56] (Fig. 1c). In all categories, the resources (or services) are totally distributed to peers and there is usually no relation between the locations of resources and the network topology. But depending on the category, central or distributed indices, clustering, super-peer concept, caching and replication can be used [1, 43].

A common feature provided by peer-to-peer overlay networks is a lookup service (i.e., searching for resources) handling flat identifiers with an ordinary query-response semantic. Such a service is often implemented using DHTs (Distributed Hash Tables), such as CAN, Chord, Pastry, and Bamboo [54, 55, 58, 62] . Unlike unstructured P2P networks with their random topology, DHTs impose a structure on the overlay topology by no longer choosing routing table entries arbitrarily. Instead, routing table entries have to satisfy certain criteria depending on the respective DHTs. At the core of each DHT lies the ability to route a packet based on a *key*, towards the node in the network that is currently responsible for the packet's *key*. This process is referred to as *indirect* or *key-based* routing. This structure enables DHTs to introduce an upper bound on the number of overlay hops towards the node currently responsible for the packet's key. This upper bound is commonly $O(logn)$, with n being the number of nodes in the network. This bound is achieved through



(a) Hybrid decentralized

(b) Purely decentralized

(c) Partially centralized

Fig. 1 Unstructured P2P overlays

routing strategies employed by the respective DHTs. Those strategies include reducing the Euclidean distance in the overlay ID space to the destination in each overlay routing step (e.g., CAN [54]), halving the numerical distance to the destination in each routing step (e.g., Chord [62]), or increasing the length of the matching prefix/suffix between the current node's overlay ID and the key in each overlay routing step (e.g., Pastry [58] and Bamboo [55]). Although DHTs can route packets very efficiently in comparison to unstructured P2P networks, they usually induce higher overhead due to the need for maintenance traffic of their routing tables. The maintenance traffic routine can be initiated by network change, such as in Chord and Pastry, or within certain periodicity regardless of network status, such as in Bamboo. While reactions to changes in the routing layer operate on very small timescale, reactions to changes in overlay structure are not so fast. In [55], the approach to use periodic updates has been shown to be beneficial during churn or in dynamic network, since it does not cause management traffic bursts during congestion. As we will show in Section 4, management traffic can impact network performance when applied to bandwidth limited wireless environments. However, as argued by [25], DHT approaches outperform unstructured approaches when the number of nodes, the number of objects, or the query rate increases, since they do not introduce flooding in the network.

2.2 Characteristics of Wireless Multi-hop and Mobility

Wireless multi-hop communication has many use cases, both in standalone deployments, but also to extend the reach of infrastructure, e.g., hotspots. Such wireless communication involving potentially multiple intermediate nodes poses several fundamental challenges, also stemming from hidden and exposed terminals resulting in packet loss, and high and variable delay and thus low performance in general. Several of these factors play a significant role in any wireless communication scenario. However, as communication is extended to multiple hops, several new wireless issues come into play. Single hop communication results in most cases in a single collision and interference domain. In contrast, in multi-hop cases the roles of collision and interference become more complex and depend on many factors such as radio environment, modulation schemes, transmission power, or sensing ranges. As a result, adjacent links and even links further separated, affect each other during transmission and they might have to share the wireless channel. In single channel networks, a two-hop configuration hence effectively halves the available bandwidth. Other links still within interference range also might affect links further down a multi-hop path, reducing the link bandwidth even further. Such behavior has many subtle performance implications to higher layers such as TCP [32], which are not visible in single hop networks.

To alleviate such problem, in WMNs mesh routers may be equipped with multiple radios (such as of-the-shelf 802.11a/b/g cards) to simultaneously transmit/receive over different orthogonal frequency channels. However, to fully exploit the available resources, it is necessary to develop mechanisms to effectively assign available channels to a limited number of radio interfaces per node. If a mesh is rather unplanned or channel allocation is done poorly, interference might be quite high leading to the same problems.

Another problem area is mobility of nodes, quite common to MANET scenarios. as a result, the network might become disconnected for a long period or the high mobility might lead to frequently changing communication paths. Such effects impose several challenges such as long delays, disrupted communications, and intermittent connectivity to communication protocols. As a result, most higher layer protocols such as TCP cease functioning or show dramatically low performance. Therefore, commonly assumed communication design principles such as the permanent availability of a dedicated end-to-end path have to be reconsidered leading to new communication paradigms that are significantly more delay tolerant than common approaches such as digital postal service through *store-carry-forward* message delivery. This style of delivery carries information between intermittent communication opportunities, and might be an attractive alternative of enabling communication where it is otherwise impossible. Such communication paradigms might also be useful for other contexts such as satellites networked into an interplanetary Internet [6] or postal service like data delivery into rural areas where communication infrastructure is not available [13]. Instead of assuming an always on connection, communication entities rather carry information between intermittent communication opportunities, leading to the opportunistic communication paradigm.

2.3 Traffic Routing in Multi-hop Networks

Routing is an essential function for Internet and also very important for wireless multi-hop networks, e.g., MANETs. Indeed, while at the MAC and physical layer it is commonly assumed that the IEEE 802.11 standard is adopted, a large number of different proposals on traffic routing have been presented within the IETF (The Internet Engineering Task Force) and are still under discussion.

Typically routing protocols in MANETs can be classified in flat and hierarchical schemes. Flat routing protocols distribute information as needed to any network node that can be reached or receive information. No effort is made to organize the network or its traffic, only to discover the best route hop-by-hop to a destination by any path. Hierarchical routing protocols, instead, group nodes together by function into a hierarchy, e.g., if there are powerful nodes, they may be selected as backbone routers, while lower powered node may be used for access purposes.

In the context of wireless ad-hoc and mesh networks, flat routing schemes have been far more successful than hierarchical solutions, thus, below, we focus on flat routing and review the most relevant schemes that have been proposed in the literature as well as those solutions that are mostly used in practical implementations. On the other hand for more opportunistic communication style in delay tolerant networks, new type of more probabilistic routing protocols have been developed as the main challenge is to cope with long periods of disconnection and opportunistically exploit communication possibilities.

2.3.1 Topology-Based Schemes

The routing protocols falling in this category exploit information related to the network topology. They can be further classified in (i) reactive protocols and (ii) proactive protocols. Reactive schemes create routes only when required by a source node. Once a route is established, it is maintained by a route maintenance procedure until either the source does not need the route any longer or there is no available path in the network. Examples of reactive solutions are the well known Ad-hoc On Demand Distance Vector (AODV) [50] routing and Dynamic Source Routing (DSR) [34] protocols. In AODV, when a route to a new destination is needed, the node broadcasts a RREQ (Route REQuest) message to find a route to the destination. A route can be determined when the RREQ reaches either the destination itself, or an intermediate node with a "fresh enough" route to the destination. A "fresh enough" route is a valid route entry for the destination whose associated sequence number is at least as great as that contained in the RREQ. The route is made available by unicasting a RREP (Route REPly) back to the origination of the RREQ. Each node receiving the request caches a route back to the originator of the request, so that the RREP can be unicast from the destination along a path to that originator, or likewise from any intermediate node that is able to satisfy the request. While AODV builds and maintains routing tables at every node, DSR obtains and encodes the source route in each packet header to the destination. It follows that DSR leads to a greater overhead with respect to AODV, although it can handle both unidirectional and bidirectional links and allows nodes to store more than one route for each source-destination pair.

Proactive schemes, instead, attempt to continuously maintain consistent, up-to-date routing information from each node to any other node in the network. As in AODV, every node has one or more tables, which are used to store routing information; upon topology changes, a node propagates update messages throughout the network in order to maintain a consistent view. Hence, in highly dynamic networks the overhead of proactive approaches is significantly higher than with reactive schemes, however when proactive solutions are applied, nodes always store routes to any possible destination in the network. Among the most interesting proactive solutions, there are the Optimized Link State Routing Protocol (OLSR) [14] and BATMAN (Better Approach to Mobile Ad-hoc Networking) [47], which deserve special attention because, along with AODV, are the protocols typically used in practical implementation of MANETs and mesh networks.

OLSR is a link-state routing protocol which exploits Hello and Topology Control (TC) messages to discover and then discriminate link state information throughout the ad-hoc network. Individual nodes use this topology information to compute next hop destinations for all nodes in the network using shortest hop forwarding paths. More specifically, using Hello messages the OLSR protocol performs a distributed election of a set of multipoint distribution relays (MPRs), such that there exists a path to each of its 2-hop neighbors via a node selected as an MPR. These MPR nodes then source and forward TC messages which contain the MPR selectors. Such approach has several benefits: the forwarding path for TC messages is not shared among all nodes but varies depending on the source, only MPRs source TC messages, and not all links of a node are advertised but only those which represent MPR selections.

BATMAN has been specifically designed for wireless mesh networks. The basic idea is to divide the knowledge about the best end-to-end paths between nodes in the mesh to all participating nodes. Each node perceives and maintains only the information about the best next hop towards all other nodes. Thereby, the need for a global knowledge about local topology changes becomes unnecessary. Additionally, an event-based but flooding mechanism prevents the occurrence of contradicting topology information and limits the amount of topology messages flooding the mesh (thus minimizing overhead of control-traffic). Since it adopts a hop-by-hop forwarding approach, BATMAN may be particularly suitable for networks whose connectivity level is not very high.

2.3.2 Geographic-Based Routing

Geographic routing protocols do not require knowledge of the network topology but rely on geographic position information, i.e., each node must be able to determine its own location and the source has to be aware of the location of the destination [63].

With this information, a message can be routed to the destination following different approaches. Greedy Perimeter Stateless Routing (GPSR) [37] tries to bring the message closer to the destination at each step, using only local information (greedy forwarding). Additionally, in regions of the network where such a greedy path does not exist, GPSR recovers by forwarding in perimeter mode. That is, a packet traverses successively closer faces of a planar subgraph of the full radio network connectivity graph until reaching a node closer to the destination, where greedy forwarding resumes. Alternatively, one can consider another notion of progress toward the destination, namely the minimum angle between neighbor and destination, as in Compass Routing [40] which however is not loop free. Clealy, whenever the destination node is mobile, geographic routing may be highly inefficient and the exchange of nodes location may lead to an exceedingly high overhead.

2.3.3 Probabilistic Routing

This approach has low complexity and is particularly suitable for networks with spotty connectivity, i.e., the so-called opportunistic networks. The basic idea is that context information, such as the users work address, the probability of physically meeting with other users or visiting particular places, can be exploited to identify suitable forwarders based on context information about the destination. Here, the mobility of nodes is exploited to deliver information from one node to another when they come into mutual communication range. Examples of protocols falling in this category are the Probabilistic ROuting Protocol (PROPHET) [42] and MaxProp [5]. PROPHET is an evolution of the epidemic approach that introduces the concept of delivery predictability. The delivery predictability is the probability for a node to encounter a certain destination. The delivery predictability for a destination increases when the node meets the destination, and decreases (according to an ageing function) between meetings. Transitivity is also taken into account, i.e., if node X frequently meets node Y, and node Y often meets node Z, then nodes X and Z have high delivery predictability with respect to each other. Also, when two nodes X and Y meet, they exchange their delivery predictability to destinations of the messages they store in their buffers, and messages are transfered from, say, X to Y only if Y's delivery predictability is higher than the one of X. The same technique is used by MaxProp, which, in addition, exploits information about frequently visited places.

3 Challenges of Deploying P2P Services in Mobile Ad-hoc Networks

The suitability of MANETs for applications that rely on a P2P architecture for information exchange presents designers with several challenges. Indeed, not only do mobile nodes require content delivery but they also act as content providers. Mobile users are expected to offer data services in an effective manner, despite the scarcity of bandwidth and the intermittent connectivity due to the highly-dynamic nature of

MANETs. Below, we list some of the technical challenges in delivering information to mobile users depending on a P2P organization.

Bandwidth Constraints

The challenge of introducing P2P concepts in multi-hop networks is that P2P overlays designed for the wired Internet rely on the IP routing infrastructure, which is resource rich especially in terms of bandwidth availability. As we have seen in Section 2.2, mobile ad-hoc networks are however rather limited in bandwidth. Therefore, a high maintenance traffic, e.g., as it is used currently in structured overlay networks, will lead to scalability problems when legacy P2P services are used "as-is" in multi-hop environments. One of the main issues is therefore how to efficiently provide the same kind of P2P services implemented in legacy wired networks in multi-hop networks, and how to enable efficient overlay services and applications on the resource constrained wireless environment. As it is presented in Section 4, several approaches try to overcome such challenge by integrating, or applying cross-layering techniques between the P2P and the MANET routing layer.

P2P Overlay Maintenance

Keeping the overlay routing table of each node up to date is one of the main tasks of a DHT system. Efficient routing depends on routing information being current and consistent. Invalid entries cause unnecessary overhead because of misrouted messages and suboptimal routing. To avoid these inconsistencies, DHT protocols employ maintenance mechanisms to keep the routing tables up to date. Typically, nodes probe their neighboring nodes via periodic ping request and response messages to learn whether they are still available or not. In MANETs, such maintenance traffic further contributes to congestion and collisions. As nodes mobility might lead to topology changes in the MANET routing layer, there might be potential for misrouted messages if the overlay routing and the MANET routing have inconsistent topology information. Also, triggering such maintenance traffic during network rerouting further contributes to network instability. To this end, cross-layer and integrated approaches are applied by, for example, exploiting the network routing messages (such as CrossROAD [18]) or cache information (such as SSR [24]) in order to maintain the P2P overlay.

Network Resiliency

In P2P networks with structured overlay, DHTs are considered to be very resistant against node failures. Backup and recovery mechanisms, that use distributed redundant information, ensure that no information is lost if a node suddenly fails. Depending on the subjacent DHT topology, the DHT experiences a reduced routing performance until the recovery has finished.

When DHT protocols are used in an ad-hoc environment, resilience is as a very important issue. The resilience of a DHT determines how much time may pass before expensive recovery mechanisms have to be evoked. As the quality of connections in ad-hoc networks is highly dependent on the environment and on the nodes mobility, nodes may often become temporarily inaccessible. If the recovery process is started too early, an avoidable overhead is caused if the node becomes accessible again. However, if the topological structure allows the DHT protocol to delay recovery mechanisms without losing routing capability, these costly recovery measures can be avoided and the maintenance costs of a DHT can be significantly reduced. As an example described in Section 4.3.1, [12] studies a compromise made between overlay management traffic in the overlay and network congestion to find a balance between lookup efficiency and management traffic overhead.

The worst case scenario is represented by a network where long delays and disrupted communications exist, as mentioned in Section 2.2. In this case, a node which is partly available and unavailable over a longer period of time can stress the whole network because of numerous join and leave procedures. Note that this scenario can easily be provoked by node movement along the network perimeter and, clearly, resilience mechanisms are needed to counteract the negative effects of this condition.

Routing Stretch

Unlike the P2P overlay in the Internet, where the neighbor is directly reachable using an underlying routing protocol, in the P2P overlay in MANETs scenario, contacting the neighbor may require going through multiple (wireless) hops. For this purpose, a pointer is maintained for every overlay's neighbor as a path through the network, consisting of a set of physical links from the node hosting the pointer to its overlay's neighbor.

When routing to a destination via DHTs, the node resorts to simple greedy routing: it selects the overlay's neighbor that makes the most progress in the ID space, and then forwards the packet along the pointer. Forwarding along this pointer can be achieved either through a source route inserted by the sender (e.g., SSR [24]) or through embedded state in the network in the form of incremental source routes to the overlay neighbor (e.g., VRR [8]). Both techniques will be discussed later. When the packet reaches the overlay-neighbor, it repeats the same greedy routing process until the packet makes it all the way to the destination. Therefore, routing proceeds at two levels: along the overlay from one overlay neighbor to another, and then from one overlay neighbor to another along the pointer source route via hop-by-hop through MANET routing protocols.

The ratio between the cost of selected route using the overlay-neighbor to the optimal shortest path routing through the MANET is defined as the *routing stretch* metric. Small routing stretch means that the selected route is efficient compared to the shortest path route. This is a key quantitative measure of route quality used by the P2P overlay, and affects global resource consumption, delay, and reliability. Thus, minimizing routing stretch is a critical issue for a multi-hop environment as both delay and packet loss increase significantly with the growth of the number of hops in the physical path.

Exploiting Heterogeneity

Another important point while deploying P2P overlay is which nodes should participate in the overlay given that not all nodes in a network may be overlay members [73]. While typically nodes in an overlay are initially placed manually, nodes may also dynamically and automatically decide to join and make services available. This issue may be especially important in multi-hop environments because overlay participation may be dictated by topological location which might change over time. Note, that other (e.g., physical) constraints may drive the decision to participate in the overlay. For example, nodes with limited power may not wish to act as overlay routers for other nodes.

Query Propagation

The propagation of query messages in the network is a critical aspect of the information sharing mechanism in P2P networks. Indeed, there are two contrasting requirements that arise in MANETs. On the one hand, queries for information must be forwarded by relays until they reach nodes holding such information, and some redundancy in forwarding is necessary to compensate for the unreliable nature of broadcast transmission of queries (i.e., no acknowledgments). On the other hand, congestion deriving from excessive spreading of queries and reply duplication must be limited. The simplest solution for query propagation is, of course, plain flooding of requests, but this is hardly viable in tightly-meshed, bandwidth-hungry wireless networks where congestion is more than likely. More refined approaches, are among others:

1. *Limiting query range*. The introduction of a query time to live (TTL) can shorten the reach of broadcast queries. A balance should be stricken between small values of TTL, which limit the success probability of a query, and query load.
2. *Smart relaying*. By forcing each relay to wait for a query lag time before rebroadcasting the query, the propagation of a request can be halted if a node in the neighborhood returns a response in the meantime (thus making any further query propagation useless). Coupling the query lag time with a smart selection of intermediate nodes for query rebroadcast may turn out to be very beneficial. As shown in [46], the Preferred Group Broadcasting (PGB) limits the network load through local, receiver-based decisions to rebroadcast a message. Intermediate nodes still wait for a lag time before rebroadcasting, however its length depends on the value of the signal-to-noise ratio (SNR) associated to the received message.
3. *Target selection*. Steering the queries toward the right direction is, of course, the main remedy against broadcast storms. Targeting a specific node that is known to store the information can be exploited at the application level, by leveraging the knowledge of the address and position of the last node encountered, which happened to cache the desired information. However, node targeting proves very inefficient in a MANET built by rapidly-moving nodes and running fast-dynamics applications. For this reason, a better approach is targeting *areas* of the network where the requested information is more likely to be cached, as proposed in [23].

Cooperative Content Caching

In purely decentralized overlays, a highly debated issue addresses the most appropriate caching strategy in an environment where a cache-all-you-see approach is clearly unfeasible but where the availability of sought-after information from nearby nodes is the key to success. This issue can be addressed through distributed caching strategies where nodes may cache highly popular contents that pass by, or record the data path and use it to redirect future requests [69]. Another viable solution is to eliminate information replicas among neighboring nodes [28], which however may require the nodes composing the MANET to coordinate their caching decisions. An interesting aspect is also how to minimize data access cost when network nodes have limited storage capacity. The scheme proposed in [64] makes use of cache tables that, in mobile networks, need to be maintained in a similar vein as routing tables.

As is clear from the above discussion, solutions to cooperative caching in mobile multi-hop networks, which are distributed and rely on lightweigth communication protocols, are still to be found. Finally, when different copies of the same information are injected in the network, maintaining cache consistency among the different nodes becomes a critical issue [10, 29].

Information Distribution and Survival

A final, critical issue pertains to achieving a desired distribution of the information within an area: regardless of how the information is distributed at the outset, the system should be able to identify where the information should be stored in the network area. In addition, a node storing the information acts as provider for that information; of course, this role may exact a high toll from nodal resources in terms of bandwidth or power consumption; it is advisable that the role of content provider be handed over to neighboring nodes quite frequently, without altering the information distribution. One or more nodes running out of power may affect the distribution of information and disrupt the P2P structure. Therefore, regardless of the initial information distribution, and of the density of nodes, information should never be allowed to die out. Related to the information survival is the evaluation of the minimum number of copies of a specific information that can satisfy users' needs (i.e., in terms of information retrieval time or response rate).

Security

Deploying security mechanisms in P2P networks is quite difficult due to the characteristics of P2P paradigm such as anonymity, decentralization, self-organization and frequent disconnections. Security in P2P over mobile ad-hoc networks is even more challenging due to node mobility and easy access to wireless channels. Most security solutions require use of public keys for authentication, shared secret establishment, or integrity checking, and hence somehow depend on a public key infrastructure (PKI) [36].

PKI is needed by asymmetric cryptography to establish the validity of the public keys. For this purpose, PKI stores digital certificates that attach a public key to the name of its owner by the digital signature of a trusted third party called the Certification Authority (CA). The management of certificates is a complex duty that requests a substantial infrastructure, especially in large-scale applications.

Integration of PKI and CAs, or a similar security infrastructure, into P2P over MANET is a challenging task due to ad-hoc and infrastructureless nature of the network and lack of centralized entities. Even in P2P networks with servers (hybrid centralized or partially centralized – see Section 2.1), these servers usually do not fully control the peer behaviors as much as servers can do in a conventional client-server model. Thus, the centralized architecture of PKI may introduce several important problems that contradict with the important characteristics of the P2P networks and MANETs. Additionally, PKI and security services may introduce substantial amount of control traffic into the network, which means more load to bandwidth-limited wireless channels of MANETs.

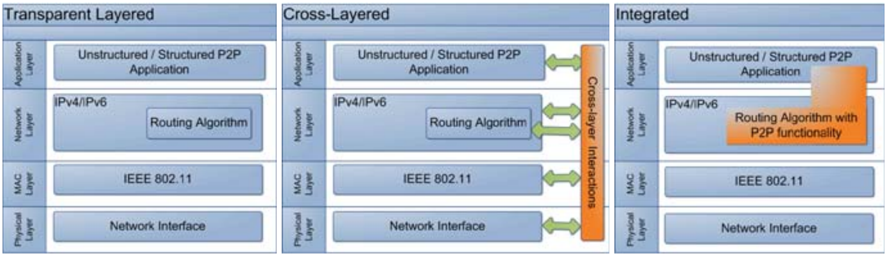
4 Overview of P2P Solutions for Mobile Ad-hoc Networks

In the following, we present and discuss various approaches to improve performance of peer-to-peer communication in wireless multi-hop networks, such as MANETs or WMNs. As several proposals try to integrate different layers to reduce bad interactions, we will first give an overview on different principles that guide the various integration and interaction possibilities, both in the area of unstructured (in Section 4.2) and structured (in Section 4.3) though there may be some overlapping similarities between the two.

4.1 Integration Principles Between P2P and MANET Routing Layer

One of the main differences between P2P and MANET is related to the level where they operate: P2P is essentially focused on building and maintaining overlay network connections at the application level, while the main focus of MANET is to provide multi-hop connectivity among wireless mobile nodes at the network level [59]. Due to the characteristics of multi-hop communication and the low resource availability in such networks, simply deploying a P2P overlay protocol as is on top of MANET routing layer (as shown in Fig. 2a) might cause poor performance, significant message overhead and redundancy in communication. The performance penalties of such transparent layering are better detailed in Section 4.3.1, where a packet level performance analysis of Bamboo over static multi-hop networks has been conducted.

One alternative for avoiding bad interactions between those layers is the paradigm of cross-layer design, as shown in Fig. 2b. Here, information from, for instance, the routing or MAC layer is made available at the peer-to-peer layer or vice versa in order to improve the performance. Various approaches implement different cross-layer interactions, as detailed in Sections 4.2 and 4.3. As a result, a cross-layered



(a) Layered design (b) Cross-layered design (c) Integrated design

Fig. 2 Design choices of P2P and MANET integration

design could offer a significant performance improvement if compared to the simple layered approach.

Another alternative to increase performance is to integrate peer-to-peer layer with routing layer beyond the strict layering rule [15], as shown in Fig. 2c. Typically new routing mechanisms (such as key-based routing) are developed, and try to implement peer-to-peer concepts in the routing layer itself. In the next sections we provide an overview of these approaches, by also trying to evaluate the key features of each of them.

4.2 Unstructured P2P Networks for MANETs

Actually, several works on the convergence of peer-to-peer systems and mobile ad-hoc networks have dealt with the straightforward implementation of unstructured P2P overlays in MANETs. Those approaches combine ad-hoc routing and unstructured overlay flooding, usually using the route discovery mechanisms of the ad-hoc routing protocol to locate the desired resource in the network.

One of the first documented system is 7DS [48], which attempts to enable P2P resource sharing and information dissemination in mobile environments, been rather a P2P architecture proposal than a practical application.

In [39], ORION aims at providing peer-to-peer services in a MANET, bringing a general purpose distributed lookup service and enhancing file transmission schemes to enable file sharing in MANETs. ORION applies the integration (Fig. 2c) of Gnutella-style [56], flooding into the AODV [49] ad-hoc routing to locate requested files in the network. With ORION, each node in the MANET has a local repository containing the files that the node is sharing. When a node wants to locate a certain file, it issues a query message that is broadcasted through the network. Whenever a node receives such a query message, it sets up the reverse route to the originator - just as AODV does with its route request (RREQ) packets - and retransmits the query message to its physical neighbors. Furthermore, each intermediate node checks its local repository for any files that match the description (e.g., file name, key words, etc.) specified in the query message. If such files are found, the node will send a response message containing the identifiers of all matching files back to the requester using the AODV-style reverse route. Each intermediate node on the response path will also update its file information cache with the file identifiers contained in the response message and the provider (i.e., the sender of the response message). After the requester has received a response, it will then send a data request for the desired files to (one of) the provider(s) using the AODV-style routes discovered during the search. The provider will then divide the requested file into blocks and send data packets containing the various blocks of the requested file back to the requester. The basis of ORION is AODV, and it concentrates only on file sharing applications, providing an application layer routing protocol which causes unnecessary overhead.

The MPP (Mobile Peer-to-Peer) protocol [26] is also proposed as a file sharing system in MANETs. In contrast to ORION, MPP adapts the overlay structure to the physical MANET structure via a cross-layer communication channel (Fig. 2b) between the MANET network layer and the P2P layer. The MPP protocol stack reuses existing network protocols as much as possible. For node-to-node communication, the protocol utilizes an enhanced version of the Dynamic Source Routing (DSR) protocol [60]. More specifically, EDSR (Enhanced Dynamic Source Routing) combines Gnutella-style flooding and DSR ad-hoc routing. For the transportation of user data it uses HTTP over TCP. To connect the application layer protocol with the network layer protocol (EDSR), the Mobile Peer Control Protocol (MPCP) is used. The MPCP is the inter-layer communication channel between the application and the network layer. Using the MPCP, the application can register itself in the EDSR layer to initialize search requests and to process incoming search requests from other nodes. It communicates to the corresponding protocol all incoming and outgoing requests and responses, except the file exchange itself. Besides file sharing applications, MPP also intends to provide location aware services.

In MPP, when a node wants to locate a desired file, it will issue a search request that is flooded throughout the MANET, leveraging the EDSR route discovery process. Whenever a node receives such a search request, it will communicate with its application using the MPP protocol stack to see if the application can provide a matching file. Each intermediate node adds its own node address to the search request to create a DSR-style route and retransmits the search request to its physical neighbors. If the application can provide the requested file, a reply message will be send back to the requester using the reverse path information as contained in the search request. After the requester has received a reply, it will download the desired file from the provider using HTTP. Responses to queries performed by MPP's nodes (and also ORION's nodes), result in a network-wide broadcast of search requests, giving a routing algorithm complexity of $O(n)$ [21], where n is the number of nodes. This is clearly a downside of both approaches as they might not scale to both growing network sizes and increasing request rates.

Hoh et al. proposes in [31] a P2P file sharing system over MANETs based on swarm intelligence, called P2PSI. Basically, it is an hybrid push-and-pull system composed by two processes. In the advertisement process (push), each hotspot[1]

[1] In [31], authors consider a quite large portion of peers to be free-riders, who only retrieve files from others without making contributions to share files. Therefore nodes willing to share files

periodically advertises a *seed* message containing digest information about files to be shared within a limited area (e.g., as determined by the hop count). Every node can independently make the decision on when to advertise and which files to advertise to its neighbors, and such decision can be based on e.g., a ranking system to maximize the number of report delivered [68]. In order to reduce seed message size, Bloom filter technique [3] is applied as a method for summarizing the list of shared files. Upon a node receives a *seed* message, it will cache this information. When been queried, the node that has the cache of the file information will send a reply to the querying node. In the discovery process (pull), the node willing to search for a file, first checks if it has cached the desired file information. If not, the node deploy query messages, forwarded at intermediate nodes based on their pheromone table, to find the identity of the node holding the desired file. The pheromone table records the pheromone intensity on each neighbor link, which denotes the probability of routing a query message via that neighbor based on the number of hops traversed by reply message.

According to [31], the search accuracy of a cross-layered approach, such as P2PSI, is always higher than that of a layered one, as request success ratio decreases at larger network sizes due to increased overhead for the layered approach. In order to avoid such redundancy overhead between P2PSI file discovery and network route discovery process, a cross-layered design (Fig. 2b) is used integrating P2PSI and ARA (Ant-based routing) protocol [4]. The advantage of such design was experimentally observed by implementing P2PSI in the ns-2 simulator and comparing it against two cross-layered design service discovery protocols: *CL_dsr* and *CL_dsdv* [65]. The results show that as the network size and node mobility increase, the request success ratio of the P2PSI outperforms *CL_dsr* and *CL_dsdv*. Indeed the performance of request success ratio of *CL_dsr* deteriorates as it utilizes flooding to search for a file which becomes the performance bottleneck when the network size grows. The same behavior emerges in *CL_dsdv* since it fails to converge as the node mobility increases.

In order to reduce the heavy overhead of always broadcasting search requests in the MANET, zone-based protocols, such as ZP2P (Zone-based P2P by [38]) have been proposed. ZP2P is based on the concept of local zones, determined by a fixed hop-count. When a node is interested in a certain object, it will first check its local cache to see whether any of its zone members can provide the desired object. However, in case the requested object is not available in the node's own zone, it will initiate a *bordercast* of the request through its border nodes, i.e., to those of its zone members that are exactly *k* hops away. In case a border node finds that there are no members in its zone that could provide the requested object, it will continue the *bordercast* by forwarding the request to its own border nodes. This process continues until either a predefined TTL expires or the whole network has been searched.

By introducing the concept of local zones into the P2P search process, some of the network-wide broadcasts may become unnecessary. However, whether or not a requested file can be provided by nodes inside the requester's own zone depends

are called hotspots and they are assumed to provide almost all popular files and some private collections.

entirely upon chance. Especially in larger networks, the cases where a request could be satisfied locally can be expected to be rare [70]. Hence, the utility of local zones will evidently not scale with growing network sizes. The propagation of requests using *bordercasts* can lower the overall traffic as a certain number of inner nodes might not have to forward the requests. Nonetheless, with growing network sizes, the *bordercast* process will quickly encounter the same problems of a regular broadcast as the number of zones that need to be contacted also increases. Furthermore, the efficacy of a *bordercast* depends entirely on factors such as the zone radius and the node density inside the zones. In networks with low or medium node density, it is likely that the routes from the center node of a zone to its border nodes will involve most (if not all) of the inner nodes. Thus, in such networks, the *bordercast* will closely resemble a regular broadcast, and the performance of ZP2P can be expected to be worse than that of a regular broadcast, due to the additional continuous (update) advertisement messages that need to be exchanged. Although not explicitly addressed in [38], nodes need to periodically re-issue their advertisements to take into consideration the effects of node mobility on zone memberships. This will cause ZP2P to generate additional traffic, with respect to a regular broadcast application.

4.3 Structured P2P Networks for MANETs

The concept of DHT was first proposed by Plaxton [51] without the aim to address P2P routing problems. But, it soon proved to be a useful substrate for large distributed systems and a number of projects have been proposed to build Internet-scale facilities leveraging the DHT concept. On the other hand, ad-hoc networks gained great importance due to the increasing occurrence of scenarios which do not have a centralized infrastructure. Whenever there is a need for a scalable data management without any infrastructure, the combination of ad-hoc network and DHT technology seems to be a promising solution [33]. The questions, whether this is beneficial, and how current solutions perform such combination will be discussed in the following sections.

4.3.1 Transparent Layered DHT on Top of Broadcast Based Ad-hoc Routing Protocol

Deploying a DHT directly on top of an existing broadcast based ad-hoc routing protocol does not require any changes to the routing or overlay layer. In that approach, every file name and peer is hashed to a key by standard hash algorithms (e.g., SHA-1 [22]). Every peer should maintain a small routing table of size $O(logn)$, in which each entry directs to an intermediate peer closer to the requested key. The peer closest to the requested key knows the address of the actual peer storing the requested file. In order to route to these intermediate peers, standard MANET routing protocols are deployed which usually acquire topology information using broadcast, increasing the routing algorithm complexity to $O(nlogn)$. As described by [21], this is due to the fact that network routing protocols in MANET introduce complexity of $O(n)$ to find the route between every two peers, although there are only $O(logn)$ peers needed in the P2P overlay.

In order to maintain the correctness of each overlay routing table, peers need to periodically communicate with each other through overlay management protocols. These protocols should be triggered more frequently in MANETs due to mobility and characteristics of the underlying physical networks. Otherwise, routing information at the overlay might not be consistent. In [12], the performance of Bamboo is evaluated in a static multi-hop environment common to ad-hoc networks. When deploying Bamboo over MANET following a layered approach (Fig. 2a), the overlay network forms a virtual network in the application layer while the underlying network is transparently managed by MANET routing protocols such as AODV.

Bamboo uses proactive management traffic in order to maintain the network structure. Neighbor ping is generated by every node in order to make sure that the node can still reach its one-hop neighbors in the overlay, and it is also used to maintain a RTT estimation for retransmission timeout calculations. Nodes also perform leafset update by periodically choosing a random node from its leafset, and execute a leafset push followed by a leafset pull. Bamboo considers that two nodes share the same level when one node contains the other node in its routing table. Therefore, the local routing table update is used to exchange the node information in that level. Data storage updates are also performed in order to maintain the desired number of replicas among the peers.

However it is expected that the proactive management maintenance introduced by Bamboo increases network traffic, and consequently as the network grows, high congestion will be experienced. In order to find a balance between management traffic in the overlay and network congestion, three different configurations for Bamboo management traffic were compared in [12]; “no” management, “standard” management (used by [55]), and “custom” management. Table 1 presents the parameters used by each configuration. The comparison carried tries to find a balance between lookup efficiency and management traffic overhead. Too frequent management traffic will lead to high overhead in multi-hop environments and thus lead to network congestion. No management, on the other hand, will leads to low lookup efficiency.

Simulations were performed using ns-2 over different scenarios, where the nodes were positioned on a grid at a distance of 200m, with 250m of transmission range and 500m of carrier sense range using two ray ground as radio propagation model. The transmission rate is set to 11Mbps, and the basic rate to 1Mbps. The AODV-UU routing protocol was adopted using default settings proposed by [66]. Simulations were performed for 60 seconds without bootstrapping period. During the experiments, every 2 seconds, each node generates a 500-byte PUT message with a random key to store data in the overlay. All nodes also try to acquire random selected keys that are located on other nodes generating a 32-byte GET message every 2 seconds.

Figure 3a presents the total Bamboo management overhead, which represents the aggregation of the overlay management traffic including: neighbor ping, leafset update, routing table update, and data storage update, for the three different scenarios.

Table 1 Bamboo management timers (secs)

	NO	Standard	Custom
Leafset update	–	1	5
Local routing update	–	5	10
Global routing update	–	10	20
Data storage update	–	2	6
Neighbor ping	0,5	0,5	0,5

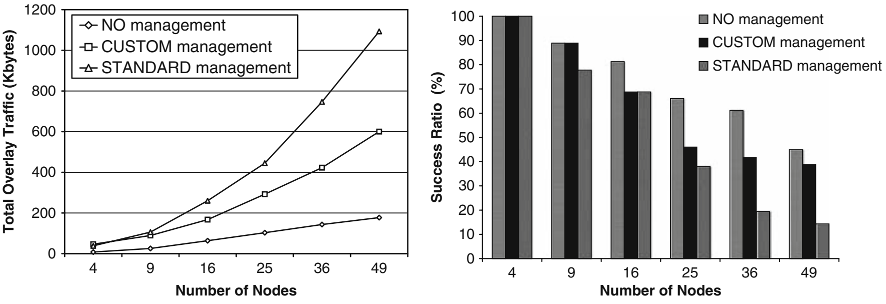
As expected, the overhead introduced by Bamboo increases with number of nodes, and is much higher for "standard" timeout settings compared to "no", and "custom" management. This is mainly due to the aggressiveness of periodic updates required by Bamboo to monitor the status of other nodes in the overlay and update the overlay data structures. On the other hand, in the case of "no" management, each node does not generate periodic updates, but neighbor ping is still performed in order to maintain the leafset peers.

Figure 3b also illustrate the success rate behavior of Bamboo over the scenarios chosen by [12]. As the number of nodes increases, network load increases and success ratio decreases accordingly as illustrated in Fig. 3b. For example, in the 36 nodes grid, the success ratio is 61, 41 and 19%, respectively for "no", "custom" and "standard" management. The lower success ratio for higher number of nodes can be explained by the higher percentage of management and routing overhead in order to maintain the overlay structure, as shown in Fig. 3a. The ability to find the destination nodes which are responsible for the specific keys degrades as management overhead increases network contention. This results in higher number of resent and dropped packets over the wireless links due to network congestion and consequently problems in the routing layer, as shown in more details over the simulation results presented in [12].

Other related publications, such as [17] which deploys Chord over MANET routing protocols, also indicate that simply deploying a standard MANET routing layer does not scale with increasing number of clients, network size, and mobility. The reasons are manifold such as the characteristics of multi-hop communication, the consistency problem between the two routing layers, and the design assumptions for MANET routing protocols which assume traffic characteristics unlike those of structured overlay protocols.

4.3.2 Integrating DHT over the Network Layer

As illustrated in Section 4.3.1, the characteristics of the underlying ad-hoc network protocol has great effect on the performance of the overlay as the DHT induces a constant flow of control and query messages. An optimized interaction between ad-hoc network and DHT is essential to create an efficient combination. There are



(a) Managment Overhead

(b) Success Rate

Fig. 3 Impact of bamboo management traffic

several approaches proposed in the literature that try to exploit similarities between ad-hoc network and DHT in order to integrate them in a system with higher performance, by also reducing the overheads. The examined approaches analyzed here are VRR [8], SSR [24], CrossROAD [18], MADPastry [70], MeshChord [7], and Hashline [61].

Virtual Ring Routing (VRR) [8], proposed by Microsoft Research Centre, is a networking routing protocol which pushes peer-to-peer concepts to the network layer itself. Caesar et al. argue that VRR brings benefits when implemented over MANETs, as it balances the load of managing hash-table keys across nodes, and avoids flooding of routing messages through the network. Based on Pastry [58], VRR organizes the nodes into a virtual ring ordered by their identifiers. Each node maintains a small number of routing paths to its neighbors in the ring.

In VRR, node identifier are fixed, unique and location independent. To maintain the integrity of the virtual ring with node and link failures, each node maintains a virtual set (vset) of cardinality r (predecessor and successor nodes). The routing path between a node and each of its virtual neighbors is called vset-path. The routing table also maintains the physical neighbor set (pset) with the identifiers of the nodes that it can directly communicate with at the link layer. Such information is gathered through broadcast of hello messages periodically. The routing information for a vset-path is also stored on the nodes along the paths. Then, a node maintains a routing table with information about the vset-paths to its virtual neighbors, other vset-paths that are routed through the node, and the pset of physical neighbors. As described in [8], VRR requires $rp + k$ routing table entries per node on the average, where p is the average path length, and k is the number of physical neighbors. Since node identifiers are random[2] and location independent the virtual neighbors of a node will be randomly distributed across the physical network. So, the probability that a random node has a path to a random destination is $O(rp/n)$. Therefore, a

[2] VRR hashes the node current IP address in order to obtain the node identifiers.

packet is expected to reach a node that has a vset-path to the destination after visiting $O(n/(rp))$ nodes.

Unlike routing protocols that forward packets based on destination address, VRR nodes route packets to destination identifiers (keys) by forwarding them to the next hop towards the path endpoints whose identifier is numerically closest to the destination identifier from among all the endpoints in their routing table. An advantage of such scheme is that these keys can identify application objects instead of just VRR nodes. Control messages to set up new vset-paths are routed using existing vset-paths avoiding the flooding on the network.

Scalable Source Routing protocol (SSR) [24] brings the same concept of VRR while trying to integrate the P2P overlay into the network layer. But while VRR does not assume any specific MANET routing protocol integration, SRR combines the Dynamic Source Routing protocol (DSR) [35] in the physical network with Chord routing in the virtual ring formed by the address space. Fuhrmann states that SSR trades off shortest path for a reduced amount of state information, leading to less maintenance overhead. Therefore, besides the successor, SSR's nodes store the addresses of $O(logn)$ additional nodes at exponentially spaced distances to reduce the average request path length from $O(n)$ to $O(logn)$, where n is the number of nodes in the network.

Following the DSR concept, data packets of SSR contain a source address, a destination address and a source route. However according to SSR design, the source route does not have to span the entire path from the source to destination. When the virtual ring has been established, SSR can route messages to any destination. By constructing the route cache, each node contains source routes to the node's neighbors in the virtual ring. Beside that, the caches will contain source routes to other destinations also. For example, all nodes that are part of a source route in the cache can be viewed as potential destinations. When routing a packet, the respective node chooses the (intermediate) destination from its cache that is physically closest to itself and virtually closest to the final destination of the packet. It appends the source route from its cache to the packet's header. The nodes along this source route can then forward the packet using the source route in the packet. This routing step is repeated at all intermediate nodes and all subsequent destinations until the packet has reached its final destination. If the virtual ring has been formed consistently, this routing algorithm is guaranteed to succeed for any source and destination pair.

To maintain the virtual ring consistency in SSR, all nodes must have valid source routes to their respective virtual neighbors; e.g., its predecessor and successor in the address ring. The nodes need also to have information about their physical neighborhood, information which is gathered through a periodic beacon message (e.g., hello message). The state maintenance of the virtual ring continues until all nodes have mutually correct virtual neighbors, in order to guarantee network convergence. In order to reduce the routing stretch, SSR's nodes use the source routes in their routing caches to prune unnecessarily long source routes, e.g., routes contain cycles or a shorter sub-path to one of the nodes in the source route is known (short cut). However, as discussed by [70] the effectiveness of this source route pruning entirely depends upon the available cache entries and there are no guarantees as to how well the source routes in the system can be pruned.

CrossROAD is proposed by [18] as a way to reduce communication overhead introduced by Pastry when deployed over mobile ad-hoc networks. Different from VRR and SSR integrated approaches, a cross-layered architecture defining interactions between P2P and routing layers allows CrossROAD to exploit additional information to optimize the overlay management. These interactions are handled by the Network Status module (NeSt) [16], an external data sharing module, which provides interfaces for cross-layer interactions throughout the protocol stack. Each node running CrossROAD piggybacks advertisements of its presence in the overlay into routing messages periodically sent by OLSR. Thus, each node in the network becomes aware of the other peers in the overlay network. Then, each node in the overlay maintains a routing table of size $O(n)$. Since each node knows all nodes taking part in the overlay, the sender of a specified message can directly identify the closest destination for the selected key, and subsequently use the OLSR protocol at the network layer to deliver the message through the shortest path ($O(1)$ virtual hops in the overlay).

Reference [18] states that such mechanism reduces the overhead required to build and maintain DHTs in legacy systems such as Pastry, however at the cost of additional overhead in the OLSR layer. However, no remote connections are required by CrossROAD to initialize the overlay routing table, neither in case of disconnection events or network partitioning. It directly exploits the network routing protocol that collects topology changes periodically sending its LSU (Link State Update) packets, and directly updates its own routing table and the related abstraction in the NeSt. In this way CrossROAD becomes aware of topology changes with the same delays of the routing protocol. Nevertheless, it is worth to mention the lack of results regarding scalability of CrossROAD to both growing network sizes and node mobility.

In order to take physical location into consideration, MADPastry is proposed by [70]. MADPastry integrates (Fig. 2c) the application layer Pastry and the reactive ad-hoc routing protocol AODV. The concept of random landmark [67] is used to create physical clusters where nodes share a common overlay ID prefix. Since there are generally no stationary nodes available in MANETs, MADPastry works without any fixed landmark nodes. Instead, it uses a set of landmark keys, which are simply overlay IDs that divide the overlay ID space into equal-sized segments. Nodes associate themselves with the temporary landmark node that is currently closest to them (e.g., as determined by the hop count) by adopting its overlay ID prefix. For that purpose, temporary landmark nodes send out beacons periodically. These beacons are broadcast and whenever a node overhears a landmark beacon, it stores the current landmark node's ID and the distance to it as given by the hop count of the beacon. As broadcast imposes serious network burden, landmark beacons are only propagated within the landmark's own cluster, i.e., beacons are only forwarded by nodes belonging to that cluster.

When a MADPastry's node intends to advertise a resource, it will now insert the resource descriptor under two different keys. The first key is the regular hash key (of the resource's URI, etc.) inserted into the network. To obtain the second key under which the resource descriptor is stored, the regular resource key is altered to make sure the descriptor will be stored in the resource host's own MADPastry cluster. For this purpose, the resource key's prefix is replaced with the host's own cluster prefix (current landmark node's ID). Hence, intra-cluster communication can be expected to travel only short physical paths, as lookups process will try to find the corresponding resource descriptor in its physical vicinity (local cluster members). However such optimization might be useful for popular files or standard services that are hosted by multiple nodes. Only if this local lookup provides no (appropriate) answer, will the request be forwarded as in a regular network-wide lookup. Following this process, the first key remains fixed during the lifetime of a node, while the second one can change depending on the node's position in the physical network.

To be able to route packets along the network, MADPastry nodes maintain three different routing tables: a standard AODV routing table for physical routes from a node to specific target nodes, as well as a *stripped down* Pastry routing table and a standard leafset for indirect routing. Differently from Pastry routing table which consist of $log_{2^b}N$ rows, the *stripped down* Pastry routing table only needs to contain $log_{2^b}K$ rows, with K being the number of landmark keys. Using such approach, MADPastry avoids the expensive Pastry routing table maintenance overhead, but it deliberately sacrifices the $O(logn)$ bound on the number of overlay hops during a key lookup. MADPastry also perform a proactive routing table maintenance, by periodic pinging its "left" and "right" leaf. According to Zahn, this is necessary to guarantee overlay routing convergence. The remaining routing entries are gained by overhearing data packets. Then, the accuracy of the Pastry routing tables and leafsets largely depend on the number of packets that a MADPastry node receives or overhears. With the idea of proximity awareness using random landmarking, physical clusters of nodes sharing a common overlay ID prefix are created, avoiding longer overlay hops per lookup.

MeshChord, proposed by [7], is an specialization of Chord applied to wireless mesh networks, where the availability of a wireless infrastructure, and the 1-hop broadcast nature of wireless communication are taken into account while performing key lookup. In MeshChord, routers are assumed to be stationary, but they can be switched on/off during network lifetime. If a client in the mesh network wants to find a certain resource, it sends a key lookup message to its reference mesh router (a mesh router within its transmission range). The reference router forwards the resource request in the DHT overlay according to the rules specified by the Chord protocol, until the resource query can be answered. As in Chord, in a n-node system, each MeshChord's node maintains information about only $O(logn)$ other nodes, and resolves lookups via $O(logn)$ messages to other nodes.

MeshChord explores location awareness by assigning IDs to peers according to their coordinates, accomplished by, for example, the use of GPS receivers. Besides that, MeshChord also takes advantages of 1-hop broadcast communication by overhearing lookup request packets in order to speed up lookup operation. Then, by overhearing a lookup request at the MAC layer, a node can reply to it if the requested ID is comprised between its ID and the ID of its predecessor in the unit ring.

It is worth observing in [7] that location awareness tends to decrease the lookup operations under dynamic network conditions. In fact, location-aware ID assignment tends to rule out the possibility of having close-by peers in the physical network which are far-away in the overlay (e.g., in Chord, possibly corresponding to the last fingers in the finger table). However, MeshChord achieves a considerable reduction in message overhead, and improvement in query response time while utilizing location awareness and overhearing strategies.

Hashline [61], a DHT-based file sharing system for wireless ad-hoc networks, also integrates the P2P query functionality with the network routing. Hashline is able to answer location queries and also discover and maintain routing information that is used to transfer files from a source peer to another peer. In this way, it enables the proposed P2P file sharing system to run on an ad-hoc collection of wireless nodes without requiring a separate MANET routing protocol at the network layer.

The basic idea in Hashline is the adaptation of the CAN [54] P2P routing protocol. Unlike CAN, however, [61] uses a one-dimensional space, called *hashline*, into which keys and node IDs are mapped. The hashline is divided hierarchically into segments so that each node is responsible from one segment. The values (location information) of the keys falling into a segment are stored in the corresponding node responsible for that segment. The relationship between segments can be considered as a tree consisting of parents and children, so that the hashline segment of a parent spans the hashline segments of all its children.

In [61], when a node would like to find the location of a file with key *k*, the node forwards the query to one of its children if *k* falls into the hashline segment of one of the children. Otherwise the query is forwarded to the parent. Hence a tree based routing is used. At the end, the node that is responsible for the hashline segment including the key receives the query. That node knows the location of the file and also the route to that location. It answers the query together with the location and route information. The requester can then download the file from that location using the learned route. Hence the download operation does not require a different routing protocol to find the route to the location where the file is stored. In this way, queries and downloaded files are efficiently routed in the network. However, the operations performed to keep the tree-based routing state up-to-date when a node leaves or joins are quite costly. Hence the proposed protocol is suitable for low mobility wireless networks. As described by [61], the number of routing table entries mantained by each Hashline's node is at most *k*, where *k* is the number of of physical neighbors.

4.4 Summary and Comparison of the Solutions

As seen, a number of different approaches exist that could potentially be used as building blocks for large scale distributed network applications in multi-hop networks, such as MANETs or Mesh Networks. The varying characteristics of the presented approaches sometimes make it difficult to compare them directly against each other. Therefore, Table 2 intends to assess the different approaches according to:

- Fusion with Underlay: integration principle between P2P and MANET interactions;
- P2P overlay protocol: inspired P2P protocol;
- Routing Algorithm: routing algorithm deployed at the network layer;
- Overlay Adaptation: overlay topology reaction to network change;
- Periodic management: periodic management information exchanged among peer nodes at the overlay layer ;
- Location Awareness: use of location information to construct the overlay;
- Proposed Applicability: proposed applicability and use cases considered;
- Prototype Implementation: prototype implementation availability.

It is interesting to analyze that all unstructured approaches utilize a Gnutella-like protocol. Structured approaches are mainly based on Chord and Pastry (as Bamboo is inspired by Pastry). Regarding routing algorithms, most approaches studied rely on reactive routing protocols, such as AODV, DSR, and ARA. Proactive routing algorithms, such as CrossROAD, appear to be very expensive in terms of resource usage and routing table maintenance traffic injected into the network.

The cross-layered or integrated design (Fig. 2b, c) of unstructured P2P overlays and ad-hoc routing is an intuitive and simple solution for the discovery of objects in MANETs. It is a straightforward approach as the changes and enhancements to the underlying ad-hoc routing protocols are minimal since, for example, reactive MANET routing protocols already have the capability of broadcasting requests and directly replying to the requester. However, first and foremost, the obvious disadvantage of such approaches is their poor scalability when network size grows. The main reason is that network-wide broadcast of search requests scales to neither growing network sizes nor increasing request rates. P2PSI and ZP2P try to scale to large MANETs under mobility by applying ant colony behavior and zone-based broadcasting, respectively.

Despite Bamboo/AODV, the DHT-based protocols avoid duplicated overhead through integration or cross-layering design. They also try to avoid broadcasting whenever possible, and optimize their DHT entries by overhearing packets. A significant difference among these systems is the use of location aware information by MADPastry and MeshChord, compared to the other DHT-based protocols. MADPastry exploits the concept of random landmarking to create overlay clusters, while MeshChord assumes that nodes are stationary, have their own position information available, and uses MAC layer overhearing to reduce search latency. Furthermore, since reply and file transfer messages are unicasted for all unstructured and structured approaches, their reliability depends entirely on the scalability and performance of the chosen (reactive or proactive) ad-hoc routing protocol.

Table 2 Assessment of related approaches

Solutions	Fusion with underlay	P2P Overlay protocol	Routing algorithm	Overlay adaptation	Periodic management	Location awareness	Proposed applicability	Prototype implementation
ORION	Integrated	Gnutella-like	AODV	N/A	N/A (Not Applied)	No	File sharing	No
MPP	Cross-layered	Gnutella-like	DSR	N/A	N/A	No	File sharing and location aware services	no
P2PSI	Cross-layered	Gnutella-like	ARA	N/A	N/A	No	File sharing	No
ZP2P	Cross-layered	Gnutella-like	AODV	N/A	N/A	Y (via local zones)	File sharing	No (SDL specification)
Bamboo/AODV	Layered	Bamboo	AODV	Proactive	Leafset and routing table	No	File sharing, decentralized name service, P2PSIP	Yes (Linux based)
VRR	Integrated	Pastry	AODV	Reactive	Leafset table	No	File sharing, decentralized name service, P2PSIP	Yes (Windows based)

Table 2 (Continued)

Solutions	Fusion with underlay	P2P Overlay protocol	Routing algorithm	Overlay adaptation	Periodic management	Location awareness	Proposed applicability	Prototype implementation
SSR	Integrated	Chord	DSR	Reactive	Successor and predecessor	No	File sharing, decentralized name service, P2PSIP	Yes (Linux based)
CrossROAD	Cross-layered	Pastry	OSLR	Proactive	OLSR topology information	No	Multicast-based, white board applications	No
MADPastry	Integrated	Pastry	AODV	Reactive	Leafset table	Y (via cluster formation)	File sharing, decentralized name service	No
MeshChord	Cross-layered	Chord	DSR	Reactive	Predecessor and finger table	Y (via coordination information)	File/Resource sharing	No
Hashline	Integrated	CAN	Custom	Reactive	N/A	Y (via tree formation)	File/Resource sharing	No

5 P2P Application Scenarios for Mobile Ad-hoc Networks

The P2P solutions presented in the previous Section provide ways to deploy efficient distributed resources in MANETs using flooding, or key-based routing. These solutions are important building blocks to realize P2P applications in MANETs. In this section, we detail their use in important applications and services such as decentralized name service, overlay-based multicast, and multimedia services.

5.1 Decentralized Name Service

Nearly all Internet applications use persistent, human-readable names for users, hosts, and services. In the current Internet, this is done using the the Domain Name System (DNS), which is a centralized, distributed system with a single root of trust.

In peer-to-peer systems such as P2PSIP [20], it is useful to have human-readable, user-friendly names, but a centralized naming service is an undesirable choke point. It is difficult to implement a centralized service in a MANET, therefore it is interesting to decentralize service using P2P concepts.

As an example, MAPNaS, a decentralized name service for MANETs, is proposed by [71] in order to identify a resource (e.g., a file, a service, etc) by a unique resource key that is mapped into the logical DHT space. Due to the lack of a fixed network topology, there are no dedicated resource directory servers. Instead, every node functions both as a resource host (e.g., of its own files and services) and as a resource directory for certain remote resources.

While mobile devices often have limited hardware and maybe storage capabilities, the design goal of MAPNaS is to keep the architecture simple, where nodes store the resource descriptors (the resource key along with the specific network address of the resource) they are responsible for in their local MAPNaS repository. Furthermore, every node advertises which resources it is willing to share through MAPNaS. When a node in the network wants to make a local resource (e.g., a service, a file, etc.) available to other nodes in the network, it assigns a hash key to that resource, e.g., by hashing the resource's URI. Using that key, the node will then construct a resource descriptor consisting of the resource key and the physical network address (e.g., IP address) of the resource provider (in this case, the node address). Using the DHT, the descriptor is routed to the node currently responsible for the resource key. That recipient node will then store the resource descriptor in its local repository.

Resource discovery with MAPNaS works similarly to the resource advertisement process. First a lookup request is sent to the node currently responsible for the hash key of the resource's identifier. Then, the eventual destination node will check its local repository and send back the matching resource descriptor (or multiple descriptors in case several nodes are hosting the same resource). As the DHT in MAPNaS is realized through MADPastry [72], location replications of resource descriptors are restricted to MADPastry's clusters.

In traditional SIP networks the main task of a SIP server is to resolve an Address of Record (AoR) to the current IP address (Contact URI) of a user. This name resolution usually depends on Domain Name Server (DNS). P2PNS [2] presents a distributed name service using DHT to resolve AoRs to Contact URIs without relying on DNS and central SIP servers. Apart from this decentralized name resolution the call setup is based on the standard SIP protocol. In P2PNS there is a separation between the overlay layer (key-based routing), the data storage layer (distributed hash table), the name resolution layer (P2PNS Cache) and the protocols, that utilize the name service (like SIP or DNS). Hence, the specification of the key-based routing protocol is independent of P2PNS, and key-based routing solutions discussed earlier could be easily applied in the MANET environment.

The P2PNS architecture comprises a name resolution and caching layer (P2PNS Cache) on top of an overlay which provides key-based routing and DHT services. In P2PNS, a two-stage name resolution mechanism is proposed to efficiently handle frequent IP address changes. A user chooses an arbitrary name as AoR (e.g., name@p2pname.org). Then a mapping from the selected AoR to the corresponding nodeID [3] is stored in the DHT. In this case the name resolution layer first queries the DHT for the nodeID (given the user's AoR) of the destination node and in a second step resolves this nodeID to the current IP address of nodes.

5.2 Overlay-Based Multicast

Overlay-based multicast is one option to implement multicast at the P2P layer. Usually, multicast protocols are classified as operating at the network layer, like routing protocols, or at the application layer, where "application" denotes all possible layers above the transport. Overlay-based multicast runs only at nodes involved in the related application, and it just requires standard unicast support from the routing level. There are basically two approaches: (1) structured approach and (2) unstructured approach. In the structured approach, a multicast routing structure, like a tree, is established at the overlay level. Hence parent-child relationships are defined between peers making up the tree and the packets are forwarded over these peers towards the receiver peers which are also part of the overlay tree. In the unstructured approach, no such structure is established and used. Instead the sender has to know which receivers are interested in the packets and sends them to each receiver using a different mechanism, such as unicasting the same packet to each receiver. This requires the sender to know the potential receivers of the multicast data, which can be achieved through a multicast group membership protocol.

Applying existing P2P multicasting solutions developed for wired and infrastructure-based networks to MANETs will not work efficiently due to various reasons

[3] Every peer chooses once a 160 bits nodeID for joining the overlay. This nodeID is retained even if the peer changes its IP address or leaves the overlay from time to time. The DHT allows to resolve the nodeID to the current IP address of a peer.

discussed before. Therefore, existing solutions must be adapted or new solutions must be developed.

XScribe [19] is an structured P2P multicasting protocol for ad-hoc networks. It is based on the well known P2P multicasting protocol Scribe [11], which was developed for wired P2P networks. XScribe can be used to implement various multicasting services and works together with CrossROAD in order to obtain the network topology.

In XScribe, the sender is required to know receivers of a multicast group using a membership management protocol. The sender obtains this knowledge using a cross-layer approach, where each multicast receiver sends its bitmask (indicating which groups it is interested in) embedded into the CrossROAD routing packets. When the sender has a packet to sent to a group/topic (hence to the receivers that are interested in that topic), the sender directly sends the packet to each receiver using the CrossROAD DHT overlay. Therefore, the packet is unicasted to each receiver without the need to setup a tree or any other multicasting structure before sending data.

Even though this seems to be inefficient, simulation results in ad-hoc networks show that XScribe performs better compared to deploying the original Scribe protocol over MANETs with standard routing due to the reduced routing stretch between peer nodes in the overlay structure.

5.3 Multimedia Services

In P2P file sharing applications, the main concern is to locate files to a given query. Once located, the user can decide to download the file, which then is downloaded out-of-band (i.e., not through the P2P overlay itself, but through the underlying networking and transport mechanisms). Hence for file and resource sharing P2P applications, data transport is not the main concern.

For P2P multimedia services, however, the situation is different. For non-realtime media, the media is typically located, downloaded and then played back from the local disk, in contrast media streaming provides faster response time at less client storage. Media streaming, however, requires a different type service provisioning and transport from the underlying network. Certain amount of network resource such as high bandwidth and controlled delay. To guaranteed smooth delivery admission control [9] needs to be implemented in order to provide real time streaming, which requires also tight control and end-to-end delay.

Providing P2P media services over ad-hoc networks is challenging due to the characteristics of multi-hop forwarding and the wireless medium (see Section 2.2. On the one hand, if some peers become hot-spots as media uploaders, the upload capacity of peers may be much more restrictive than the upload capacity of media servers located on the Internet; as thse peers are usually connected via bandwidth-constrained wireless links. On the other hand, if the load is evenly distributed among peers, serving the media content from lots of peers provides scalability and can increase system throughput. Another issue is that the connection between an uploading peer and a downloading peer may not be stable during the duration of the streaming session, due to node mobility or peer disconnections [27]. Additionally the download path that is going over multiple peers may cause additional delay and increased jitter.

P2P streaming can utilize multiple peers as the sources of the same media file. As a result, if there are N such source peers, then each one will require R/N upload capacity where R is the streaming rate. Additionally, a peer that has downloaded the content may start serving the content to other peers, in this way increasing the number of serving peers.

The characteristics of wireless multi-hop networks require modifications of existing P2P media applications to run efficiently. For example, in [44], the authors propose a new set of criterias and methods to select super-peers in a P2P network providing IP telephony service. For ad-hoc networks, the selection criteria depend not only on the CPU, memory and storage capabilities of candidate super-peers, but also on the location of super-peers, their accessibility and their distance to other super-peers.

6 Summary

In this chapter, we have investigated the opportunities and challenges of the application of peer-to-peer concepts to mobile ad-hoc networks. An overview of P2P overlay networks shows that unstructured P2P systems do not impose a rigid relation between the overlay topology and resource locations, representing an easy implementation for dynamic environments such as MANETs. DHTs impose a structure on the overlay topology by satisfying certain criteria depending on the respective DHTs. An overview of mobile ad-hoc networks characteristics shows that mobile ad-hoc networks impose several problems in terms of wireless multi-hop characteristics leading to high and varying packet loss and delay, caused by collisions and interference among nodes. Future challenges such as disrupted communications, and intermittent connectivity in these scenarios are also envisioned. Most of the relevant schemes of MANET routing protocols are also briefly presented, giving focus on flat routing approaches such as topology-based, geographic-based, and probabilistic routing.

Although there is an inherent similarity, common peer-to-peer systems must be modified in many ways to enable their use in ad-hoc networks. Several approaches improve the performance of unstructured and structured P2P communication in wireless multi-hop networks. Meanwhile, different principles, such as layered, integrated and cross-layered design, guide to different integration and interaction possibilities between the peer-to-peer layer and the network layer. According to the simulation results, the deployment of a P2P protocol as is on top of ad-hoc routing layer cause significant message overhead and redundancy in communication. Thus, the integrated and cross-layered designs for unstructured P2P are shown to be intuitive and simple as modifications to the ad-hoc routing protocols are minimal. However, the network-wide broadcast of ad-hoc routing due to search requests (reactive) or topology change (proactive) does not scale to neither growing network sizes nor network mobility. In order to overcome that, some proposals push the DHT concept to the ad-hoc routing layer, enabling key-based routing for MANETs. Moreover, some of them explicitly considers physical locality in order to construct the overlay, while trying to keep minimum overhead.

As peer-to-peer applications gain greater importance in the infrastructure Internet, efficient porting of such applications to wireless scenarios is also discussed. Therefore, the solutions presented in Session V pave the way to the deployment of distributed applications such as decentralized name service, overlay-based multicast, multimedia service, and several other possibilities.

Acknowledgement

This work was supported by the European Commission in the framework of the FP7 Network of Excellence in Wireless COMmunications NEWCOM++ (contract n. 216715)

References

1. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Computing Surveys **36**(4), 335–371 (2004)
2. Baumgart, I.: P2pns: A secure distributed name service for p2psip. In: Proc. of Mobile P2P (2008)
3. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (1970). DOI 10.1145/362686.362692. URL `http://portal.acm.org/citation.cfm?id=362692`
4. Bouazizi, I.: Ara - the ant-colony based routing algorithm for manets. In: Proc. of ICPPW. IEEE Computer Society, Washington, DC, USA (2002). URL `http://portal.acm.org/citation.cfm?id=848286`
5. Burgess, J., Gallagher, B., Jensen, D., Levine, B.: MaxProp: Routing for vehicle-based disruption-tolerant networks. In: Proc. of INFOCOM. Vancouver, Canada (2006)
6. Burleigh, S., Hooke, A., Torgerson, L., Fall, K., Cerf, V., Durst, B., Scott, K., Weiss, H.: Delay-tolerant networking: an approach to interplanetary internet. IEEE Communications Magazine **41**(6), 128–136 (2003). DOI http://dx.doi.org/10.1109/MCOM.2003.1204759. URL `http://dx.doi.org/10.1109/MCOM.2003.1204759`
7. Burresi, S., Canali, C., Renda, M.E., Santi, P.: Meshchord: A location-aware, cross-layer specialization of chord for wireless mesh networks. In: Proc. of PerCom. Hong Kong (2008)
8. Caesar, M., Castro, M., Nightingale, E.B., OShea, G.: Virtual ring routing: Network routing inspired by dhts. In: Proc. of ACM SIGCOMM. Pisa, Italy (2006)
9. Calafate, C., Cano, J.C., Manzoni, P., Malumbres, M.: A qos architecture for manets supporting real-time peer-to-peer multimedia applications. In: ISM (2005)
10. Cao, J., Zhang, Y., Cao, G., Xie, L.: Data consistency for cooperative caching in mobile environments. IEEE Computer **37**, 60–66 (2007)

11. Castro, M., Jones, M.B., Kermarrec, A.M., Rowstron, A., Theimer, M., Wang, H., Wolman, A.: An evaluation of scalable application level multicast built using peer-to-peer overlays. In: Proc. of INFOCOM (2003)
12. Castro, M.C., Villanueva, E., Ruiz, I., Sargento, S., Kassler, A.J.: Performance evaluation of structured p2p over wireless multi-hop networks. In: Proc. of MESH (2008)
13. Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., Weiss, H.: Delay-tolerant networking architecture. `http://www.ietf.org/rfc/rfc4838.txt`. IETF RFC 4838
14. Clausen, T., Jacquet, P., Laouiti, A., Muhlethaler, P., Qayyum, A., Viennot, L.: Optimized link state routing protocol. In: Proc. of the IEEE INMIC. Pakistan, USA (2001)
15. Conti, M., Maselli, G., Turi, G., Giordano, S.: Cross-layering in mobile ad hoc network design. Computer **37**(2), 48–51 (2004)
16. Conti, M., Maselli, G., Turi, G., Giordano, S.: Cross layering in mobile ad hoc network design. In: IEEE Computer (2004)
17. Cramer, C., Fuhrmann, T.: Performance evaluation of chord in mobile ad hoc networks. In: Proc. of MobiShare (2006)
18. Delmastro, F.: From pastry to crossroad: Cross-layer ring overlay for ad hoc networks. In: Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on, pp. 60–64 (2005). URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1392801`
19. Delmastro, F., Passarella, A., Conti, M.: P2p multicast for pervasive ad hoc networks. Pervasive and Mobile Computing **4**(1), 62–91 (2007). DOI http://dx.doi.org/10.1016/j.pmcj.2007.03.001
20. Dhara, K.K., Krishnaswamy, V., Baset, S.: Dynamic peer-to-peer overlays for voice systems. In: Proc. of IEEE International Conference on Pervasive Computing and Communications Workshops (2006)
21. Ding, G., Bhargava, B.: Peer-to-peer file-sharing over mobile ad hoc networks. In: Proc. of PERCOMW (2004)
22. Eastlake, D.E., Jones, P.E.: Us secure hash algorithm 1 (sha1). `http://www.ietf.org/rfc/rfc3174.txt`. IETF RFC 3174
23. Fiore, M., Casetti, C., Chiasserini, C.F.: Efficient retrieval of user contents in manets. In: Proc. of INFOCOM. Anchorage, AK, USA (2007)
24. Fuhrmann, T., Di, P., Kutzner, K., Cramer, C.: Pushing chord into the underlay: Scalable routing for hybrid manets. Tech. rep., Universität Karlsruhe (TH) (2006)
25. Gerla, M., Lindemann, C., Rowstron, A.: P2p manet's - new research issues. In: Perspectives Workshop: Peer-to-Peer Mobile Ad Hoc Networks - New Research Issues (2005)
26. Gruber, I., Schollmeier, R., Kellerer, W.: Performance evaluation of the mobile peer-to-peer protocol. In: Proc. of GP2PC (2004)
27. Guangtao, X., Li, M.L., Deng, Q.N., You, J.Y.: Stable group model in mobile peer-to-peer media streaming system. In: First IEEE International Conference on Mobile Ad Hoc and Sensor Systems (2004)
28. Hara, T.: Effective replica allocation in ad hoc networks for improving data accessibility. In: Proc. of INFOCOM, pp. 1568–1576 (2001)
29. Hara, T.: Replica allocation methods in ad hoc networks with data update. Mobile Networks and Applications **8**(4) (2003)
30. Heckmann, O., Bock, A.: The edonkey2000 protocol. Tech. rep., Darmstadt University of Technology (2002)
31. Hoh, C., Hwang, R.: P2p file sharing system over manet based on swarm intelligence: A cross-layer design. In: Proc. of WCNC (2007)
32. Holland, G., Vaidya, N.H.: Analysis of tcp performance over mobile ad hoc networks. In: Proc. of MobiCom, pp. 219–230 (1999)
33. Hu, Y.C., Das, S.M., Pucha, H.: Peer-to-peer overlay abstractions in manets. In: J. Wu (ed.) Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks, pp. 858–871. CRC Press (2005)

34. Johnson, D.B.: Routing in ad hoc networks of mobile hosts. In: Proc. of the IEEE Workshop on Mobile Computing Systems and Applications, pp. 158–163. Santa Cruz, USA (1994)
35. Johnson, D.B., Maltz, D.A.: Dynamic source routing in ad hoc wireless networks. In: Imielinski, Korth (eds.) Mobile Computing, vol. 353. Kluwer Academic Publishers (1996). URL citeseer.ist.psu.edu/johnson96dynamic.html
36. Karakaya, M., Korpeoglu, I., Ulusoy, O.: Free riding in peer-to-peer networks. In: IEEE Internet Computing, to appear, 2009 (2009)
37. Karp, B.N., Kung, H.T.: Gpsr: Greedy perimeter stateless routing for wireless networks. In: Proc. of MobiCom, pp. 243–254 (2000)
38. Kellerer, W., Schollmeier, R.: Proactive search routing for mobile peer-to-peer networks: Zone-based p2p. In: Proc. of ASWN (2005)
39. Klemm, A., Lindemann, C., Waldhorst, O.P.: A special-purpose peer-to-peer file sharing system for mobile ad hoc networks. In: Proc. of IEEE VTC (2003)
40. Kranakis, E., Singh, H., Urrutia, J.: Compass routing on geometric networks. In: Proc. of CCCG, pp. 51–54. Vancouver, Canada (1999)
41. Liang, J., Kumar, R., Ross, K.: Understanding kazaa. http://citeseer.ist.psu.edu/liang04understanding.html (2004)
42. Lindgren, A., Doria, A., Schelen, O.: Probabilistic routing in intermittently connected networks. ACM Mobile Computing and Communications Review **7**(3), 19–20 (2003). DOI http://dx.doi.org/10.1109/MCOM.2002.1018018
43. Lua, E.K., Crowcroft, J., Pias, M., Sharma, R., Lim, S.: A survey and comparison of peer-to-peer overlay networks schemes. IEEE Communications Surveys and Tutorials **7**(2), 72–93 (2004)
44. M. Mani W. Seah, N.C.: Super nodes positioning for p2p ip telephony over wireless ad hoc networks. In: MUM (2007)
45. M. Waldman A. D. Rubin, L.F.C.: Publius: a robust, tamper-evident, censorship-resistant web publishing system. In: Proc. of USENIX. ACM Press, Denver, Colorado, USA (2000)
46. Naumov, V., Baumann, R., Gross, T.: An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces. In: Proc. of ACM MobiHoc, pp. 1568–1576. Florence, Italy (2006)
47. Neumann, A., Aichele, C., Lindner, M., Wunderlich, S.: Better approach to mobile ad-hoc networking (b.a.t.m.a.n.). http://tools.ietf.org/html/draft-wunderlich-openmesh-manet-routing-00 (2008). Work in progress
48. Papadopouli, M., Schulzrinne, H.: A performance analysis of 7ds: a peer-to-peer data dissemination and prefetching tool for mobile users. In: Proc. of IEEE Advances in Wired and Wireless Communications (2001)
49. Perkins, C., Royer, E.: Ad hoc on-demand distance vector routing. In: Proc. of IEEE WMCSA (1999)
50. Perkins, C.E., Royer, E.M.: Ad hoc on-demand distance vector routing. In: Proc. of the IEEE Workshop on Mobile Computing Systems and Applications, pp. 90–100. New Orleans, USA (1999)
51. Plaxton, C., Rajaraman, R., Richa, A.: Accessing nearby copies of replicated objects in a distributed environment. In: Proc. of ACM SPAA (1997)
52. Pouwelse, J., Garbacki, P., Epema, D., Sips, H.: The bittorrent p2p file-sharing system: Measurements and analysis. Peer-to-Peer Systems IV pp. 205–216 (2005). URL http://dx.doi.org/10.1007/11558989_19
53. Pucha, H., Das, S.M., Hu, Y.C.: Ekta: An efficient dht substrate for distributed applications in mobile ad hoc networks. In: Proc. of the Sixth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004). English Lake District, UK (2004)
54. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Proc. of ACM SIGCOMM. San Diego, USA (2001)
55. Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J.: Handling churn in a DHT. In: Proc. of USENIX (2004). URL http://citeseer.ist.psu.edu/648942.html

56. Ripeanu, M.: Peer-to-peer architecture case study: Gnutella network. In: Proc. of Peer-to-Peer Computing, pp. 99–100 (2001). URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=990433
57. Roger Dingledine Michael J. Freedman, D.M.: The free haven project: Distributed anonymous storage service. In: Proceedings of the Workshop on Design Issues in Anonymity and Unobservability, LNCS (2000)
58. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Proc. of ACM/IFIP Middleware. Heidelberg, Germany (2001)
59. Schollmeier, R., Gruber, I., Finkenzeller, M.: Routing in mobile ad hoc and peer-to-peer networks: a comparison. In: Proc. of Workshop on Peer-to-Peer Computing. In Networking (2002)
60. Schollmeier, R., Gruber, I., Niethammer, F.: Protocol for peer-to-peer networking in mobile environments. In: Proc. of ICCCN (2003)
61. Sozer, H., Tekkalmaz, M., Korpeoglu, I.: A peer-to-peer file search and download protocol for wireless ad-hoc networks. to appear in Computer Communications (To Appear)
62. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proc. of ACM SIGCOMM. San Diego, USA (2001)
63. Stojmenovic, I.: Position based routing in ad hoc networks. IEEE Communications Magazine **40**(7), 128–134 (2002). DOI http://dx.doi.org/10.1109/MCOM.2002.1018018
64. Tang, B., Gupta, H., Das, S.: Benefit-based data caching in ad hoc networks. IEEE Trans. on Mobile Computing **7**(3), 62–91 (2008)
65. Varshavsky, A., Reid, B., de Lara, E.: A cross-layer approach to service discovery and selection in manets. Proc. of IEEE Mobile Adhoc and Sensor Systems Conference (2005). URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1542832
66. Wiberg, B.: Porting aodv-uu implementation to ns2 and enabling tracebased simulation. Master's thesis, Uppsala University (2002)
67. Winter, R., Zahn, T., Schiller, J.: Random landmarking in mobile, topology-aware peer-to-peer networks. In: Proc. of FTDCS (2004)
68. Wolfson, O., Xu, B., Yin, H., Cao, H.: Search-and-discover in mobile p2p network databases. In: Proc. of Int. Conf. on Distributed Computing Systems (2006)
69. Yin, L., Cao, G.: Supporting cooperative caching in ad hoc networks. IEEE Transactions on Mobile Computing **5**(1) (2006)
70. Zahn, T.: Structured peer-to-peer services for mobile ad hoc networks. Phd thesis, Freien University Berlin (2006)
71. Zahn, T., Schiller, J.: Mapnas: A lightweight, locality-aware peer-to-peer based name service for manets. In: Proc. of LCN, pp. 499–500. IEEE Computer Society, Washington, DC, USA (2005). DOI http://dx.doi.org/10.1109/LCN.2005.87
72. Zahn, T., Schiller, J.: Dht-based unicast for mobile ad hoc networks. In: Proc. of PerCom (2006). URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1598963
73. Zhu, Y., Rexford, J., Bavier, A., Feamster, N.: Ufo: A resilient layered routing architecture. In: Technical Report TR-780-07, Princeton University (2007)

Opportunistic Information Retrieval in Sparsely Connected Ad Hoc Networks

Mooi-Choo Chuah and Jian-bin Han

1 Introduction

With the advancement in technology, many users carry wireless computing de-vices e.g., PDAs, cell-phones etc. Such devices can form mobile ad hoc networks and communicate with one another via the help of intermediate nodes. Such ad hoc networks are very useful in several scenarios e.g., battlefield operations, vehicular ad hoc networks and disaster response scenarios. The ability to access important information in these scenarios is highly critical. Many ad hoc routing schemes have been designed for ad hoc networks but such routing schemes are not useful in some challenging network scenarios where the nodes have intermittent connectivity and suffer frequent partitioning. Recently, disruption tolerant network technologies [5, 12] have been proposed to allow nodes in such extreme network-ing environment to communicate with one another. Several DTN routing schemes [4, 14] have been proposed.

Although routing is an important design issue for such sparsely connected networks, it is important to design an information retrieval system that works in extreme networking conditions so that information can be accessed rapidly and efficiently. For example, in a battlefield, soldiers need to access information related to detailed geographical maps, intelligent information about enemy locations, new commands from the general, weather information etc. In addition, a particular data item may be of interests to multiple soldiers so replicating the data item at multiple nodes permits other nodes to access it more easily. This allows us to save battery

Mooi-Choo Chuah
Department of Computer Science and Engineering, Lehigh University, 19 Memorial Drive West, Bethlehem, PA 18015, USA, e-mail: chuah@cse.lehigh.edu

Jianbin Han
Department of Computer Science and Engineering, Lehigh University, 19 Memorial Drive West, Bethlehem, PA 18015, USA, e-mail: jih206@lehigh.edu

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_38, © Springer Science+Business Media, LLC 2010

power, bandwidth consumption and the data item retrieval time. Such data caching also means that the source of the data items need not know the identities of the nodes that need to access the data items.

Researches on data access and dissemination techniques in ad hoc and sensor networks are not new. For example in [15], the authors design schemes for placing the storage node such that the total energy cost for gathering data to the storage nodes and replying queries can be minimized. The approach in [15] is not directly applicable to our problem since they assume stationary nodes in their environments. They also do not consider how a node can discover the replicas of the data items. In [1], the authors propose three distributed caching techniques for well-connected ad hoc networks, namely CacheData, CachePath and Hybrid-Cache. However, their techniques are only useful for well-connected ad hoc net-works since they assume the routes from the publisher to the receivers do not change frequently. In [8, 9], we design and study information retrieval schemes for single-attribute queries. In [8, 9], we assume every node is trusted to store data or queries and we do not address multiple-attribute queries.

In this chapter, we consider a system where we have queries with multiple attributes. In addition, a small number of moving index and storage point(ISP)s may be deployed to store replicated data and/or advertise indices of published data items. We compare such a system with another system that does not have ISPs but uses regular nodes encountered opportunistically to store replicated indices or data. Specifically, we present three information retrieval schemes, namely (a) the Pre-determined ISP advertisement (PISA) scheme where ISPs advertise indices of data items with attribute values that fall within different pre-determined ranges, (b) the ORNA-ID scheme which uses opportunistically encountered nodes for advertising indices, and (c) ORNA-DD scheme which uses opportunistically encountered nodes for storing replicated data. Then, via simulation studies, we compare the query performance of these three schemes. Our results indicate that the query success rates achieved by the ORNA-ID scheme is slightly better than that for the PISA scheme in dense environment. This indicates that either architecture can be deployed depending on the trust levels nodes have among themselves or with some specially deployed storage nodes. In addition, our results indicate that the ORNA-DD (that incorporates data duplication) scheme provides 91% gain in query success rate and 128% gain in the overall success rate in the sparsest network scenario which we studied. This shows that index duplication alone is not sufficient to achieve high query success rate in sparse network scenarios. Instead, data duplication needs to be used. Last but not least, our results also demonstrate that the square-root duplication approach can improve the query performance but is not as effective in sparse mobile ad hoc networks as in the wired network environments.

The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we describe our system model. In Section 4, we present the various data, index and query replication techniques and two specific information retrieval schemes we study. In Section 5, we describe our simulation model and present performance evaluation of the proposed schemes. In Section 7, we provide our concluding remarks and discuss future research.

2 Related Work

In [9, 21], the authors considers the storage node placement problem in sensor networks with the goal of minimizing the total energy cost for gathering data to the storage nodes and replying queries. The authors consider both fixed and dynamic tree models. For fixed tree model, they give an exact solution on how to place storage nodes to minimize total energy cost. For the dynamic tree model, they al-low each sensor node to select the storage nodes such that the communication cost for data forwarding and query diffusion and reply can be minimized once the storage nodes have been positioned.

In [21], the authors present three distributed caching schemes, namely (a) CacheData which caches data items that pass by a node, (b) CachePath which caches the path to the nearest cache, and (c) HybridCache which caches the data item if its size is small enough or else the path to the data will be cached. LRU policy is used for cache replacement. These three approaches however do not take into considerations how nodes within a neighborhood can collaborate such that the data caching will always generate better benefits in terms of access cost. In addition, their approach also assumes well-connected ad-hoc networks. In sparse net-works, there may not be a path connecting a querying node and a data generating node. Thus, to ensure good querying performance, a query needs to be replicated and stored at some intermediate nodes to increase the chance of having that query answered by data generating nodes. Alternatively, published data items need to be replicated to increase this chance.

In [1], the authors design a protocol called Mercury to support multi-attribute, range-based searches. There are two main components in Mercury: (a) Mercury handles multi-attribute queries by creating a routing hub for each attribute in the application schema. Each routing hub is a logical collection of nodes. Queries are passed to exactly one of the hubs corresponding to the attributes that are queried, while a new data item is sent to all hubs for which it has an associated attribute. (b) Each routing hub is organized into a circular overlay of nodes and data is placed contiguously on this ring. Their evaluation shows that Mercury is able to achieve logarithmic-hop routing and near-uniform load balancing. Mercury is de-signed for overlay networks with static nodes. Thus, its technique is not directly applicable to the network environments that we study. The PISA scheme we de-sign and evaluate in this paper is an adapted version of the Mercury protocol for mobile environments.

In [20], the authors describe Snoggle, an implementation which uses information retrieval techniques to index information and process user queries. Their hierarchical system consists of indexing points (IPs), a key indexing point (KIP) and distributed object sensors. The IPs collect data from object sensors within their range and organize the data into an inverted index table. IPs periodically send aggregated information to the KIP. The KIP publishes the aggregated information forwarded by all IPs. The IPs also provide message routing for the traffic between IPs, KIP and users. A user can issue a local or distributed query. A user performs a local query by directing his query to a specific IP within the transmission range of his wireless device. A distributed query is performed when the user queries the KIP. The KIP then returns a list of IPs that may have the data items that the user is interested in. The user can then query the individual IPs. Our approach in this pa-per differs from theirs in multiple ways: (a) our architecture is distributed such that it can still work without the indexing point, (b) our system uses replicated data or indices to improve query performance, (d) users can query data-generating nodes directly.

In [13], the author considered the data replica allocation problem in ad hoc networks. More specifically, the author proposed three replica allocation methods, namely (a) Static Access Frequency (SAF) which makes use of the access frequency to each data item, (b) dynamic access frequency and neighborhood (DAFN) method which makes use of the access frequency of data items and the neighborhood of mobile hosts, and (c) dynamic connectivity based group (DCG) which considers both the access frequency and the whole network topology. In SAF, each mobile host caches the data items that it accesses most frequently. In DAFN, the approach is similar to SAF except that replications among neighboring nodes are eliminated. In DCG, nodes that are connected form groups and the replica allocation is determined based on the group access frequency. The disadvan-tage of these approaches is that they assume full knowledge of access frequencies and the ability to share such information in well-connected ad hoc networks.

3 System Model

In this chapter, we assume that there are two types of network entities: (a) regular nodes which are wireless devices e.g., PDAs and cell-phones carried by individuals, and (b) indexing and storage points (ISPs) which are special moving nodes which run server software modules that support content-based information retrievals. These ISPs may be wireless routers mounted on moving vehicles e.g., campus buses [4]. All the nodes (either ISPs or regular nodes) support communi-cations using disruption tolerant networking technology [5, 12]. Users can subscribe to the information retrieval services via the traditional subscription procedures similar to ordering Internet services. The users' wireless devices will run the client modules that support content-based information retrievals.

We further assume that users can publish their data items e.g., region-based weather, sensor data readings, enemy intelligent information for others to query. The data items can be labeled using some meta-data descriptions. For example, a data item describing enemy intelligent information may take the form: (<cate-gory= “Al-Qaeda.GRPlocation”>,<clearance, “ >″, 5 >, encrypted data). This data item basically describes the location of an Al-Qaeda group that only soldiers with a clearance level higher than 5 can access. The nodes that publish data items are referred to as the data nodes. The published data items can be stored at regular nodes or at the ISPs. Any publisher can store his data items at some designated ISPs based on the categories the data items belong to or store his data items at any ISPs that he encounters opportunistically. In addition, a publisher can choose to replicate his published data items to increase the chances of letting others find them.

Any authorized user can send a query to retrieve data items of interests to him. A query may take the form: Query(<category="Al-Qaeda.GRPLocation", <clearance, " >″, 3>, expired:20:00). This query looks for the locations of all Al-Qaeda groups that can be accessed with a clearance level above 3. The query expires at a certain time. Since the nodes in the network scenarios we are considering are frequently partitioned, a user can choose to duplicate his query to increase his chances of getting a reply before his query expires. The nodes that generate the queries are referred to as the querying nodes.

There are four important components in our information retrieval system: (a) the data caching component is related to how and where the published data items are stored, (b) the index advertising component is related to how published data is advertised, and (c) the query dissemination component is related to how queries are disseminated, and (d) the message routing component which determines how a response can be sent to a querying node. We will explain the approaches used for different components in Section 4.

4 Data/Index/Query Dissemination Schemes

In this work, we assume that each data item has two attributes denoted as (x,y) and each of the attribute has a value that falls within a range [dmin,dmax]. Each query has a query identifier and an expiration time. Each data also has a data identifier, some meta-descriptions, and an expiration time. Each data-generating node maintains two bloom filters: one for attribute 1, and one for attribute 2. In this work, we assume that every node sends beacons periodically for node discovery. When a data node or ISP hears the beacon of a new node, it sends an index advertisement (IA) message that contains its bloom filters information. That way, any node knows if the other node (which sent the IA) it encounters has any data items of interest to itself.

4.1 Data Caching

A publisher can (a) store published data items only in his own devices but let ISPs advertise the meta-data descriptions (referred to as index) of his published data items, or (b) select other nodes to store replicated copies of his published data items. Our earlier work [8, 9] has indicated that data replication helps to improve the success rate of single-attribute queries with an expiration time.

A publisher may choose to replicate its data items. In some schemes, special moving nodes called the Index Storage Point (ISP)s are deployed to store data items so a publisher can push its data items to such ISPs. In other schemes, a publisher can binary spread its replicated copies to any regular nodes it encounters. The binary spread [8, 16] method can quickly disseminate all the replicated copies. In non-homogeneous mobility models, a publisher can choose nodes which move faster to store his replicated data items since such nodes can meet more nodes. A friendliness metric (FM), which is a moving average of the number of new nodes encountered by a node within an observation window, is used to identify nodes with higher mobility. The nodes that move faster can meet more nodes and hence have a higher FM value. The FM metric can be included in the beacons periodically sent out by a node. When a publisher's wireless device encounters another node, it can tell whether the other node has a FM value that exceeds a certain threshold. If it does, then the publisher will send his replicated data copies to that node. If the FM metric is not supported, then, a publisher merely binary spreads his replicated copies to any nodes that his wireless device encounters.

4.2 Index Advertising

A publisher may decide to ask other nodes to advertise the data items that he publishes. There are two ways whereby this can be done: (a) publishing the indices at Index Storage Points (ISPs) or (b) publishing the indices at regular nodes that a publisher encounters.

Approach (a) works as follows: when a publisher generates a data item, it will send meta-data descriptions of this data item to some predetermined ISPs. Each ISP is chosen to advertise indices of the data items with their first or second at-tribute values falling into a particular range as in the Mercury [1] approach. For example, assume that the minimum and maximum value of each of the two attributes are 1 and 8 (i.e., dmin=1,dmax=8). ISP1 advertises indices of data items with values of their first attribute falling within the range [1,4] or values of their second attribute falling within the range [5–8], ISP2 advertises indices of data items with values of their first attribute falling within the range [5–8], or values of their second attribute falling within the range [5–8], ISP3 advertises indices of data items with values of their first attribute falling within the range [1–4] or values of their second attribute falling within the range [1–4], and ISP4 advertises indices of data items with the values of their first attribute within the range [5–8] or values of their second attribute within the range [1–4]. Then, when a publisher generates a data item, it will send notification messages to three ISPs which take care of the ranges its attribute values fall into. For example, if a data item has attribute values (1,5), then notification messages will be sent to ISP1, ISP2, and ISP3. Approach (a) assumes that all regular nodes know how the index ranges are distributed among the different ISPs.

Approach (b) works as follows: a publisher generates M tokens for each index of a newly generated data items and binary spread them to nodes that the publisher encounters. That means, each node that has an index with more than one token is allowed to pass half of the tokens to another node that it encounters until it only has an index with one token.

4.3 Query Dissemination

When ISPs are deployed, a querying node knows which ISP to enquire for the information of data nodes with matching data items. Upon receiving the information from the ISP, the querying node can proceed to retrieve the relevant data items. In addition, a node also can retrieve data items from a data node that it en-counters.

A node, n_1, which is interested in data items with certain attribute values e.g., (3,6) does not know where such items are stored if no ISPs are deployed. In such cases, there are two ways whereby node n_1 can find such data items: (a) since each data node broadcasts its index advertisement, any node that can hear such an advertisement can determine if the other node has data items of interest to itself, (b) if n_1 encounters another node that advertises indices, it can query that node if its advertised indices match with pending queries which n1 stores. A response message that contains the data node identifier will be sent to the querying node. Then, the querying node can directly retrieve the data items from the relevant data-generating nodes.

To allow more queries to successfully retrieve data items, a node can decide to replicate its query when no ISPs are deployed. Our early work [8, 9] has shown that query duplication also helps in improving the query success rate and reduces the query response time. If a query is replicated and stored at multiple nodes, then multiple query responses may be generated. Such redundant responses cause the transmission efficiency of the system to be low. To increase the transmission efficiency of the information retrieval system, each node can cache the identifiers of the query responses it has generated so that it does not relay any redundant query responses.

4.4 DTN Message Routing Scheme

Once a query response is generated by a node, the query response will be delivered to the querying node using the underlying DTN message routing scheme. In this paper, we use Prophet [14] as our DTN message routing scheme. Other DTN routing schemes can be used as well.

Prophet uses the history of encounters and transitivity to route messages for intermittently connected networks. In this scheme, each node broadcasts a beacon periodically. This probabilistic routing scheme establishes a probabilistic metric called delivery predictability at every node A for each known destination B. This metric indicates how likely it is that node A will be able to deliver a message to that destination. The delivery predictability ages with time and also has a transitive property, i.e., a node A that encounters node B which encounters node C al-lows node A to update its delivery predictability to node C based on its (A's) de-livery predictability to node B and node B's delivery predictability to node C. In Prophet, a node will forward a message to another node it encounters if that node has higher delivery predictability to the destination than itself. Such a scheme was shown to produce superior performance than epidemic routing [18]. The three equations used for updating the delivery predictability are as follow:

$$P(a,b) = P(a,b)_{old} + (1 - P(a,b)_{old}) \cdot \alpha \tag{1}$$

$$P(a,b) = p(a,b)_{old} \times \gamma^{k} \tag{2}$$

$$P(a,c) = P(a,c)_{old} + (1 - P(a,c)_{old}) \times P(a,b) \times P(b,c)\beta \tag{3}$$

In [14], α is set to 0.75, β is set to 0.25 and γ is set to 0.98. When a node receives a message that needs to be relayed, it will pass it to another node which has the highest delivery predictability to the destination.

4.5 Three Information Retrieval Schemes

In this work, we specifically consider three schemes: (a) Pre-determined ISP Advertisement (PISA) scheme, (b) Opportunistic Regular Node Advertisement with Index Duplication (ORNA-ID) scheme, and (c) Opportunistic Regular Node Advertisement with Data Duplication (ORNA-DD).

In PISA, only index duplication is used but not data or query replication. In PISA, ISPs are used to store indices of all generated data items. Every time a data generating node generates a data item, it will send meta-descriptions of that data item to three pre-determined ISPs. The ISPs also periodically broadcast all the indices they have received. In addition, a data node will also advertise indices of all data items they publish to any newly discovered neighbor. When a node n_1 encounters a data-generating node, n_2, n_1 can determine from the index advertisement sent by n_2 whether n_2 carries any data items that match the stored queries n_1 has. If n_2 has matching data items, then n_1 will retrieve such items from n_2. When a node n_1 generates a query, it will send the query to the closest ISP that contains the index information of data items that match this query. Upon receiving the reply from the ISP, node n_1 will send queries to the appropriate data-generating nodes to retrieve the data items of interest.

In the ORNA-ID scheme, only index duplication but not query or data replication is used. In this scheme, there is no ISP. Regular nodes are used to store indices of the published data items and/or queries that have not expired yet. Every node that generates a data item creates M copies of an index, and binary-spray these M copies of the index to nodes that they encounter. Every node that has more than one copy of the index can give half of the copies it has to any node that it encounters that does not have this index yet. Binary spraying is used because this is the quickest way of spreading extra message copies to other nodes in sparsely connected networks. When a node encounters another node, they will exchange their aggregated indices with each other. Thus, a node n1 can determine if the other node (say n_2)

it encounters has any data items that match its stored queries. If there is, n_1 will send n_2 queries to retrieve those data items. Every query contains the identifier of the original querying node. If n_2 does not contain matching data items but have index information of the matching data items, then n_2 will send information of the data-node that generate these matching data items to n_1. Next, n_1 can send messages directly to the originating data nodes to retrieve such data items. In addition to index duplication, query duplication can be used in this scheme too. The advantage of the ORNA-ID scheme over the PISA scheme is that no special ISP nodes need to be deployed. The disadvantage is that a node has to trust all other regular nodes.

In the ORNA-DD scheme, only data duplication is used. Regular nodes are used to store replicated copies of the published data items. Every data-node generates W copies of any data item it produces and binary spread these copies to the nodes it encounters. Again, any node that stores data items advertises the aggregated indices of these data items to any new nodes it encounters. If the other node discovers that the advertised list contains data items that match its stored queries, it will send messages to retrieve the data items.

5 Performance Evaluation

5.1 Simulation Setup

To investigate the usefulness of the three informational retrieval schemes we describe in Section 4.5, we implement these schemes in NS-2 version 2.31 [17]. We assume that the wireless bandwidth is 2 Mb/s and the transmission range is 250 m. We use two network scenarios: (a) a network that consists of 40 nodes randomly distributed over (i) 1400×1400 m^2, (ii) 2000×2000 m^2, (iii) 3000×3000 m^2 , and (iv) 4000×4000 m^2, (b) a network with 100 nodes distributed over a larger area than scenario (a) but maintains the same node densities.

5.1.1 Node Movement Model

The nodes move either according to (a) the random waypoint (RWP) mobility model [3], (b) the Zebranet [19] model, (c) the UMass [23] model, or (d) the Community-Based (CB) model. For the RWP model, each node selects a random destination and moves towards the destination with a speed chosen randomly between (vmin, vmax) m/s. After the node reaches its destination, it pauses for a period of time and repeats this movement pattern. Unless otherwise stated, vmin is set to 0, and vmax is set to 5 m/s for all nodes in the homogenous model. For the Zebranet model [19], we create a semi-synthentic model as follows: we synthesize node speed and turn angle distributions from the observed data and create other node-movements using the same distribution. We use both distance and time scaling to fit the original data found in the trace into the network environment that we are interested in. The average node speed in the Zebranet model is 6m/s. The UMass model is based on traces collected from a real vehicular ad hoc network. For this model, we extract the locations of twenty buses at different times from multiple traces, and scale their relative locations to fit into the geographical area of interest. We mimic the CB model proposed in [16] as follows: x% of nodes randomly choose their initial locations and move according to RWP within a restricted area (1% of the total area) centered around their initial locations; while, the other (100–x)% of nodes moves freely according to RWP within the entire simulation area. The default value of x is 50.

5.1.2 Data Item Generation Model

There are 10 data generating nodes. Each data node generates a data item every 50s. The values of the two attributes of each data item are chosen uniformly from the range [1,8]. Therefore, there are 64 types of data items. We keep track of the number of data items of each type in the system and re-select attribute values of a new data item such that there are about 3 copies of data items of a particular type. Each data item has a default expiration time of 1000s. The data size is fixed at 1000 bytes.

5.1.3 Query Model

There are 20 nodes that generate queries. By default, each querying node generates a query every 50s. The query looks for all data items of a particular type (recall that we have 64 types of data items and there are on the average three data items per type). Each query has an expiration time of 1000s. Each query either uniformly chooses one of these 64 data types or chooses it based on Zipf-like distribution. The Zipf-like distribution has been frequently used [16] to model non-uniform access pattern. In the Zipf-like distribution, the access probability of the *i*th (data item is represented as follows:

$$P_i = \frac{1}{i^{\Theta} \sum_{k=1}^{1} \frac{1}{k^{\theta}}} \tag{4}$$

where $0 < \theta < 1$. When $\theta = 1$ (the default value we use), it follows the strict Zipf distribution. $\theta = 0.8$ has been shown to generate access pattern similar to those real web traces [2].

The performance metrics that we use to compare different combinations of schemes are

- Query success ratio – there are two metrics related to this. The first one (referred to as query success rate) measures the ratio of the number of queries which successfully retrieve at least 1 data item over the total number of generated queries. The second one (referred to as overall success ratio) measures the ratio of queries that successfully retrieve all data items in that category over the total number of generated queries.

- Query response time – this is measured as the average time it takes for the successful query response to arrive back at a node that issues a query.
- Data efficiency – this is measured as the number of useful data bytes over the number of total transmitted data bytes (does not include control over-head but include the data bytes due to the transmissions of replicated data copies). Each data point reported in the simulation results section is the average of 10 runs.

5.2 Simulation Results

5.2.1 Impact of Node Density

In our first experiment, we compare the performance of the various schemes using networks of different node density. For the ORNA-ID scheme, we replicate four copies of each index. For the ORNA-DD scheme we replicate four copies of each generated data item. Each query uniformly chooses one of the data categories. We use the 40-nodes network but vary the area which the nodes move. The nodes move according to either the RWP or the Zebranet mobility model. We will discuss more on the impact of mobility models in the next subsection. Here, we focus our discussion on the performance comparison of the three schemes.

The results for average query success ratio and overall success ratio for the PISA, ORNA-ID, ORNA-DD schemes are plotted in Figs. 1, 2 and 3 respectively. The results indicate that as the network becomes sparser, fewer queries can be successful due to the increasing node encounter time. The results also indicate that the ORNA-ID scheme performs slightly better than the PISA scheme in denser envi-

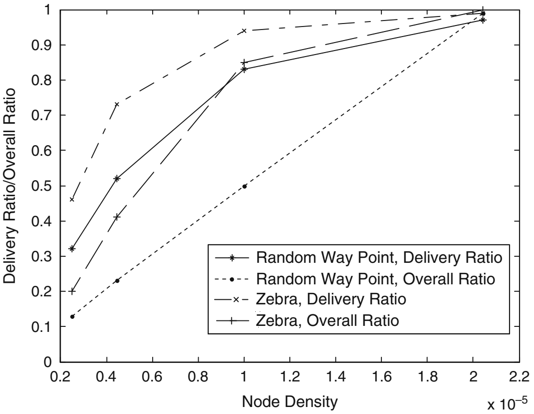


Fig. 1 Success ratio/overall success ratio for PISA

ronment but similar success rate in very sparse environment irrespective of whether the RWP or Zebranet model is used. This indicates that either architecture can be used in real-life deployments. The PISA scheme should be used if nodes only trust specially deployed storage nodes while the ORNA-ID scheme can be used if nodes trust one another. The ORNA-DD gives the best query success ratio, the overall success rate, and smaller query response time. The ORNA-DD scheme achieves 91% higher query success rate and 128% higher overall success rate when compared to those achieved using the other two schemes in the sparsest scenario (40 nodes over 4000 × 4000 m^2). This shows that data duplication yields better results than index duplication in sparse networks. The only drawback of the ORNA-DD scheme is that more storage space is required in the network. We also have conducted some simulation studies where the query replication feature is added to the ORNA-ID scheme but the results indicate that there is no further improvement in the query performance.

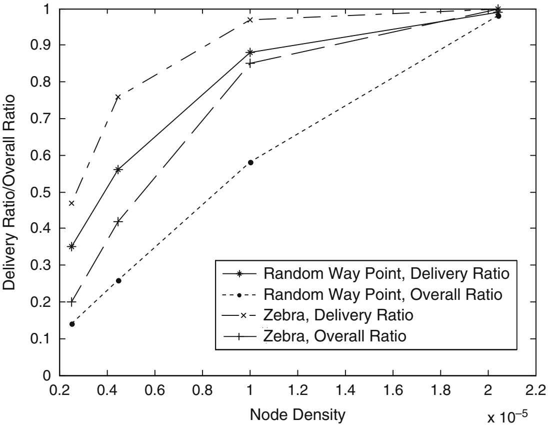


Fig. 2 Success ratio/overall success ratio of the ORNA-ID scheme

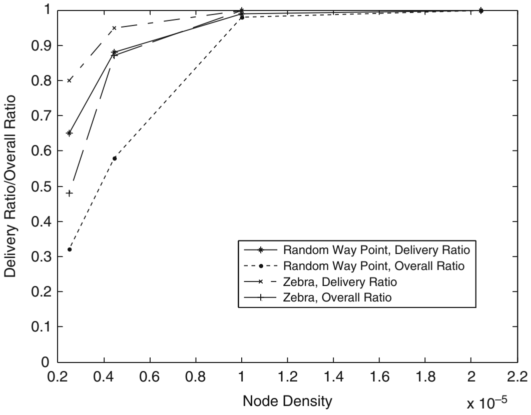


Fig. 3 Success ratio/overall success ratio of the ORNA-DD scheme

Figures 4 and 5 plot the successful query response time distribution obtained using the PISA and the ORNA-DD schemes. We did not include the response time distribution of the ORNA-ID scheme because it is very similar to that achieved using the PISA scheme. The average query response time achieved by the ORNA-DD scheme is the smallest because there are more copies of the same data item in the network and hence a querying node can retrieve it faster. The average query response time for the PISA scheme is slightly better than the ORNA-ID scheme because the querying nodes know which ISP to query for the information of data items that will match its stored queries.

Figure 6 plots the data efficiency obtained using the PISA and the ORNA-DD schemes. The data efficiency for the ORNA-ID scheme is not shown because it is similar to that for the PISA scheme since the same index replication factor is used. From Fig. 6, we see that the data efficiency for the ORNA-DD scheme is lower than that of the PISA or the ORNA-ID scheme because the query generation rate we use is comparable to the data generation rate. Thus, the cost of data duplication is

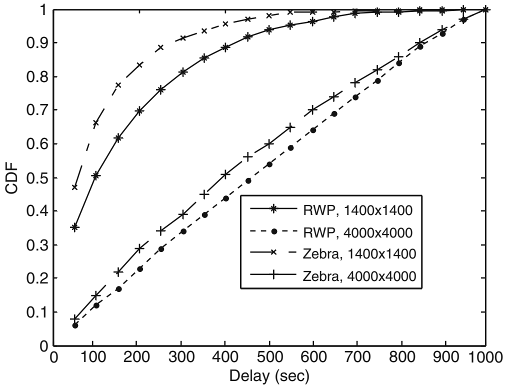


Fig. 4 CDF of the successful query response time for the PISA scheme

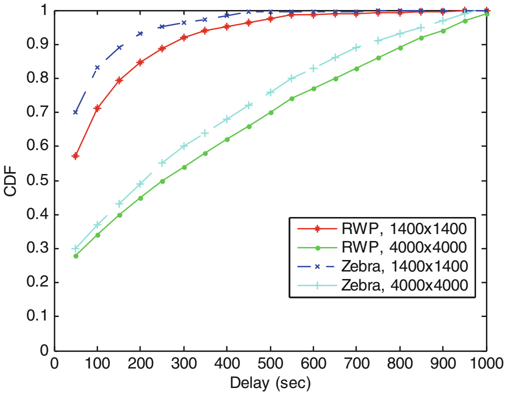


Fig. 5 CDF of the successful query response time for the ORNA-DD scheme

not amortized over many successful queries. When we increase the query rate to 1 query/10s, then ORNA-DD achieves higher data efficiency than the PISA/ORNA-ID scheme.

Another interesting observation from Fig. 5 is that the data efficiency achieved using the PISA scheme has a concave shape with its lowest data efficiency occurring when the node density corresponds to the scenario with 40 nodes distributed over 2000 × 2000 m^2. The concave shape can be explained as follows: we only compute the data efficiency of successful queries. As the network becomes sparser, it takes longer time for the querying node to get a reply from the ISP and then retrieve data items from a data-node so the data efficiency drops. Beyond a certain node density level, the inter-node encounter time becomes too long that such queries can no longer be successful. In sparser networks, queries can only be successful if the querying nodes happen to discover data nodes (which results in higher data efficiency because data items are retrieved using single hop transmissions). However,

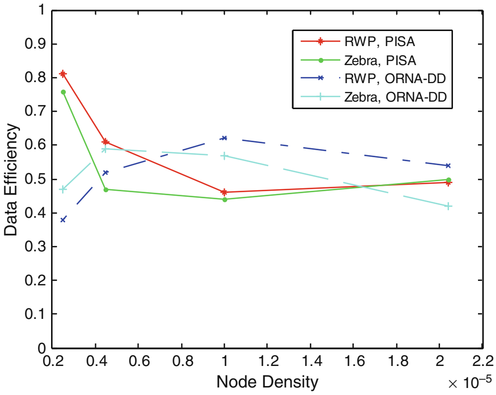


Fig. 6 Data efficiency vs. node density for the PISA and the ORNA-DD schemes

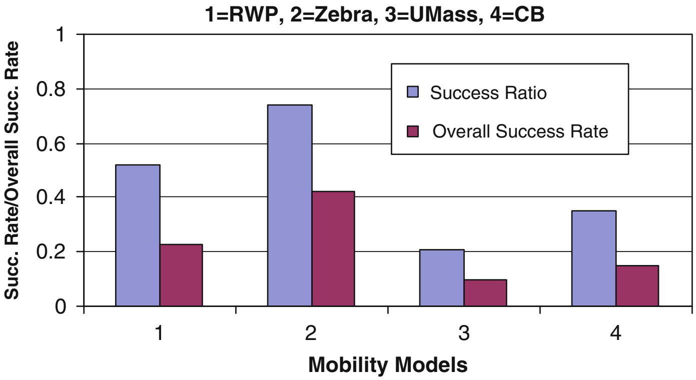


Fig. 7 Success rate/overall success rate for different mobility models

a convex shape is observed for the ORNA-DD scheme. This may be due to the fact that having 4 replicated copies in the 40 nodes over 2000 × 2000 m^2 scenario gives the querying nodes the best chance to encounter data nodes and hence the highest data efficiency. It will be interesting to develop an analytical framework to compute the optimal number of data copies to replicate to achieve certain query success rate and data efficiency.

5.2.2 Impact of Mobility Model

From the plots in Figs. 1, 2, 3, 4, 5 and 6, one can see that the ORNA-DD scheme achieves the best query performance irrespective of which mobility model is used. In addition, we observe that higher query success rate and overall success rate is achieved when the Zebranet model is used as compared to those achieved using the RWP model. This is because the nodes within the Zebranet model move faster than the nodes in the RWP model. Recall that the average node speed for the RWP model is 2.5 m/s while the average node speed for the Zebranet is 6 m/s. The mean successful query response time is also smaller with the Zebranet model.

To further understand the impact of mobility models on the query performance of the ORNA-DD scheme, we choose the 40 nodes distributed over 3000 × 3000 m^2 scenario and repeat the experiment in Section 5.2.1 using the UMASS and CB models. The results for the query success rate, the overall success rate and data efficiency are plotted in Figs. 7 and 8 respectively. From the plots, one can see that the query performance achieved when the nodes are moving according to the Zebranet model is the best because the nodes move on the average with a speed of 6 m/s. The performance for the CB model is poorer than the RWP model because fewer queries can be satisfied with some nodes moving only within a small local area in the CB model. The UMASS model has the highest internode encounter time among

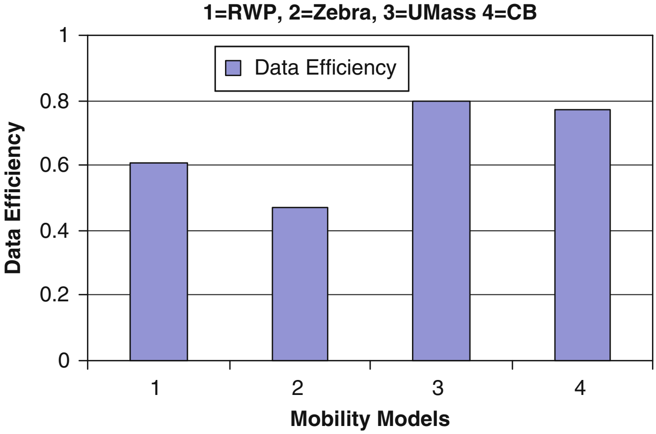


Fig. 8 Data efficiency for different mobility models

the 4 mobility models, and hence fewer queries can be answered before they expire. The data efficiency plot is a bit misleading since it only represents the data efficiency of successful queries. This plot indicates that for UMASS and CB models, only those queries that can be satisfied with fewer hops can be successful.

5.2.3 Impact of the Index/Data Replication Factors

In order to ensure that the schemes we design still perform well in larger networks, we conduct some experiments using 100 nodes that are distributed over (a) 1600 × 1600 m^2, (b) 3200 × 3200 m^2, (c) 4750 × 4750 m^2, and (d) 6300 × 6300 m^2. The size of the geographical area is chosen such that the node densities for the four cases are the same as in the 40-node scenario. The nodes move according to the RWP

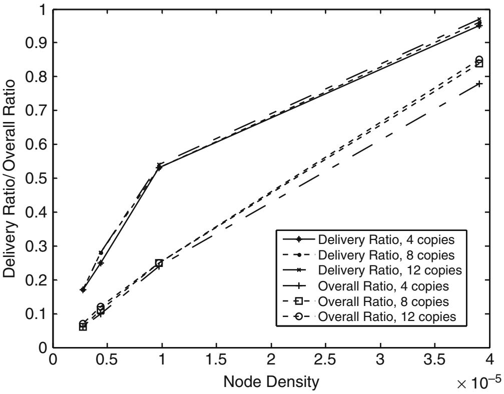


Fig. 9 Query success rate/overall success rate for the ORNA-ID scheme with different index replication factors

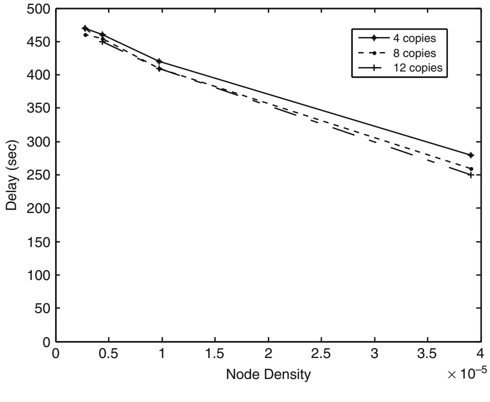


Fig. 10 Average query response time for the ORNA-ID scheme with different index replication factors

model within a specified area. Ten nodes are selected to be data generating nodes with each node generating a data item every 50s. 20 nodes are randomly selected to be querying nodes with each generating queries at a rate of 1 query/10s. Again, a uniform query model is used in this experiment. We study both the ORNA-ID and ORNA-DD schemes with varying number of replicated index or data copies. We vary the number of replicated indices from 4 to 12 for the ORNA-ID scheme and vary the number of replicated data copies from 4 to 12 for the ORNA-DD schemes. Figures 9, 10, and 11 plot the query success rate, overall success rate, average query response time and data efficiency for the ORNA-ID scheme. Figures 12, 13, 14 plot the query success rate, overall success rate, average query response time, and data efficiency for the ORNA-DD scheme.

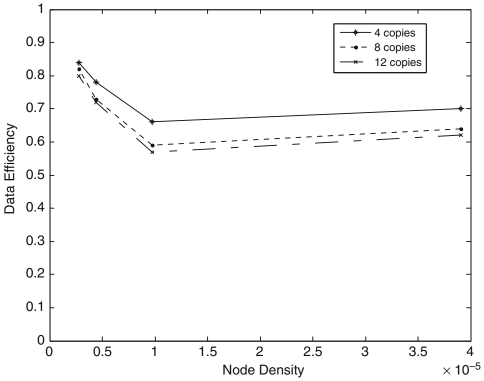


Fig. 11 Data efficiency for the ORNA-ID scheme with different index replication factors

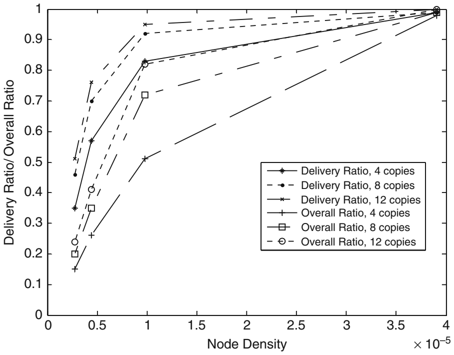


Fig. 12 Query success rate/overall success rate for the ORNA-DD scheme with different data replication factors

Figures 9 and 10 indicates that the query success rate/overall success rate, and the average response time does not improve significantly as the number of index duplications increases from 4 to 12. In addition, the data efficiency drops (shown in Fig. 11) as the index replication factor increases from 4 to 12. However, if data replication is used, we see from Figs. 12 and 13 that the query success rate/overall success rate can improve by 8.4 to 31.4% and the average response time reduces by a factor of 8 to 27% when the data replication factor increases from 4 to 8 and the node density varies from the least dense to the densest scenario. From Fig. 14, one can see that the price to pay is a decrease in data efficiency. The decrease in data efficiency is about 16 to 10% when the data replication factor increases from 4 to 8. The improvement in the delivery ratio as the data replication factor changes from 8 to 12 is another 8–10% but at the cost of another 10% drop in the data efficiency. Thus, 8 copies seem to be a good compromise if the fixed replication factor approach is used for this network model.

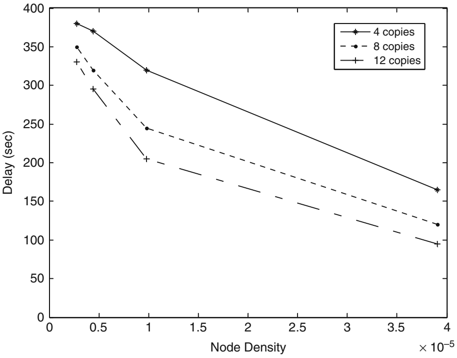


Fig. 13 Average query response time for the ORNA-DD scheme with different index replication factors

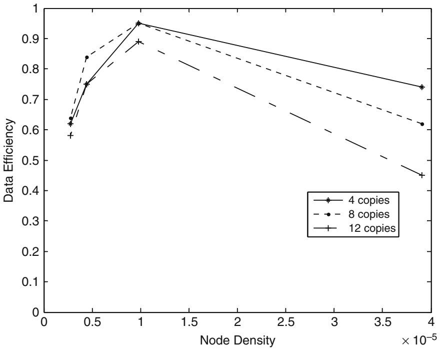


Fig. 14 Data efficiency for the ORNA-DD scheme with different index replication factors

5.2.4 Non-uniform Access Pattern

Next, we investigate how the access patterns affect the query success rate, delay, etc when the ORNA-DD scheme is used. As indicated in Section 5.1, Zipf-based queries are often used to represent non-uniform access patterns. In [10], the authors found that using square-root replication factor is useful for non-uniform access patterns. Thus, we also compare the query performances we obtain by using a fixed number of data replication factor (set to 4 per data item) or using a square root replication factor which is based on the access frequency of each data item. For the square root approach, the replication factor used for the top 5 frquently accessed data items are (i) 13, (ii) 10, (iii) 8, (iv) 6, and (v) 6. The rest of the 32 more frequently accessed data items are replicated 4 times, while the remaining 27 items are replicated 3 times. We use the 100-node scenario and the nodes move according to the RWP model. The query model used is the Zipf-based model. Figures 15, 16, and 17 plot the results we obtain for the query success rate/overall success rate, average response time, and data efficiency respectively. From the plots, we see that the

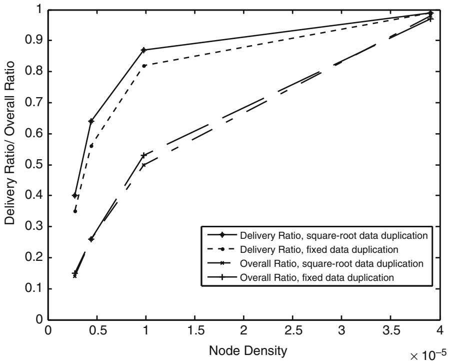


Fig. 15 Query success rate/overall success rate for the ORNA-DD scheme

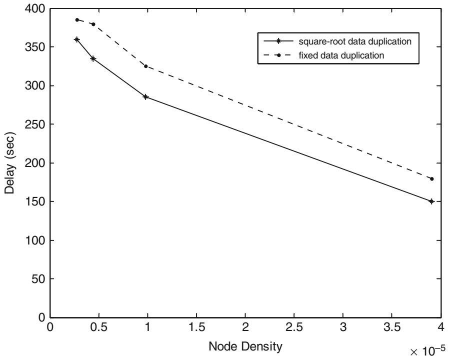


Fig. 16 Avg response time for the ORNA-DD scheme

square root replication factor does give higher query success rate but similar overall success rate when compared to that achieved in the fixed copy scheme. The average response time using the square-root replication factor is smaller than using the fixed copy approach. Since more frequently accessed data items have higher replication factors, they can be found more easily by the querying nodes, and hence result in smaller query response times. Since the more frequently accessed data items can be retrieved using fewer hops, the data efficiency using the square root approach is slightly higher than that achieved us-ing the fixed copy approach.

We are also interested in understanding how our scheme performs when a non-homogeneous mobility model e.g., the Community-based mobility model is used. Thus, we repeat the experiment using the CB model where 50 nodes move locally and 25 nodes move globally. The locally moving nodes are constrained to move within an area which is 1% of the whole area. In this experiment, we still have 10 nodes generating data items and 20 nodes generating queries. For the CB model, both the data and querying nodes are randomly selected from the locally moving nodes. The data generation rate is set such that there are a total of 200 data items

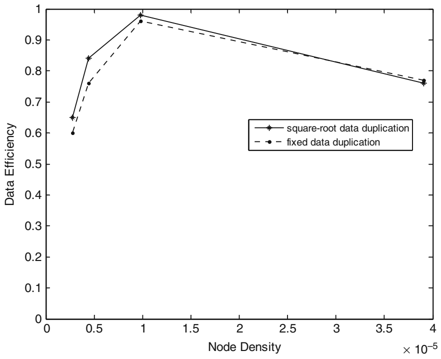


Fig. 17 Data efficiency for the ORNA-DD scheme

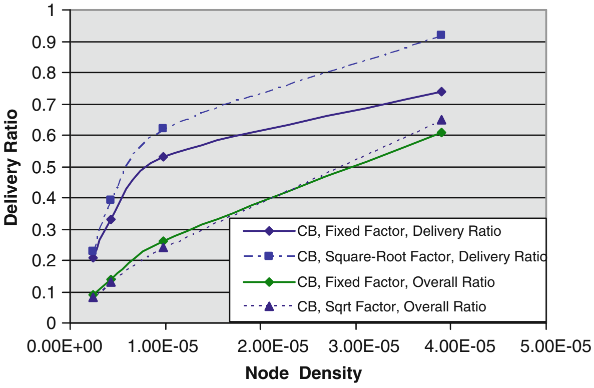


Fig. 18 Query success rate/overall success rate (CB Model)

in the network. The data expiration time is set at 1000s. The query generation rate is set to 1 every 10s. The query expiration is also set to 1000s. The query model we use is the Zipf-based query model. The replication approach we use is either the square-root-based or fixed replication factor (set to 4 copies) approach.

We plot the query success rate/overall success rate, average retrieval latency and data efficiency results obtained for both replication approaches with the CB model in Figs. 18, 19, and 20 respectively. Our results indicate that the square-root replication approach still gives better performance than the fixed approach with the CB model. The query performance difference between the two replication approaches is more obvious in the CB model. Compared Figs. 15 and 18, we see that the query success ratio is lower with the CB model when compared to that with the RWP model. The degradation is caused by the fact that if a data node selects a locally moving node to cache its data, then it is more difficult for the querying nodes (which are also locally moving nodes) to retrieve data items from them. The data efficiency (in Fig. 20) for the CB model is lower than that for the RWP model (in Fig. 20).

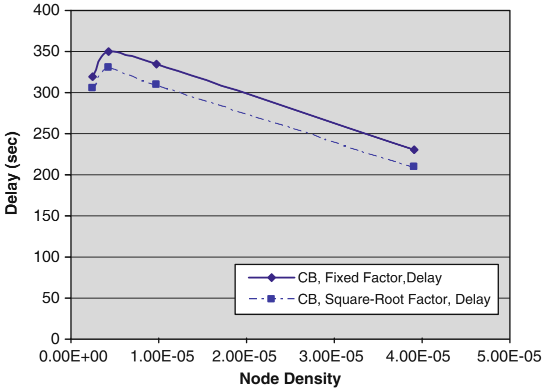


Fig. 19 Avg retrieval latency (CB model)

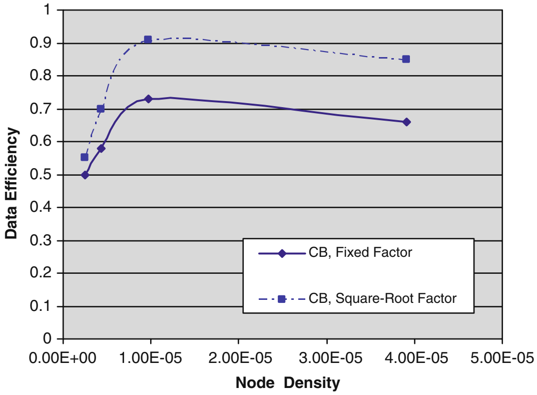


Fig. 20 Data efficiency (CB Model)

6 Discussion

In the previous section, we have evaluated three information retrieval schemes, namely, the PISA, ORNA-ID and ORNA-DD schemes. The PISA scheme assumes that special nodes are deployed to act as the index storage points. The per-formance of the PISA scheme will be similar to the scheme proposed in the Snoggle paper [20]. These two schemes assume that index storage points advertise the available data items as well as where such data items can be found in the system. These two schemes also allow regular nodes that happen to be within the transmission range of the storage nodes to retrieve data items directly from the storage nodes. In a sparse network environment, the inter-node encounter times are long and hence these two schemes will not perform better than the ORNA-DD scheme where data items are replicated and stored at multiple storage nodes. However, one may argue that the improvement in the data retrieval success rate achieved by the ORNA-DD scheme comes with a cost – extra storage capacity needs to be provided to store the replicated data items. We counter argue that storage capacity is cheap these days compare to the transmission cost. By providing replicated data copies, a querying node has a higher chance of encountering a storage node that has the data items it is looking for. Our results have shown that relatively high data retrieval success rate can be achieved by merely replicating four copies of each data item.

In addition, one would be interested in knowing how the retrieval performance achieved by the ORNA-DD scheme compares to that of the CACHE DATA scheme [21] we discuss in the related work section. To allow such comparison, we assume that each querying node knows the identifiers of the ten data generating nodes. Thus, a querying node will send a data retrieval message to all these ten nodes whenever a query is generated. Upon receiving a query, each data generating node will check if it carries data items that match the attributes of interest to the querying node. We implement this CACHE DATA scheme and compare the results we obtain for the ORNA-DD scheme earlier using network scenarios with different node densities. Both the data and query expiration time is set to 1000s. Tables 1 and 2 tabulate the results we obtain for the CACHE-DATA scheme and the ORNA-DD scheme. The results indicate that the ORNA-DD scheme achieves higher data retrieval success rate at a lower storage cost. Since nodes exchange their query lists upon encountering each other, and queries are replicated to all the ten data-generating nodes, the

Table 1 Simulation results for CACHE-DATA scheme

	First delivery ratio (%)	Overall delivery ratio (%)	Latency	Data efficiency	Buffer usage
1400 × 1400	54	40	14.5	0.13	19.7
2000 × 2000	52	32	51	0.09	16.4
3000 × 3000	41	6	131	0.06	13.2
4000 × 4000	28	2	181	0.04	11.9

Table 2 Simulation results for ORNA-DD scheme

	First delivery ratio (%)	Overall delivery ratio (%)	Latency	Data efficiency	Buffer usage
1400 × 1400	90	87	211	0.25	3.8
2000 × 2000	81	58	320	0.22	3.8
3000 × 3000	55	29	455	0.29	3.7
4000 × 4000	36	17	509	0.36	3.5

intermediate nodes end up caching many data items on transit,and hence the storage cost of the CACHE-DATA scheme becomes higher than that for the ORNA-DD scheme.

7 Concluding Remarks

In this chapter, we have presented three information retrieval schemes, namely the PISA, ORNA-ID and ORNA-DD schemes which can work well in sparsely connected ad hoc networks. Via simulations, we compare their performances using two-attribute queries which either uniformly retrieve available data categories or retrieve categories based on a Zipf-based model. Our results indicate that when the network is sparse, index duplication is not sufficient to achieve high query success rate. Data duplication needs to be used instead. Our results also show that with data duplication, the query success rate can improve by 91% and the overall success rate can improve by 128%. In addition, we have shown that for non-uniform access patterns, the square root based replication technique can still improve the query performance even though not all data copies can be disseminated before the data expires in very sparse environment.

In this chapter, we have used synthetically generated mobility models. In the near future, we intend to compare the performance of our schemes with another scheme that has been designed [22] using more realistic mobility models [6, 11]. In addition, we intend to study the retrieval performance with range queries. Last but not least, before any information retrieval system using our designed schemes can be deployed, security issues need to be addressed. In [7], we present the security design of such a system.

Acknowledgement

This work has been supported by DARPA under Contract W15P7T-06-C-P430. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsor of this work. The authors would like to thank Peng Yang for helping to generate the simulation results for the CACHE data scheme.

References

1. A. R. Bharambe et al. (2004). Mercury: supporting scalable multi-attribute range queries. In *Proceedings of ACM Sigcomm*, Aug/Sept
2. L. Breslau et al. (1999). Web Caching and Zipf-like Distributions: Evidence and Implications. In *IEEE Infocom*.
3. J. Broch et al. (1998). A Performance Comparison of Multhop wireless Adhoc Network Routing Protocols. In *ACM Mobicom*.
4. J. Burgess et al. (2006). MaxProp: Routing for vehicle-based disruption tolerant networks. In *Proceedings of IEEE Infocom*
5. V. Cerf et al. (2007). Delay Tolerant Networking Architecture. *RFC4838*
6. A. Chaintreau et al. (2007). Impact of Human Mobility on Opportunistic Forwarding Algorithms.*IEEE Transaction on Mobile Computing*, pp 606–620
7. M. Chuah, R. Metgzer (2008). Secure Data Retrieval System for DTNs. In *Proceedings of IEEE Milcom*, Oct/Nov
8. M. Chuah, P. Yang (2007). Data-Centric information retrieval schemes for Disruption Tolerant Net-works. In *Lehigh CSE Department Technical Report*.
9. M. Chuah, P. Yang (2007). Performance evaluations of a content-based information retrieval scheme for DTNs. In *Proceedings of Milcom*
10. E. Cohen, S. Shenker (2002). Replication strategies in unstructured peer-to-peer networks. In *Proceedings of ACM Sigcomm*
11. N. Eagle, A. Pentland (2005). Reality Mining: Sensing Complex Social Systems. *Journal of Per-sonal and Ubiquitous Computing.*
12. K. Fall (2003). A delay tolerant network architecture for challengednetworks. In *Proceedings of ACM Sigcomm*
13. T. Hara (2001). Effective Replica Allocation in AdHoc Networks for Improving Data Accessibility. In *Proceedings of IEEE Infocom*.
14. A. Lingren et al. (2004). Probabilistic Routing in Intermittently Connected Networks. In *Proceedings of Workshop on Service Assurance with Partial and Intermittent Resources*
15. B. Sheng et al. (2006). Data Storage Placement in Sensor Networks. In *Proceedings of ACM Mobihoc*
16. T. Spyropoulos et al. (2007). Efficient routing in intermittently connected mobile networks: multiple copy case. *IEEE/ACM Transactions on Networking*
17. In *The network simulator ns-2*. `http://www.isi.edu/nsnam/ns/`.
18. A. Vahdat, D. Becker. Epidemic Routing for partially connected adhoc networks. *Technical Report CS-200006*, Duke University
19. Y. Wang et al. (2005). Erasure-Coding Based Routing for Opportunistic Networks. In *Proceedings of ACM workshop on DTN*.
20. H. Wang et al. (2008). Snoogle: A search engine for the physical world. In *Proceedings of IEEE Infocom*.
21. L. Yin, G. Cao. (2004). Supporting Cooperative Caching in Adhoc Networks. In *Proceedings of IEEE In-focom*
22. E. Yoneki et al. (2007). A Social-Aware Overlay for Publish/Subscribe Communication in Delay Tolerant Networks. In *Proceedings of ACM MSWiM*, Oct
23. X. Zhang et al. (2007). Modeling of a Bus-based Disruption Tolerant Network Trace: Mobility Model-ing and Impact on Routing. In *Proceedings of ACM Mobicom*.

The MOBI-DIK Approach to Searching in Mobile Ad Hoc Network Databases

Yan Luo, Ouri Wolfson, and Bo Xu

Abstract In this chapter, we introduce the mobile ad-hoc network (MANET) database by discussing its definition, historical background and scientific fundamentals. Existing related projects are presented and classified into two main categories, pedestrian and vehicular projects based on their target users. Two main paradigms (i.e., report pulling and report pushing) for answering queries in MANET databases are discussed in details. Then we present the MOBIDIK approach to searching in MANET databases and compare it with alternatives. Finally, the key applications and the future research directions are addressed.

1 Introduction: MANET Databases

A Mobile Ad-Hoc Network (MANET) database is a database that is stored in the peers of a MANET. The network is composed by a finite set of mobile peers that communicate with each other via short range wireless protocols, such as IEEE 802.11, Bluetooth, Zigbee, or Ultra Wide Band (UWB). These protocols provide broadband (typically tens of Mbps) but short-range (typically 10–100 meters) wireless communication. On each mobile peer there is a local database that stores and manages a collection of data items, or reports. A report is a set of values sensed or entered by the user at a particular time, or otherwise obtained by a mobile peer. Often a report describes a physical resource such as an available parking slot. All the local databases maintained by the mobile peers form the MANET database. The

Yan Luo
Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA

Ouri Wolfson
Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA

Bo Xu
Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_39, © Springer Science+Business Media, LLC 2010

peers communicate reports and queries to neighbors directly, and the reports and queries propagate by transitive multi-hop transmissions. Figure 1 below illustrates the definition.

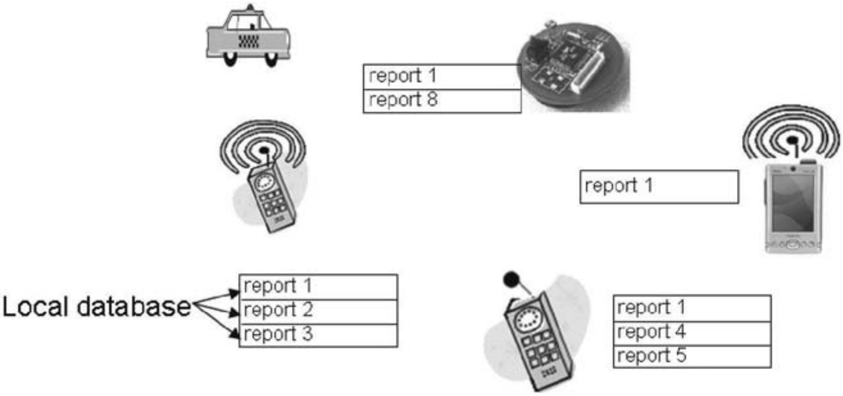


Fig. 1 A MANET database

MANET databases enable matchmaking or resource discovery services in many application domains, including social networks, transportation, mobile electronic commerce, emergency response, and homeland security.

Communication is often restricted by bandwidth and power constraints on the mobile peers. Furthermore, often reports need to be stored and later forwarded, thus memory constraints on the mobile devices constitute a problem as well. Thus, careful and efficient utilization of scarce peer resources (specifically bandwidth, power, and memory) are an important challenge for MANET databases.

2 Historical Background

Consider mobile users that search for local resources. Assuming that the information about the existence and location of such a resource resides on a server in the cloud, a communication infrastructure is necessary to access the server. Such an infrastructure may not be available in military/combat situations, disaster recovery, in a commercial flight, etc. Even if the infrastructure and a server are both available, a user may not have the properly equipped device (e.g., the device may have Bluetooth but not Wi-Fi), or may not be willing to pay the dollar-cost that is usually involved in accessing the server through the cellular infrastructure. In other words, a cloud-computing search may have accessibility and cost problems.

Currently, Google and local.com provide static local information (e.g., the location of a restaurant, pharmacy, etc.), but not dynamic information such as the current exact location of a taxi cab, a nearby person of interest, or an available parking slot.

These dynamic resources are temporary in nature, and thus require timely, real-time update rates that cannot be provided by web crawling.

In cloud-computing search, the searches issued by a user may be tracked by a search engine and used to find the identity of the user. The association of searches with the user identity may generate serious privacy concerns, a lesson given by the recent AOL incident (see [1, 8]) . A concern that is particular to conducting local search in the cloud is that the location of the user is potentially disclosed when the user provides his/her area of geographical interest to the search engine. Furthermore, some information, such as the status of home appliances, is simply not worth putting in the cloud.

Cloud-computing search can be complemented by a MANET database search. The MANET database is distributed among local peers, thus the MANET search does not depend on availability of an infrastructure. This also makes the MANET database more amenable to host the information that is otherwise not worth putting in the cloud. Furthermore, communication among mobile peers is free since it uses the unlicensed spectrum, thus the cost concern is addressed. The response time of MANET database search is higher because the database of neighboring peers can be accessed faster than the web is crawled. Finally, MANET database search addresses the privacy concern because the P2P communication can be totally anonymous, and searches as well as the database are distributed among autonomous entities.

Currently, there are quite a few experimental projects in MANET databases. These can be roughly classified into pedestrians and vehicular projects. Vehicular projects deal with high mobility and high communication topology change-rates, whereas pedestrians projects have a strong concern with power issues. The following are several active experimental MANET database projects for pedestrians and vehicles:

3 Scientific Fundamentals

There are two main paradigms for answering queries in MANET databases, one is report pulling and the other one is report pushing.

Report pulling means that a mobile peer issues a query which is flooded in the whole network, and the answer-reports will be pulled from the mobile peers that have them (see e.g., [5]). Report pulling is widely used in resource discovery, such as route discovery in mobile ad hoc networks and file discovery by query flooding in wired P2P networks like Gnutella. Flooding in a wireless network is in fact relatively efficient as compared to wired networks because of the wireless broadcast advantage, but there are also disadvantages which will be explained below.

Another possible approach for data dissemination is report pushing. Report pushing is the dual problem of report pulling; reports are flooded, and consumed by peers whose query is answered by received reports. So far there exist mechanisms to broadcast information in the complete network, or in a specific geographic area (geocast), apart from to any one specific mobile node (unicast/mobile ad-hoc routing) or any one arbitrary node (anycast). Report pushing paradigm can be further

Pedestrians projects	**Features**
7DS – Columbia University [19] http://www.cs.unc.edu/ maria/7ds/	Focuses on accessing web pages in environments where only some peers have access to the fixed infrastructure.
iClouds – Darmstadt University http://iclouds.tk.informatik.tu-darmstadt.de/	Focuses on the provision of incentives to brokers (intermediaries) to participate in the mobile P2P database.
MoGATU – University of Maryland, Baltimore County http://mogatu.umbc.edu/	Focuses on the processing of complex data management operations, such as joins, in a collaborative fashion.
PeopleNet – National University of Singapore [16] http://www.ece.nus.edu.sg/research/projects/abstract.asp?Prj=101	Proposes the concept of information Bazaars, each of which specializes in a particular type of information; reports and queries are propagated to the appropriate bazaar by the fixed infrastructure.
MoB – University of Wisconsin and Cambridge University http://www.cs.wisc.edu/ suman/projects/agora/	Focuses on incentives and the sharing among peers of virtual information resources such as bandwidth.
Mobi-Dik – University of Illinois at Chicago http://www.cs.uic.edu/˜wolfson/html/p2p.html	Focuses on information representing physical resources, and proposes stateless algorithms for query processing, with particular concerns for power, bandwidth, and memory constraints.

divided into stateful methods and stateless methods. Most stateful methods are topology-based, i.e., they impose a structure of links in the network, and maintain states of data dissemination. PStree [9], which organizes the peers as a tree, is an example of topology based methods.

Another group of stateful methods is cluster- or hierarchy-based method, such as [20], in which moving peers are grouped into some clusters or hierarchies and the cluster heads are randomly selected. Reports are disseminated through the network in a cluster or hierarchy manner, which means that reports are first disseminated to every cluster head, and each cluster head then broadcasts the reports to the member peers in its group. Although cluster- or hierarchy-based methods can minimize the energy dissipation in moving peers, these methods will fail or cost more energy in highly mobile environments since they have to maintain a hierarchy structure and frequently re-select cluster heads.

Another stateful paradigm consists of Location-based methods (see [15]). In location-based methods, each moving peer knows the location of itself and its neighbors through some localization techniques, such as GPS or Atomic Multilateration (see [15]).

The simplest location-based data dissemination is Greedy Forwarding, in which each moving peer transmits a report to a neighbor that is closer to the destination than itself. However, Greedy Forwarding can fail in some cases, such as when a

Vehicular projects	Features
CarTALK 2000 – A European project http://www.cartalk2000.net/	Develops a co-operative driver assistance system based upon inter-vehicle communication and mobile P2P databases via self-organizing vehicular ad-hoc networks.
FleetNet – Internet on the Road Project http://www.ccrle.nec.de/Projects/fleetnet.htm	Develops a wireless multi-hop ad hoc network for intervehicle communication to improve the driver's and passenger's safety and comfort. A data dissemination method called "contention-based forwarding" (CBF) is proposed in which the next hop in the forwarding process is selected through a distributed contention mechanism based on the current positions of neighbors.
VII – Vehicle Infrastructure Integration, a US DOT project http://www.its.dot.gov/vii/	The objective of the VII project is to deploy advanced vehicle-to-vehicle (using the mobile P2P paradigm) and vehicle-to-infrastructure communications that could keep vehicles from leaving the road and enhance their safe movement through intersections.
Grassroots – Rutgers University [7] http://paul.rutgers.edu/ ∼gsamir/dataspace/grassroots.html	Develops an environment in which each vehicle contributes a small piece of traffic information to the network based on the P2P paradigm, and each vehicle aggregates pieces of the information into a useful picture of the local traffic information.

report is stuck in local minima, which means that the report stays in a mobile peer whose neighbors are all further from the destination. Therefore, some recovery strategies are proposed, such as GPSR (Greedy Perimeter Stateless Routing [12]). Other location-based methods, such as GAF (Geographic Adaptive Fidelity [24]) and GEAR (Geographical and Energy Aware Routing [28]), take advantage of knowledge about both location and energy to disseminate information and resources more efficiently.

In stateless methods, the most basic and simplest one is flooding-based method, such as [17]. In flooding-based methods, mobile peers simply propagate received reports to all neighboring mobile peers until the destination or maximum a hop is reached. Each report is propagated as soon as is received. Flooding-based methods have many advantages, such as no state maintenance, no route discovery, and easy deployment. However, they inherently cannot overcome several problems, such as implosion, overlap, and resource blindness. Implosion refers to the waste of resources taking place when a node forwards a message to a neighbor although the latter may have already received it from another source. Overlap occurs when two nodes read the same product record or coupon, and thus push into the network the same information. Resource blindness denotes the inability of the protocol to adapt the node's behaviour to its current availability of resources, mainly power [18]. Therefore, other stateless methods are proposed, such as gossiping-based methods and negotiation-based methods.

Gossiping-based methods, such as [6], improve flooding by transmitting received reports to a subset of randomly selected neighbors; another option is to have some neighbours simply drop the report. For example, the neighbors that are not themselves interested in the report drop it. The advantages of gossiping-based methods include reducing the implosion and lowering the system overhead. However, dissemination, and thus performance, is reduced compared to pure flooding.

Negotiation-based methods solve the implosion and overlap problem by transmitting first the id's of reports; the reports themselves are transmitted only when requested (see [13]). Thus, some extra data transmission is involved, which costs more memory, bandwidth, and energy. In addition, in negotiation-based methods, moving peers have to generate meta-data or a signature for every report so that negotiation can be carried out, which will increase the system overhead and decrease the efficiency.

Another important stateless paradigm for data dissemination in MANET databases is store-and-forward. In contrast to flooding, store-and-forward does not propagate reports as soon as they are received; rather they are stored and rebroadcast later. This obviously introduces storage and bandwidth problems, if too many reports need to be saved and rebroadcast at the same time . To address these, methods such as [22] rank all the reports in a peer's database in terms of their relevance (or expected utility), and then the reports are communicated and saved in the order of their relevance. Or, the reports requested and communicated are the ones with the relevance above a certain threshold. The notion of relevance quantifies the importance or the expected utility of a report to a peer at a particular time and at a particular location. Other store-and-forward methods include PeopleNet [16] and 7DS [19].

In summary, the paradigms for data dissemination in MANET databases are summarized in Fig. 2 below.

4 The MOBI-DIK Approach

Report pulling and stateful approaches suffer when the network topology is dynamic, disconnected, and/or sparse. Thus we propose a store-and-forward algorithm, MOBI-DIK, based on the following ideas and results:

1. In MOBI-DIK the growing-local-database problem of store-and-forward algorithms is addressed by prioritization; each mobile peer prioritizes the reports in order to accommodate them in limited power, bandwidth, and memory. The priority of a report depends on its size (the larger the report, the more resources it consumes), demand (how many peers are querying it), and supply (how many peers already have it). The demand of a report is estimated by sampling of the queries that it satisfies. But sampling does not work for estimating the supply, because supply increases continuously as the report is being disseminated. Thus,

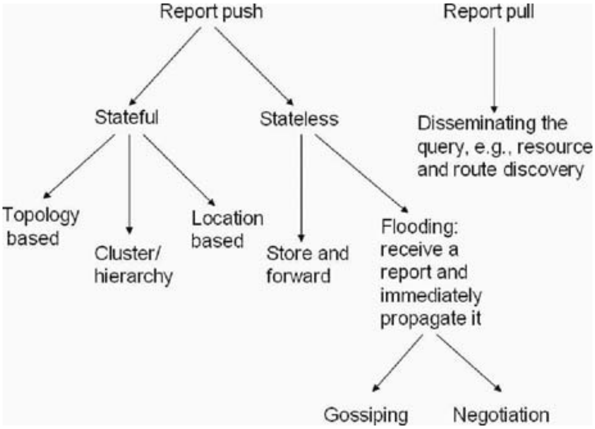


Fig. 2 Query answering methods in MANET databases

a. We examined estimating the supply of a report based on various indicators such as the age of the report and the number of times it is received by a mobile peer. We found that no single indicator is a good predictor of supply in all environments. For example, in some environments the intuition that the age of the dataitem is a good predictor of novelty is correct. However, in an environment where many new peers are entering the system, the number of times a report is received by a mobile peer is a much better indicator of supply.
b. We developed and implemented an algorithm called MALENA that combines various indicators to estimate the current supply [25]. The combination uses a machine learning system that is trained from previously received reports, and automatically learns the best indicator for the current environment.
c. We compared MALENA with three approaches to ranking of reports [25]. In the first two approaches, ranking is based on a single individual supply indicator that is found to be an optimal indicator in an environment that disfavors another. The third approach is PeopleNet [16]. The comparison is conducted by simulations using real Bluetooth traces collected at a major conference event. The comparison shows that in each individual environment MALENA approaches or outperforms the best algorithm for that environment and outperforms the inferior one by up to 5 times. This is important since the best algorithm depends on the global environment, and the global environmental parameters change and are usually not known to a mobile peer.
d. Simulations show that, with each mobile peer storing 1000 reports and allocating 10% of its power to MALENA, the performance of MALENA reaches 56% of the ideal benchmark (where a central server is employed, and it instantaneously broadcasts each new report to all the mobile peers). 30% of the reports received by MALENA are received within 10 minutes. We also calculated the CPU consumption of MALENA. On a typical PDA, each execution of the MALENA computation takes less than half a millisecond.

2. An additional issue arising in a Store-and-forward algorithm such as MOBI-DIK is how many reports to communicate in each transmission. If too many, excessive collisions arise, and if too few, then the search capability suffers. We developed an analytical model that computes the throughput of a transmission in an 802.11 ad hoc network (see [22]). The throughput is computed based on collision factors such as the transmission size, the transmission frequency, the density of mobile peers, and so on. Using this analytical model we proposed a method by which a mobile peer dynamically adjusts the P2P transmission size depending on the period of time between two P2P transmissions of a peer. The objective is to optimize the bandwidth and energy utilization; optimization occurs when the number of reliably received reports delivered per unit of energy is maximized. Simulations show that by dynamically adjusting the transmission size, the performance of the mobile P2P search is improved by up to an order of magnitude.
3. In MOBI-DIK a peer propagates reports when a new neighbor is encountered or when new reports are received (old reports to new neighbors, or new reports to old neighbors), thus adapting to both low and high mobility environments.
4. We compared MOBI-DIK with the three existing mobile P2P algorithms, namely RANDI [21], PeopleNet [16], and 7DS [19]. MOBI-DIK outperforms PeopleNet and 7DS by an order of magnitude, and is up to two times better than RANDI. More details are provided in Appendix A. The advantage of MOBI-DIK in a practical application, namely parking slot discovery, was demonstrated in [26].
5. We studied how the availability of a fixed infrastructure (e.g., the internet) can be exploited to augment the P2P data dissemination (absence of a central server is assumed). The general idea is that when a match (i.e., a query and a report that match each other) occurs at a broker, the broker sends the matching report to the query originator via the infrastructure. Since the cellular transmission is, per byte, 16 times more energy-costly than P2P transmission, it is not even clear that backchannel communication is beneficial. However, we determined that it is. More details are provided in Appendix A.
6. We demonstrated the MOBI-DIK algorithm in the context of specific queries and reports. The queries are K-nearest-neighbor queries, and the reports are the current locations of mobile sensors. Thus, the MOBI-DIK algorithm is specialized to process this specific spatial query in a data-to-query fashion. Similarly, it can be specialized for in-network processing of other types of queries, e.g. spatial window queries. We compare the MOBI-DIK continuous-KNN performance with that of the in-network KNN algorithm given in [23], called DIKNN. It processes queries in a query-to-data fashion. The results show that MOBI-DIK is up to 50 times more accurate than DIKNN when the P2P network is sparse, but DIKNN is more accurate when the network is dense. More details are provided in appendix B.
7. We studied the application of MOBI-DIK in the dissemination of real-time traffic information. In this application, the real-time traffic information is produced by and disseminated to vehicles throughout a road network. This application complements the existing real-time traffic information dissemination methods which tend to cover only selected highways where speed sensors are deployed. We

compared MOBI-DIK with Grassroots [7], a flooding based mobile P2P traffic information dissemination algorithm. The results show that MOBI-DIK outperforms Grassroots when the vehicle density is sparse or when the available bandwidth is small. In some cases MOBI-DIK outperforms Grassroots by 50%. These results demonstrate the benefit of store-and-forward, information prioritization, and bandwidth adaptation. More details are provided in Appendix C.

5 Key Applications

MANET databases provide mobile users a search engine for transient and highly dynamic information in a local geospatial environment. MANET databases employ a unified model for both the cellular infrastructure and the mobile ad hoc environments. When the infrastructure is available, it can be augmented by the MANET database approach.

Consider a MANET database platform, i.e., a set of software services for data management in a MANET environment; it is similar to a regular Database Management System, but geared to mobile P2P interactions. Such a platform will enable quick building of matchmaking or resource discovery services in many application domains, including social networks, emergency response and homeland security, the military, airport applications, mobile e-commerce, and transportation.

1. **Social Networks** In a large professional, political, or social gathering, MANET databases are useful to automatically facilitate a face-to-face meeting based on matching profiles. For example, in a professional gathering, MANET databases enable attendees to specify queries (interest profiles) and resource descriptions (expertise) to facilitate face-to-face meetings, when mutual interest is detected. Thus, the individual's profile that is stored in MANET databases will serve as a "wearable web-site". Similarly, MANET databases can facilitate face-to-face meetings for singles matchmaking.
2. **Emergency Response, Homeland Security, and the Military** MANET databases offer the capability to extend decision-making and coordination capability. Consider workers in disaster areas, soldiers and military personnel operating in environments where the wireless fixed infrastructure is significantly degraded or non-existent. As mobile users involved in an emergency response naturally cluster around the location of interest, a self-forming, high-bandwidth network that allows database search without the need of potentially compromised infrastructure could be of great benefit. For instance, the search could specify a picture of a wanted person.
3. **Airport Applications** A potential opportunity that will benefit both the consumer and the airport operations is the dissemination and querying of real-time information regarding flight changes, delays, queue length, parking information, special security alerts and procedures, and baggage information. This can augment the present audio announcements that often cannot be heard in nearby restaurants, stores, or restrooms, and augment the limited number of displays.

4. **Mobile E-commerce** Consider short-range wireless broadcast and mobile P2P dissemination of a merchant's sale and inventory information. It will enable a customer (whose cell phone is query-capable) that enters a mall to locate a desired product at the best price. When a significant percentage of people have mobile devices that can query retail data, merchants will be motivated to provide inventory/sale/coupons information electronically to nearby potential customers The information will be disseminated and queried in a P2P fashion (in, say, a mall or airport) by the MANET database software.
5. **Transportation Safety and Efficiency** MANET databases software can improve safety and mobility by enabling travelers to cooperate intelligently and automatically. A vehicle will be able to automatically and transitively communicate to trailing vehicles its "slow speed" message when it encounters an accident, congestion, or dangerous road surface conditions. This will allow other drivers to make decisions such as finding alternative roads. Also, early warning messages may allow a following vehicle to anticipate sudden braking, or a malfunctioning brake light, and thus prevent pile-ups in some situations. Similarly, other resource information, such as ridesharing opportunities, transfer protection (transfer bus requested to wait for passengers), will be propagated transitively, improving the efficiency of the transportation system.

6 Future Research Directions

Further work is necessary on data models for mobile P2P search applications. Work on sensor databases (e.g. Tinydb [14]) addresses data-models and languages for sensors, but considers query processing in an environment of static peers (see e.g., POS [4]). Cartel [10] addresses the translation of these abstractions to an environment in which cars transfer collected data to a central database via fixed access points. Work on MANET protocols deals mainly with routing and multicasting. In this landscape there is a gap, namely general query-processing in MANET's; such processing needs to be cognizant of many issues related to peer-mobility. For example, existing mobile P2P query processing methods deal with simple queries, e.g., selections; each query is satisfied by one or more reports. However, in many application classes one may be interested in more sophisticated queries, e.g., aggregation. For instance, in mobile electronic commerce a user may be interested in the minimum gas price within the next 30 miles on the highway. Processing of such P2P queries may present interesting optimization opportunities.

After information about a mobile resource is found, localization is often critical for finding the physical resource. However, existing (self)-localization techniques are insufficient. For example, GPS is not available indoors and the accuracy of GPS is not reliable. Thus, furthering the state of the art on localization is important for mobile P2P search.

As discussed above, MANET databases do not guarantee answer completeness. In this sense, the integration with an available infra-structure such as the internet or a cellular network may improve performance significantly. This integration has two aspects. First, using the communication infrastructure in order to process queries more efficiently; and second, using data on the fixed network in order to provide better and more answers to a query. The seamless integration of MANET databases and infrastructure databases introduces important research challenges.

Other important research directions include: incentives for broker participation in query processing (see [27]), and transactions/atomicity/recovery issues in databases distributed over mobile peers (virtual currency must be transferred from one peer to another in an atomic fashion, otherwise may be lost).

Of course, work on efficient resource utilization in mobile peers, and coping with sparse networks and dynamic topologies is still very important for mobile P2P search.

References

1. M. Barbaro and T. Zeller. A Face Is Exposed for AOL Searcher No. 4417749, New York Times, August 9, 2008. http://www.nytimes.com/2006/08/09/technology/09aol.html?_r=1&scp=11&sq=google
2. R. Barr, "An Efficient, Unifying Approach to Simulation Using Virtual Machines", PhD thesis, Cornell University, 2004.
3. D. Choffnes and F. Bustamante, "An Integrated Mobility and Traffic Model for Vehicular Wireless Networks", VANET, 2005.
4. L. Cox, M. Castro, and A. Rowstron, "POS: Practical Order Statistics for Wireless Sensor Networks", 26th IEEE ICDCS, Lisbon, Portugal, July 2006.
5. S. M. Das, H. Pucha, and Y. C. Hu, "Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad hoc Networks", in Proc. of the 6th IEEE Workshop on Mobile Computing Systems and Applications, English Lake District, UK, 12 2004.
6. A. Datta, S. Quarteroni, and K. Aberer, "Autonomous Gossiping: A Self-Organizing Epidemic Algorithm For Selective Information Dissemination in Wireless Mobile Ad-Hoc Networks," in The International Conference on Semantics of a Networked World, 2004.
7. S. Goel, T. Imielinski, K. Ozbay, "Ascertaining Viability of WiFi based Vehicle-To-Vehicle Network for Traffic Information Dissemination," The 7th Intel. IEEE Conf. on Intelligent Transportation Systems, 2004.
8. S. Hansell. AOL Removes Search Data On Vast Group Of Web Users, New York Times, August 8, 2008. http://query.nytimes.com/gst/fullpage.html?res=9504E5D81E3FF93BA3575BC0A9609C8B63
9. Y. Huang and H.G. Molina, "Publish/subscribe in a Mobile Environment", MobiDE, 2001.
10. B. Hull, et al., "CarTel: A Distributed Mobile Sensor Computing System", In Proc. of 4th Intl. Conf. on Embedded Networked Sensor Systems, pp. 125–138, Nov 2006.
11. B. Karp and H. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In Proc. of MOBICOM, 2000.
12. B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Sensor Networks," in The 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00), pp. 243–254, Aug 2000.
13. J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks," Wireless Networks, vol. 8, pp. 169–185, 2002.
14. S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an Acquisitional Query Processing System for Sensor Networks", ACM Transactions on Database System, 30(1):122–173, 2005.

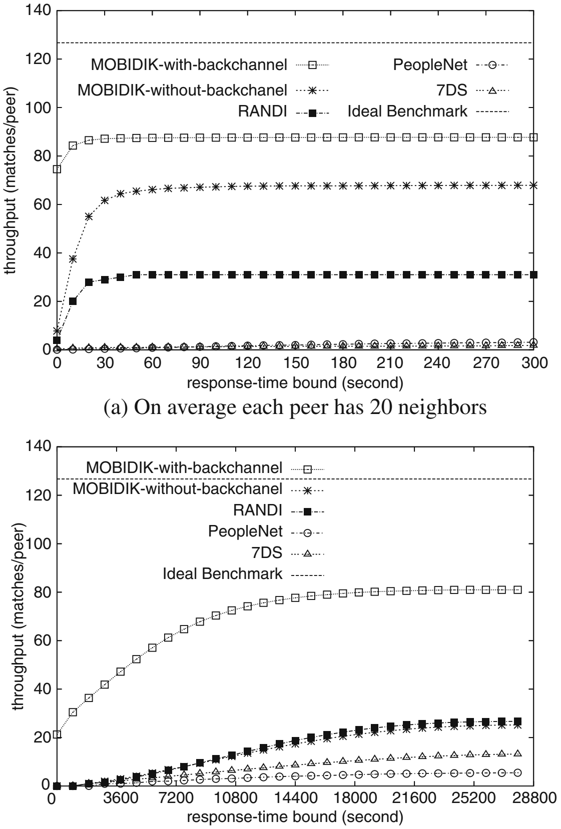
15. M. Mauve, A. Widmer, and H. Hartenstein, “A Survey on Position-Based Routing in Mobile Ad-Hoc Networks,” IEEE Network, vol. 15, no. 6, pp. 30–39, 2001.
16. M. Motani, V. Srinivasan, and P. Nuggehalli, “PeopleNet: Engineering a Wireless Virtual Social Network,” International Conference Mobile Computing and Networking (MobiCom’05), Aug 2005.
17. R. Oliveira, L. Bernardo, and P. Pinto, “Flooding Techniques for Resource Discovery on High Mobility MANETs”, Workshop on Wireless Ad-hoc Networks, 2005.
18. A. A. Papadopoulos and J. A. McCann, “Towards the Design of an Energy-Efficient, Location-Aware Routing Protocol for Mobile, Ad-hoc Sensor Networks”, in Proc. of the 15th Int. Workshop on Database and Expert Systems Applications (DEXA), 2004.
19. M. Papadopouli and H. Schulzrinne, “Design and Implementation of a P2P Data Dissemination and Prefetching Tool for Mobile Users”, First NY Metro Area Networking Workshop, IBM TJ Watson Research Center, Hawthorne, NY, 2001.
20. A. Visvanathan, J. H. Youn, and J. Deogun, “Hierarchical Data Dissemination Scheme for Large Scale Sensor Networks,” in IEEE International Conference on Communications (ICC’05), pp. 3030–3036, May 2005.
21. O. Wolfson and B. Xu, “Mobile Peer-to-peer Data Dissemination with Resource Constraints”, MDM, 2007.
22. O. Wolfson, B. Xu, H.B. Yin, and H. Cao, “Search-and-Discover in Mobile P2P Network Databases”, in Proc. of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS’06), Lisbon, Portugal, 2006
23. S.-H. Wu et al: Diknn: an itinerary-based knn query processing algorithm for mobile sensor networks. ICDE07.
24. Y. Xu, J. Heidemann, and D. Estrin, “Geography-informed Energy Conservation for Ad hoc Routing,” in The ACM International Conference on Mobile Computing and Networking, pp. 70–84, Rome, Italy, July 2001.
25. B. Xu, O. Wolfson, and C. Naiman, “Flooding by Machine Learning in Mobile Ad Hoc Networks”, accepted, to appear in ACM Transactions on Autonomous and Adaptive Systems (TAAS).
26. B. Xu, O. Wolfson, C. Naiman, N. Rishe, and R. Tanner, “A Feasibility Study on Disseminating Spatio-temporal Information via Vehicular Ad-hoc Networks”, V2VCOM, 2007.
27. B. Xu, O. Wolfson, and N. Rishe, “Benefit and Pricing of Spatio-temporal Information in Mobile Peer-to-Peer Networks”, Proc. of Hawaii Intl. Conf. on System Sciences (HICSS-39), Jan 2006.
28. Y. Yu, R. Govindan, and D. Estrin., “Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks,” Technical Report UCLA/CSD-TR-01-0023, UCLA, May 2001.

Appendix A. Comparison of MOBI-DIK with RANDI, PeopleNet, and 7DS

From the algorithmic point of view, the main differences of 7DS and PeopleNet from MOBI-DIK are: (1) they have no energy/bandwidth management for determining the transmission size; (2) the broker function is much more simplistic (no ranking); (3) 7DS does not have a good strategy to determine when to communicate. The main difference of RANDI from MOBI-DIK is that it does not take supply into account when ranking reports. The four algorithms are implemented in SWANS (Scalable Wireless Ad-hoc Network Simulator) built at Cornell [2]. We augmented

SWANS with a feature that tracks the energy consumed for computation and for 802.11 communication (including transmitting, receiving, and listening). In order to put the performance of the algorithms in perspective, we also define an ideal benchmark offline algorithm. In the ideal benchmark algorithm, when a report is produced, it is instantaneously disseminated to all the mobile peers currently in the system.

Figure 3 shows some simulation results. The performance measure, response-time bounded throughput, was developed by us and it integrates two traditional evaluation metrics for data access, namely throughput and response-time. Specifically, the response-time bounded throughput is the average number of distinct matches received by a mobile peer within a certain time limit *c*. *c* is called the response-time bound. The following observations are made from the simulation results:



(a) On average each peer has 20 neighbors

(b)On average each peer has 0.1 neighbor

Fig. 3 Comparison among MOBI-DIK, RANDI, PeopleNet, and 7DS. 802.11 bandwidth=2 Mbps, transmission range=100 meters, 15% of battery energy allocated to a mobile P2P algorithm, 1 report produced every 10 seconds, mean of report size=1500 bytes, query size = 100 bytes, mean of reports database size=100K bytes

1. With at least 15% of the battery energy allocated to each mobile peer, a mobile P2P database performs reasonably close to an ideal centralized database.
2. MOBI-DIK outperforms PeopleNet and 7DS by an order of magnitude.
3. Supply-and-demand ranking doubles the performance of a mobile P2P database (see Fig. 3 a).
4. The backchannel improves the performance of MOBI-DIK by nearly 30% even though it is more energy costly.
5. The backchannel is particularly beneficial when the peer density is low.

Appendix B: Query-to-Data and Data-to-Query in P2P KNN-Query Processing

In this appendix we demonstrate the MOBI-DIK algorithm in the context of K-nearest-neighbors (KNN) queries, where the reports are the current locations of mobile sensors. Thus, the MOBI-DIK algorithm is specialized to process this specific spatial query. We compare the MOBI-DIK continuous-KNN performance with that of the in-network KNN algorithm given in [23] (called DIKNN). This section is organized as follows. In ‡B.1 we discuss the instantiation of MOBI-DIK for KNN query processing. In ‡B.2 we compare MOBI-DIK and the DIKNN algorithm [23] which is designed especially for KNN query processing in mobile sensor networks.

B.1 KNN Query Processing in MOBI-DIK

We consider a system where there is a single static sink. The sink is continuously interested in knowing the k mobile peers that are closest to a static query point q. Thus the sink and every other peer have a query Q which contains the coordinates of q. Q is known to all the peers before the KNN query processing and therefore it is not transmitted during a P2P interaction. A report is produced by each mobile peer O every one second. The report, referred to as the location report, contains the current timestamp, the current location of O, and the peer-id of O. We refer to the location included in a report R as the home location of R.

For the computation of the demand, $Q(R)$, i.e., the degree to which a report R satisfies the KNN query Q is negatively correlated to the distance between the home location of R and the query point q, and is $[0-1]$ normalized. For example, if the distance between the home location of R and q is 1.4 mile, and the maximum distance between q and the home location of a report in the peer's database is 2 miles, then the demand for R is $(1-1.4/2=0.3)$.

The reports propagate via P2P to the sink, which is one of the peers. The sink answers the KNN queries based on the latest location of each peer maintained in the sink's reports database.

B.2 Comparison Between Query-to-Data by DIKNN, and Data-to-Query by MOBI-DIK

The DIKNN Algorithm. In the literature there is no algorithm for in-network processing of continuous KNN queries [23]. Thus we use an instantaneous algorithm, DIKNN, and execute it repeatedly. In DIKNN, the query issued by the sink is geographically routed from the sink to the nearest neighbor of the query point p by GPSR [11]. The query is then disseminated to all the peers in a circular searching area centered at p. The size of the searching area is determined based on the density of the network, such that the k nearest neighbors are inside the searching area with a high probability. At the end of the dissemination, the aggregated query response is routed back to the sink. Simulation Method. The mobile P2P algorithms are implemented in SWANS (Scalable Wireless Ad-hoc Network Simulator) built at Cornell. 100 peers move within a $L \times L$ square area according to the *random way-point* mobility model with mean speed 1 mile/hour and mean pause time 180 seconds. In this setup, the average number of peers D within a circle with radius 100 meters (i.e., the transmission range), is

$$D = 100\frac{\pi \times 100 \times 100}{L^2}$$

We refer to D as the peer density. D is a system parameter. It has values 20 and 1, representing high and low connectivities. In other words, even though MOBI-DIK is geared towards a sparse connectivity environment, we evaluate it in dense ones as well. L is chosen such that the peer density equals to the value of D. The whole simulation runs for 8 simulated hours.

The sink is located at the center of the northwest quadrant of the simulated square area. The query point is located at the center of the southeast quadrant. The query is issued every one second. For MOBI-DIK, this means that KNN's are computed at the sink using the local database every 1 second. The local database has the last known location of each peer, but some of these locations may be outdated, thus the computed KNN may not be accurate. The accuracy of each KNN query is measured by the fraction of the correctly returned KNN's.

The size of each report is 24 bytes, including 16 bytes for the coordinates of the location, 4 bytes for the timestamp, and 4 bytes for the peer-id. The size limit of each reports database is randomly chosen from [1200, 3600] bytes. Thus the average number of members in a reports database is $2400K/24 = 100$. All the simulation parameters and their values are listed in Table B.1.

For DIKNN, we do not simulate its operation. Instead, we compute the upper bound of its query accuracy. Assume that the geographic routing is reliable and instantaneous, and the searching area contains all the K-nearest neighbors, thus the query is correctly delivered to them. Then with DIKNN, a correct KNN is returned to the sink if and only if there exists a path between this KNN and the sink at the time when the query is issued. For each query, we compute the fraction of the correct

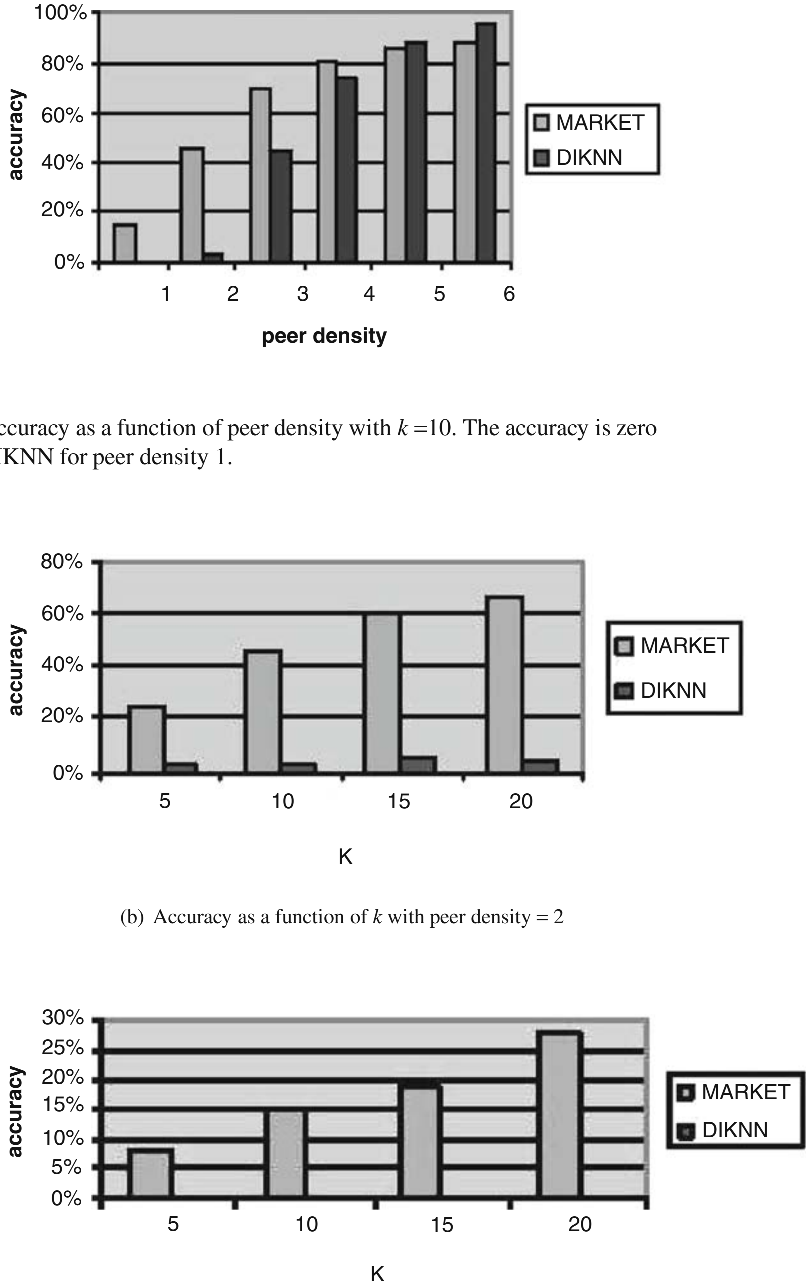
Table B.1 Simulation parameters and their values

Parameter		Unit	Values
Number of concurrent peers			100
Total number of peers			100
Transmission range	*r*	Meter	100
Peer density	*D*	Number of peers in a circle with radius *r*	1, 2, 3, 4, 5, 6, 7
Data transmission speed		Bits/second	2M
Battery allocation fraction	*F*	1	
Report production rate	*P*	Reports/second/peer	1
Report size range		Byte	24
Query size		Byte	16
Reports database size range		Byte	[1200, 3600]
Demand database size		Byte	16
Average motion speed		Miles/hour	1
Average pause time		Second	180
Mobility model			Random way-point
Simulation run time		Hours	8

KNN's for which there are paths between them and the sink, and take this fraction to be the accuracy of DIKNN.

The above simulation method favors MOBI-DIK in the sense that MOBI-DIK is designed for continuous queries and DIKNN is designed for instantaneous queries. On the other hand, it favors DIKNN in the following senses. First, DIKNN is optimized for KNN query processing whereas MOBI-DIK is a general purpose query processing algorithm. Second, MOBI-DIK is compared with the upper bound accuracy of DIKNN.

Simulation results. Figure 4 a shows the accuracy of MOBI-DIK and DIKNN when k is 10, for different peer densities. Figure 4 b, c show the accuracy when peer density is 2 and 1 respectively, for different values of k. From these figures it can be seen that the accuracy of MOBI-DIK is by far higher than that of DIKNN when the peer



(a) Accuracy as a function of peer density with k =10. The accuracy is zero for DIKNN for peer density 1.

(b) Accuracy as a function of k with peer density = 2

(c) Accuracy as a function of k with peer density =1. The accuracy is zero for DIKNN for all the k values.

Fig. 4 Accuracy

density is low. When the peer density is below 2, the DIKNN algorithm completely fails, with 0% accuracy. In this case MOBI-DIK manages to provide up to 30% accuracy. When the peer density is 2, MOBI-DIK outperforms DIKNN by an order of magnitude, regardless of the value of k. In this case the accuracy of MOBI-DIK reaches 60% for $k = 20$. The reason for the poor performance of DIKNN in a sparse network is that, with very high probability there does not exist a path between the sink and a KNN. When the sink and the KNN are disconnected, the query does not reach the KNN and the KNN is not collected in the query response. MOBI-DIK, on the other hand, does not need a contemporaneous path to collect results. This demonstrates the benefit of store-and-forward in a sparse network.

When the peer density is higher than 5 (see Fig. 4 a), DIKNN starts to outperform MOBI-DIK.

Appendix C. Comparison of MOBI-DIK with Grassroots in Dissemination of Real-time Traffic Information

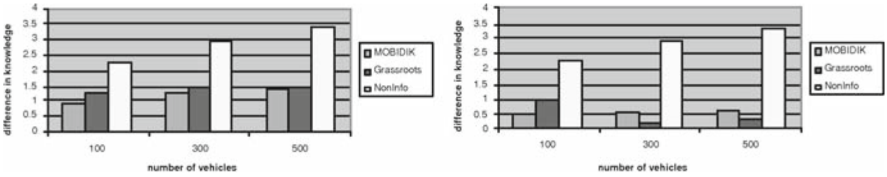
The application scenario is as follows. A set of vehicles move on a road network. Each vehicle m has a *digital map* of the road network. For each road segment s in the road network, the digital map stores the estimated travel time of s. Each vehicle m is equipped with a GPS receiver. Every time m travels through a road segment s and reaches the end of it, m produces a report that includes the travel time experienced by m on s. The report is disseminated using a mobile P2P algorithm (MOBI-DIK or Grassroots). Upon receiving a report, m updates the estimated travel time of the corresponding road segment in its own digital map.

We propose a novel metric, called the difference in knowledge (DIK), as the performance measure. The DIK measures the difference between the actual traffic condition on a road segment and that known to a vehicle. Formally, let $s_1, s_2, \ldots, s_n$ be all the road segments in the road network. Let p be the location of a vehicle O at t. Let be the actual travel time of a road segment s_k. Let $T(O, s_k)$ be the travel time of s_k maintained at O's digital map. The difference in knowledge of O at t, denoted $DIK(O,t)$ is

$$DIK(O,t) = \sum_{k=1} \left(\frac{1}{g_k} \left| T(s_k) - T(o, s_k) \right| \right) \tag{1}$$

where g_k is the free-flow travel time along the shortest-distance path from p to the middle point of sk. Intuitively, the difference in knowledge of O is the weighted sum of the difference in knowledge of O between the true map and its local digital map for each road segment. The farther away the road segment is from O, the lower its weight. Specifically, the weight of a road segment s_k is $1/g_k$. To put the results in perspective we evaluated MOBI-DIK and Grassroots against the case in which reports are not disseminated at all. By comparing with this case, called Non-Info, we evaluate the benefit of reports dissemination. The comparison uses the

STRAW/SWANS simulation test-bed [3]. The STRAW system simulates vehicle traffic mobility on a 3.2 km × 2.2 km region of downtown Chicago.



(a) Bandwidth allocation = 1%

(b) Bandwidth allocation = 100%

Fig. 5 Comparison among MOBI-DIK, Grassroots, and NonInfo. 802.11 band-width=2 Mbps, transmission range=250 meters

Figure 5 shows some simulation results. From the figure it can be seen that both MOBI-DIK and Grassroots significantly outperform NonInfo, by up to an order of magnitude. Furthermore, when the vehicle density is small, MOBI-DIK outperforms Grassroots; this demonstrates the advantage of store-and-forward. When a small fraction of 802.11 bandwidth is allocated to traffic information dissemination (the rest may be reserved for emergent dissemination of safety-related messages such as a malfunctioning brake or a deployment of an airbag), MOB-DIK also outperforms Grassroots; this demonstrates the benefit of information prioritization and bandwidth adaptation. Grassroots outperforms MOBI-DIK when the bandwidth allocation and vehicle density are high, and therefore we are defining a system that adaptively transitions between the two algorithms depending on these parameters.

Part X
Fault Tolerance in P2P Networks

Managing Network Partitions in Structured P2P Networks

Tallat M. Shafaat, Ali Ghodsi, and Seif Haridi

Abstract Structured overlay networks form a major class of peer-to-peer systems, which are touted for their abilities to scale, tolerate failures, and self-manage. Any long-lived Internet-scale distributed system is destined to face network partitions. Consequently, the problem of network partitions and mergers is highly related to fault-tolerance and self-management in large-scale systems. This makes it a crucial requirement for building any structured peer-to-peer systems to be resilient to network partitions. Structured overlays have mainly been studied under churn (frequent joins/failures), which as a side effect solves the problem of network partitions, as it is similar to massive node failures. Yet, the crucial aspect of network mergers has been ignored. In this chapter, we motivate the problem of network partitions and mergers in structured overlays. We discuss how a structured overlay can automatically detect a network partition and merger. We present an algorithm for merging multiple similar ring-based overlays when the underlying network merges. We evaluate the algorithm for various scenarios and show that even when falsely detecting a merger, the algorithm quickly terminates and does not clutter the network with many messages. The algorithm is flexible as the tradeoff between message complexity and time complexity can be adjusted by a parameter.

Tallat M. Shafaat
Royal Institute of Technology (KTH), Electrum 229, 164 40 Kista, Sweden,
`http://www.ict.kth.se/~tallat`,
e-mail: `tallat@kth.se`

Ali Ghodsi
Swedish Institute of Computer Science (SICS), Box 1263, 164 29 Kista, Sweden,
`http://www.sics.se/~ali`,
e-mail: `ali@sics.se`

Seif Haridi
Royal Institute of Technology (KTH), Electrum 229, 164 40 Kista, Sweden,
`http://www.ict.kth.se/~haridi`,
e-mail: `haridi@kth.se`

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_40, © Springer Science+Business Media, LLC 2010

1 Introduction

Structured overlay networks and distributed hash-tables (DHTs) are touted for their ability to provide scalability, fault-tolerance, and self-management, making them well-suited for Internet-scale distributed applications. As these applications are long lived, they will always come across network partitions. Since overlays[1] have been touted for their ability to be fault-tolerant and self-manage, they have to be resilient to network partitions.

Although network partitions are not very common, they do occur. Internet failures, resulting in partitioned networks can occur due to large area link failure, router failure, physical damage to a link/router, router misconfiguration and buggy software updates. Overloaded routers, network wide congestion due to denial of service (DoS) attacks and routing loops [27] can also have the same effect as a network partition. Similarly, natural disasters can result in Internet failures. This was observed when an earthquake in Taiwan in December 2006 exposed the issue that global traffic passes through a small number of seismically active "choke points" [32]. Several countries in the region connect to the outside world through these choke points. A number of similar events leading to Internet failures have occurred [13]. On a smaller scale, the aforementioned causes can disconnect an entire organization from the Internet [26], thus partitioning the organization. Apart from physical failures, DoS (denial of service) might control enough nodes to effectively partition the overlay.

Infact, while deploying an overlay-based application on Internet, the first major problem reported and strongly suggested to be solved by Mislov et. al. [24] was:

A reliable decentralized system must tolerate network partitions.

While deploying ePOST [24] on PlanetLab, Mislov et. al. recorded the number of partitions experienced over a period of 85 days. Figure 1 shows their results, which clearly suggests that partitions occur all the time.

Apart from network partitions, the problem of merging multiple overlays is very interesting and useful in itself. It is common to build independent overlays regardless of existing overlays, and later merge them. This helps in decentralized bootstrapping of overlays [6]. Also, it may happen that overlays originally build independently have to be later merged due to overlapping interests.

Consequently, a crucial requirement for practical overlays is that they should be able to deal with network partitions and mergers. Most overlays cope with network partitions, but not with network mergers. This is because a network partition, as seen from the perspective of a single node, is identical to massive node failures. Since overlays have been designed to cope with churn (node joins and failures), they can self-manage in the presence of such partitions. However, most overlays cannot cope with network mergers, which is the focus of this chapter.

[1] In this chapter, we use the word overlay for structured overlay network.

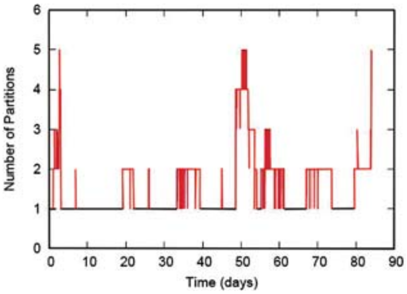


Fig. 1 Number of connected components observed over a 85 day period on PlanetLab. Figure taken from ePOST [24]

The merging of overlays gives rise to problems on two different levels: *routing level* and *data level*. The routing level is concerned with healing of the routing information after a partition merger. The data level is concerned with the consistency of the data items stored in the overlay. In this chapter, we focus mainly on the problem of dealing with partition mergers at the routing level, with a short discussion of issues faced on the data level.

Chapter Organization

Section 2 serves as a background for this chapter. A reader already familiar with a ring-based overlay such as Chord can safely skip this section. Section 3 shows how an overlay can detect a network partition and later, merger, in the underlying network, thus triggering the overlay merger algorithm. Section 4 introduces solutions on the routing level for merger of multiple ring-based overlays. Thereafter, Section 5 evaluates different aspects of the introduced algorithms. Section 6 discusses related work, including solutions for tree-based overlays. Finally, Section 7 summarizes the chapter and presents some open challenges currently faced by the research community.

2 Preliminaries

The rest of the chapter focuses on ring-based structured overlay networks. Next, we motivate this choice, and thereafter briefly define ring-based overlays. Finally, we show how Chord deals with network partitions and failures.

Motivation for the Unidirectional Ring Geometry

We confine ourselves to unidirectional ring-based SONs, such as Chord [31], SkipNet [12], DKS [9], Koorde [16], Mercury [4], Symphony [23], EpiChord [17], and Accordion [19]. But we believe that our algorithms can be adapted easily to other ring-based overlays, such as Pastry [28]. For a more detailed account on directionality and structure in SONs, please refer to Onana et al. [3] and Aberer et al. [1].

The reason for confining ourselves to ring-based overlays is twofold. First, ring-based overlays constitute a majority of the overlays. Second, Gummadi et al. [11] diligently compared the geometries of different overlays, and showed that the ring geometry is the one most resilient to failures, while it is just as good as the other geometries when it comes to proximity.

To simplify the presentation of our algorithms, we use notation that indicates the use of the Chord [31] overlay. But the ideas are directly applicable to all unidirectional ring-based SONs.

A Model of a Ring-based Overlay

A ring-based overlay makes use of an *identifier space*, which for our purposes is defined as a set of integers $\{0, 1, \cdots, \mathscr{N} - 1\}$, where $\mathscr{N}$ is some apriori fixed, large, and globally known integer. This identifier space is perceived as a ring that wraps around at $\mathscr{N} - 1$.

Every node in the system, has a unique identifier from the identifier space. Node identifiers are typically assumed to be uniformly distributed on the identifier space. Each node keeps a pointer, *succ*, to its *successor* on the ring. The successor of a node with identifier p is the first node found going in clockwise direction on the ring starting at p. Similarly, every node also has a pointer, *pred*, to its *predecessor* on the ring. The predecessor of a node with identifier q is the first node met going in anti-clockwise direction on the ring starting at q. A *successor-list* is also maintained at every node r, which consists of r's c immediate successors, where c is typically set to $\log_2(n)$, where n is the network size.

Ring-based overlays also maintain additional routing pointers on top of the ring to enhance routing. To be concrete, assume that these are placed as in Chord. Hence, each node p keeps a pointer to the successor of the identifier $p + 2^i \pmod{\mathscr{N}}$ for $0 \leq i < \log_2(\mathscr{N})$. The algorithms presented can easily be adapted to other schemes for placing these additional pointers.

Dealing with Partitions and Failures in Chord

Chord handles joins and leaves using a protocol called *periodic stabilization*. Leaves are handled by having each node periodically check whether *pred* is alive, and setting *pred* := *nil* if it is found dead. Moreover, each node periodically checks to see if *succ* is alive. If it is found to be dead, it is replaced by the closest alive successor in the successor-list.

Joins are also handled periodically. A joining node makes a lookup to find its successor s on the ring, and sets $succ := s$. Each node periodically asks for its successor's *pred* pointer, and updates *succ* if it finds a closer successor. Thereafter, the node notifies its current *succ* about its own existence, such that the successor can update its *pred* pointer if it finds that the notifying node is a closer predecessor than *pred*. Hence, any joining node is eventually properly incorporated into the ring.

As mentioned previously, a single node cannot distinguish massive simultaneous node failures from a network partition. As periodic stabilization can handle massive failures [20], it also recovers from network partitions, making each component of the partition eventually form its own ring. We have simulated such scenarios and confirmed these results. The problem that remains unsolved, which is the focus of the rest of the paper, is how several independent rings can efficiently be merged.

3 Detecting Network Partitions and Mergers

For multiple overlays to be merged, at least one node needs to have knowledge about at least one node in another ring. This is facilitated by the use of *passive lists*. Whenever a node detects that another node has failed, it puts the failed node, with the failed node's routing information[2] into its passive list. Every node periodically pings nodes in its passive list to detect if a failed node is again alive. When this occurs, it starts an overlay merging algorithm. Hence, a network partition will result in many nodes being placed in passive lists. When the underlying network merges, this will be detected and rectified through the execution of an overlay merging algorithm.

An overlay merging algorithm can also be invoked in other ways than described above. For example, it could occur that two overlays are created independently of each other, but later their administrators decide to merge them due to overlapping interests. It could also be that a network partition has lasted so long, that all nodes in the rings have been replaced, making the contents of the passive lists useless. In cases such as these, a system administrator can manually insert an alive node from another overlay into the passive list of any of the nodes. The overlay merger algorithm will take care of the rest.

The detection of an alive node in a passive list does not necessarily indicate the merger of a partition. It might be the case that a single node is incorrectly detected as failed due to a premature timeout of a failure detector. The overlay merging algorithm should be able to cope with this by trying to ensure that such false-positives will terminate the algorithm quickly. It might also be the case that a previously failed node rejoins the network, or that a node with the same overlay and network address as a previously failed node joins the ring. Such cases are dealt with by associating with every node a globally unique random *nonce*, which is generated each time a

[2] By routing information we mean a node's overlay identifier, network address, and nonce value (explained shortly).

node joins the network. Hence, if the algorithm detects that a node in its passive list is again alive, it can compare the node's current nonce value with that in the passive list to avoid a false-positive, as that node is likely a different node that coincidentally has the same overlay and network address.

4 Merging the Overlays

This section presents both simple and gossip-based Ring Unification [29]. Since the latter algorithm builds on the previous, we hope that this has a didactic value. In this section, we use Chord notation to simplify the presentation of the algorithms, though the ideas are directly applicable to all unidirectional ring-based overlays [1].

4.1 Simple Ring Unification

In this section, the simple ring unification algorithm (Algorithm 25) is presented. As shown later, the algorithm will merge the rings in $O(N)$ time for a network size of N. Though the problem of dealing with network mergers is crucial, such events happen rarely. Hence, it might be justifiable in certain application scenarios that a slow paced algorithm runs in the background, consuming little resources, while ensuring that any potential problems with partitions will eventually be rectified. Later, it is shown how the algorithm can be improved to make it complete the merger in substantially less time.

Algorithm 25 makes use of a queue called *detqueue*, which will contain any alive nodes found in the passive list (Section 3). The queue is periodically checked by every node p, and if it is non-empty, the first node q in the list is picked to start a ring merger. Ideally, p and q will be on two different rings. But even so, the distance between p and q on the identifier space might be very large, as the passive list can contain any previously failed node. Hence, the event MLOOKUP(id) is used to get closer to id through a lookup. Once MLOOKUP(id) gets near its destination id, it triggers the event TRYMERGE(a,b), which tries to do the actual merging by updating *succ* and *pred* pointers to a and b.

The event MLOOKUP(id) is similar to a Chord lookup, which tries to do a greedy search towards the destination id. One difference is that it terminates the lookup if it reaches the destination and locally finds that it cannot merge the rings. More precisely, this happens if MLOOKUP(id) is executed at id itself, or at a node whose successor is id. If an MLOOKUP(id) executed at n finds that id is between n and n's successor, it terminates the MLOOKUP and starts merging the rings by calling TRYMERGE. Another difference between MLOOKUP and an ordinary Chord lookup is that an MLOOKUP(id) executed at n also terminates and starts merging the rings if it finds that id is between n's predecessor and n. Thus, the merge will proceed in both clockwise and anti-clockwise direction.

Algorithm 25: Simple Ring Unification Algorithm

```
every $\gamma$ time units and $detqueue \neq \emptyset$ at $p$
    $q := detqueue$.dequeue()
    sendto $p$ : MLOOKUP($q$)
    sendto $q$ : MLOOKUP($p$)
end event

receipt of MLOOKUP($id$) from $m$ at $n$
    if $id \neq n$ and $id \neq succ$ then
        if $id \in (n, succ)$ then
            sendto $id$ : TRYMERGE($n, succ$)
        else if $id \in (pred, n)$ then
            sendto $id$ : TRYMERGE($pred, n$)
        else
            sendto closestprecedingnode($id$) : MLOOKUP($id$)
        end if
    end if
end event

receipt of TRYMERGE($cpred, csucc$) from $m$ at $n$
    sendto $n$ : MLOOKUP($csucc$)
    if $csucc \in (n, succ)$ then
        $succ := csucc$
    end if
    sendto $n$ : MLOOKUP($cpred$)
    if $cpred \in (pred, n)$ then
        $pred := cpred$
    end if
end event
```

The event TRYMERGE takes a candidate predecessor, *cpred*, and a candidate successor *csucc*, and attempts to update the current node's *pred* and *succ* pointers. It also makes two recursive calls to MLOOKUP, one towards *cpred*, and one towards *csucc*. These recursive calls attempt to continue the merging in both directions. Figure 2 shows the working of the algorithm.

In summary, MLOOKUP closes in on the target area where a potential merger can happen, and TRYMERGE attempts to do local merging and advancing the merge process in both directions by triggering new MLOOKUPs.

4.2 Gossip-Based Ring Unification

The simple ring unification presented in the previous section has two disadvantages. First, it is slow, as it takes $O(N)$ time to complete the ring unification. Second, it cannot recover from certain pathological scenarios. For example , assume two distinct rings in which every node points to its successor and predecessor in its own ring. Assume furthermore that the additional pointers of every node point to nodes

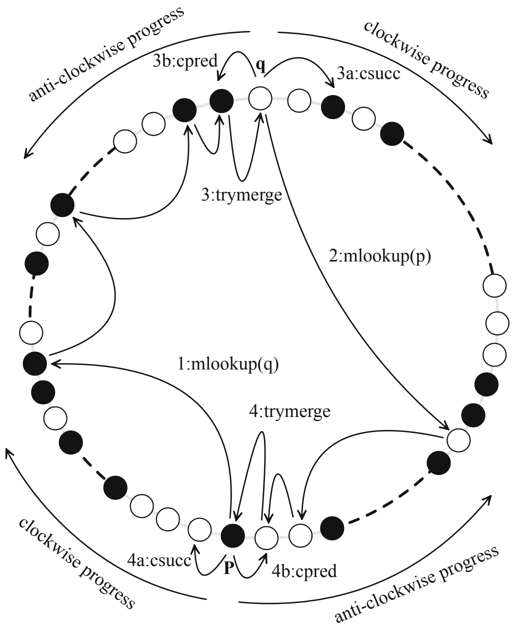


Fig. 2 Filled circles belong to Overlay1 and empty circles belong to Overlay2. The algorithm starts when p detects q, p makes an MLOOKUP to q and asks q to make an MLOOKUP to p.

in the other ring. In such a case, an *mlookup* will immediately leave the initiating node's ring, and hence terminate. We do not see how such a pathological scenario could occur due to a partition, but the *gossip-based ring unification algorithm* (Algorithm 26) rectifies both disadvantages of the simple ring unification algorithm. Moreover, the simple ring unification is less robust to churn, as we discuss in the evaluation section.

Algorithm 26 is, as its name suggests, gossip-based. The algorithm is essentially the same as the simple ring unification algorithm, with a few additions. The intuition is to have the initiator of the algorithm immediately start multiple instances of the simple algorithm at random nodes, with uniform distribution. But since the initiator's pointers are not uniformly distributed, the process of picking random nodes is incorporated into MLOOKUP. Thus, MLOOKUP(*id*) is augmented so that the current node randomly picks a node r in its current routing table and starts a ring merger between *id* and r. This change alone would, however, consume too much resources.

Two mechanisms are used to prevent the algorithm from consuming too many messages, which could give rise to positive feedback cycles that congest the network. First, instead of immediately triggering an MLOOKUP at a random node, the event is placed in the corresponding node's *detqueue*, which is only checked periodically. Second, a constant number of random MLOOKUPs are created. This is regulated by a fanout parameter called F. Thus, the fanout is decreased each time a random node is picked, and the random process is only started if the fanout is larger

Algorithm 26: Gossip-based Ring Unification Algorithm

```
every $\gamma$ time units and $detqueue \neq \emptyset$ at $p$
    $\langle q, f \rangle := detqueue$.dequeue()
    sendto $p$ : MLOOKUP($q, f$)
    sendto $q$ : MLOOKUP($p, f$)
end event

receipt of MLOOKUP($id, f$) from $m$ at $n$
    if $id \neq n$ and $id \neq succ$ then
        if $f > 1$ then
            $f := f - 1$
            $r :=$ randomnodeinRT()
            at r : $detqueue$.enqueue($\langle id, f \rangle$)
        end if
        if $id \in (n, succ)$ then
            sendto $id$ : TRYMERGE($n, succ$)
        else if $id \in (pred, n)$ then
            sendto $id$ : TRYMERGE($pred, n$)
        else
            sendto closestprecedingnode($id$) : MLOOKUP($id, f$)
        end if
    end if
end event

receipt of TRYMERGE($cpred, csucc$) from $m$ at $n$
    sendto $n$ : MLOOKUP($csucc, F$)
    if $csucc \in (n, succ)$ then
        $succ := csucc$
    end if
    sendto $n$ : MLOOKUP($cpred, F$)
    if $cpred \in (pred, n)$ then
        $pred := cpred$
    end if
end event
```

than 1. The *detqueue*, therefore, holds tuples, which contain a node identifier and the current fanout parameter. Similarly, MLOOKUP takes the current fanout as a parameter. The rate for periodically checking the *detqueue* can be adjusted to control the rate at which the algorithm generates messages.

5 Performance Evaluation

In this section, a short evaluation of the two algorithms for ring-based overlays from various aspects and in different scenarios is presented. A detailed evaluation can be found in the original paper [29]. For the evaluation, there are two measures of interest: *message complexity*, and *time complexity*. A differentiation is made between the *completion* and *termination* of the algorithm. Completion means the time when the rings have merged. Termination means the time when the algorithm terminates sending any more messages. Unless said otherwise, message complexity is until termination, while time complexity is until completion.

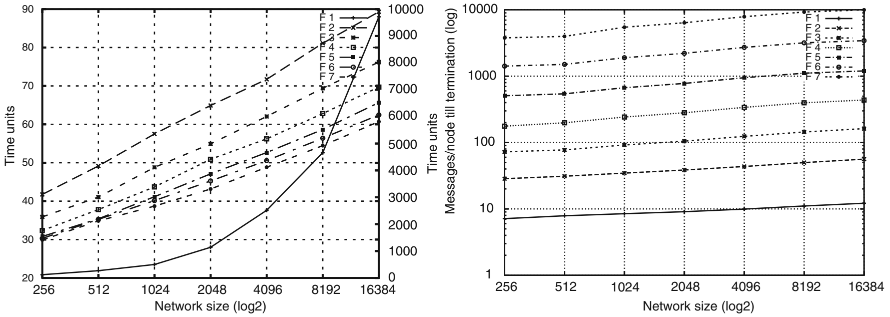
The simulation scenario had the following structure. Initially nodes join and fail. After a certain number of nodes are part of the system, a partition event is inserted, upon which the simulator divides the set of nodes into as many components as requested by the partition event, dividing the nodes randomly into the partitions but maintaining an approximate ratio specified. The simulator then drops all messages sent from nodes in one partition to nodes in another partition, thus simulating a network partition in the underlying network and therefore triggering the failure handling algorithms. Furthermore, node join and fail events are triggered in each partitioned component. Thereafter, a network merger event simply allows messages to reach other network components, triggering the detection of alive nodes in the passive lists, and hence starting the ring unification algorithms.

The simple ring unification algorithm and the gossip-based ring unification algorithm were simulated for partitions creating two components of approximately the same size, and for fanout values from 1 to 7. For all the simulation graphs to follow, a fanout of 1 represents the simple ring unification algorithm. A time unit was equal to the time it takes for a message to reach its destination node.

To proper understand the performance of the proposed algorithm, let us study scenarios where only one node starts the merger of the two rings. The two nodes that are involved in the merger are randomly selected, with uniform probability, i.e. the node p that detects the merger and the node that p detects from its passive list.

To study the scalability of ring unification, consider network sizes of powers of 2. Results for time complexity are shown in Fig. 3a. The graph for the gossip-based algorithms is linear, which suggests a $O(\log n)$ time complexity. In contrast, the simple ring unification graph (F=1) is exponential, indicating that it does not scale well, i.e. $\omega(\log n)$ time complexity. Figure 3b plots the number of ring unification messages sent by each node during the merger, i.e. the total number of messages induced by the algorithm until termination divided by the number of nodes. The linear graph on a log-log plot indicates a polynomial messages complexity. As expected, the number of messages per node grows slower for simple ring unification compared to gossip-based ring unification.

It is evident from Fig. 3a, b that the simple ring unification algorithm ($F = 1$) consumes minimum messages but takes maximum time when compared to different variations of the gossip-based ring unification algorithm. For higher values of F, the time complexity decreases while the message complexity increases. Increasing the fanout after a threshold value (around 3–4 in this case) will not considerably decrease the time complexity, but will just generate many unnecessary messages. Figure 4 illustrates this tradeoff between time and message complexity. It shows that the goals of decreasing time and message complexity are conflicting. Thus, to decrease the number of messages, the time for completion will increase. Similarly, opting for convergence in lesser time will generate more messages. A suitable fanout value can be used to adapt the ring unification algorithm according to the requirements and network infrastructure available.



(a) Time complexity.
Only F 1 is plotted against the right y-axis.

(b) Message complexity

Fig. 3 Evaluation of time and message complexity when only one node starts the merger

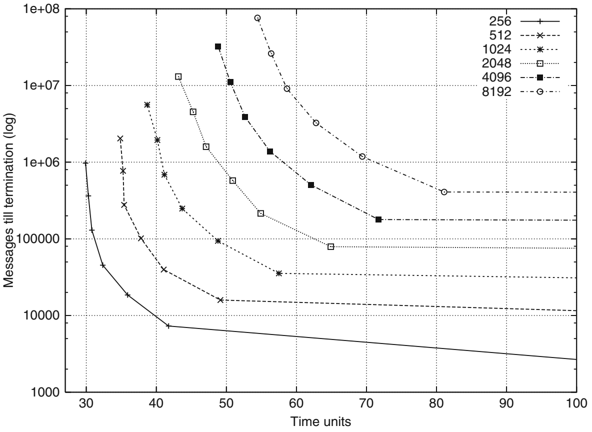


Fig. 4 Tradeoff between time and message complexity

We now consider a scenario where a node may falsely detect a merger. Figure 5 shows the message complexity of the algorithm in case of a false detection. As can be seen, for lower fanout values, the message complexity is less. Even for higher fanouts, the number of messages generated are acceptable, thus showing that the algorithm is lean. This result is important as most overlays do not have perfect failure detectors, and hence can give rise to inaccurate suspicions.

Since overlays are always subject to churn, it is important to study how often the partitioned components do not converge to a ring under churn. Experiments with 200 different seeds for sizes ranging from 256 to 2048 nodes were executed. An execution was considered successful if 95% of the nodes had correct successor pointers, as all successor pointers can not be correct while nodes are joining and failing. Thereafter, the remaining pointers are updated by Chord's periodic stabilization. For the 200 executions, only 1 unsuccessful execution was observed for network size 1024 and 2 unsuccessful executions for network size 2048. The unsuccessful executions happened only for simple ring unification, while executions with

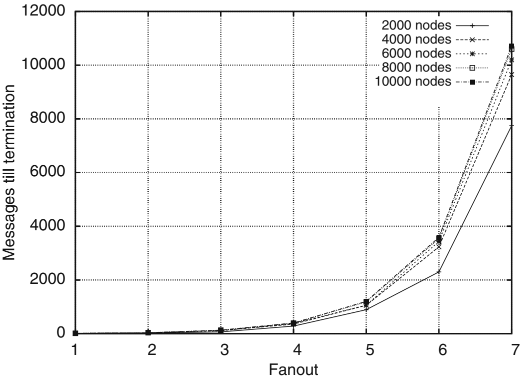


Fig. 5 Message complexity for false detection of merger

gossip based ring unification were always succecessful. Even for the unsuccessful executions, given enough time, periodic stabilization updates the successor pointers to correct nodes.

As we discuss in Section 6.3, the problem of merging overlays can be solved by a self-stabilizing ring algorithm. Thus, a comparison between ring unification with a Self-Stabilizing Ring Network (SSRN) [30] protocol is necessary. The results comparing time and message complexity for various network sizes for the two algorithms are presented in Fig. 6a,b, depicting that ring unification consumes lesser time and messages compared to SSRN. The main reason for the better performance of Ring Unification is that it has been designed specifically for merging rings. On the other hand, SSRN is a non-terminating algorithm that runs in the background like periodic stabilization to find closer nodes. As evaluated previously, simple ring unification (fanout=1) does not scale well for time complexity, which can be seen in Fig. 6a.

Simulations suggest that a fanout value of 3–4 is good for a system with several thousand nodes, even with respect to churn and false-positives.

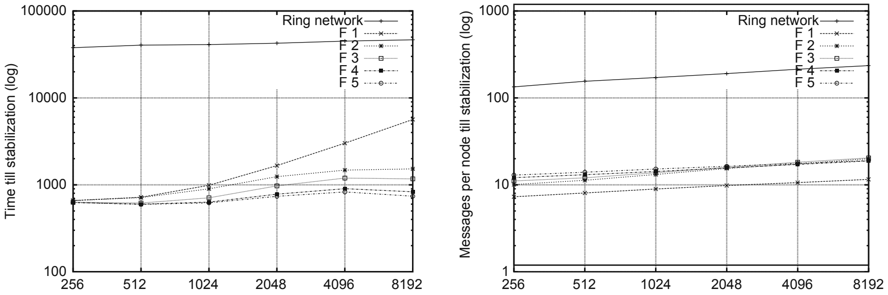
6 Related Work

Much work has been done to study the effects of churn on a structured overlay network [22], showing how overlays can cope with massive node joins and failures, thus showing how overlays are resilient to partitions. In contrast, there has not been much focus on merging the overlays once the underlying network merges. In this section, we discuss some related work, including merging tree-based overlays.

6.1 Merging P-Grid Overlays

In this section, we briefly describe P-Grid [2] and show how multiple P-Grid overlays can be merged [6].

Merging Overlays



(a) Time complexity.

(b) Message complexity.

Fig. 6 Comparison of ring unification and SSRN [30]

6.1.1 P-Grid

P-Grid structures the overlay as a tree, and uses prefix-based routing. P-Grid divides the key-space into mutually exclusive partitions, thus each key belongs to a unique partition. To lookup a key, prefix routing is employed to reach the partition to which the key belongs. Nodes in P-Grid take responsibilities of key-partitions. Each node is reponsible for a single key-partition, while a key-partition can have multiple nodes responsible for it. The number of nodes per key-partition varies, and is dependent on the load, i.e. size of data items stored in a particular key-partition. Nodes in one key-partition are called *structural replicas*, as they replicate the same keys. Thus, when a lookup reaches the key-partition, any structural replica in the key-partition can reply to the lookup. For routing, each node maintains pointers to other key-partitions.

Figure 7a, b show two P-Grid overlays. In Fig. 7a, the key space is divided into four key-partitions, $\{00\cdot, 01\cdot, 10\cdot, 11\cdot\}$. Nodes *a* and *e* are structural replicas for key-partition $00\cdot$, and have routing entries for key partitions $01\cdot$ and $1\cdot$.

6.1.2 Merger

Assume there are two overlays, $O1$ and $O2$, and a peer from one overlay knows about a peer from the other overlay by one of the methods discussed in Section 3. We describe the merger of multiple P-Grid overlays by explaining what happens when peers from different overlays interact, and how do peers from one overlay find peers from the other overlay to interact.

Meeting of peers from different networks

Assume two peers *p* and *q* from two different overlays meet. There can be the following three cases.

Case 1: p **and** q **have same path**

When peers from two different overlays meet such that their paths are exactly the same, *i.e.* they are structural replicas of the same key-partition though the key-partitions belong to different overlays, they become mutual structural replicas. Since P-Grid employes anti-entropy to keep structural replicas updated, the peers will reconcile their content and view. For example, peer *c*, belonging to *O*1 (Fig. 7a), and peer *y*, belonging to *O*2 (Fig. 7b), share the same path, i.e prefix 10. Thus, they become structural replicas and execute anti-entropy. It is worth noting that in this case, despite the overlay merger process, whichever keys were originally accessible will continue to be accessible during the merge process.

Case 2: p**'s path is a strict prefix of** q

When peers from two different overlays meet such that the path to one, *p*, is a strict prefix to the path of the other peer *q*, the peer with the shorter path executes a normal network joining algorithm. This rejoining can be done by *p* by either extending its path to replicate *q*, or replicate another peer recommended by *q* due to other considerations, like load balancing.

For instance, peer *x*, belonging to *O*2 (Fig. 7b), shares a strictly smaller prefix 0 with peer *a*, belonging to *O*1 (Fig. 7a). Thus, if *x* and *a* interact, *x* can extend its path to replicate *a*, i.e. structurally replicate prefix 00. In this case, the key-space partitioning changes, thus some keys might need to be shipped to other nodes before rejoining/extending the path.

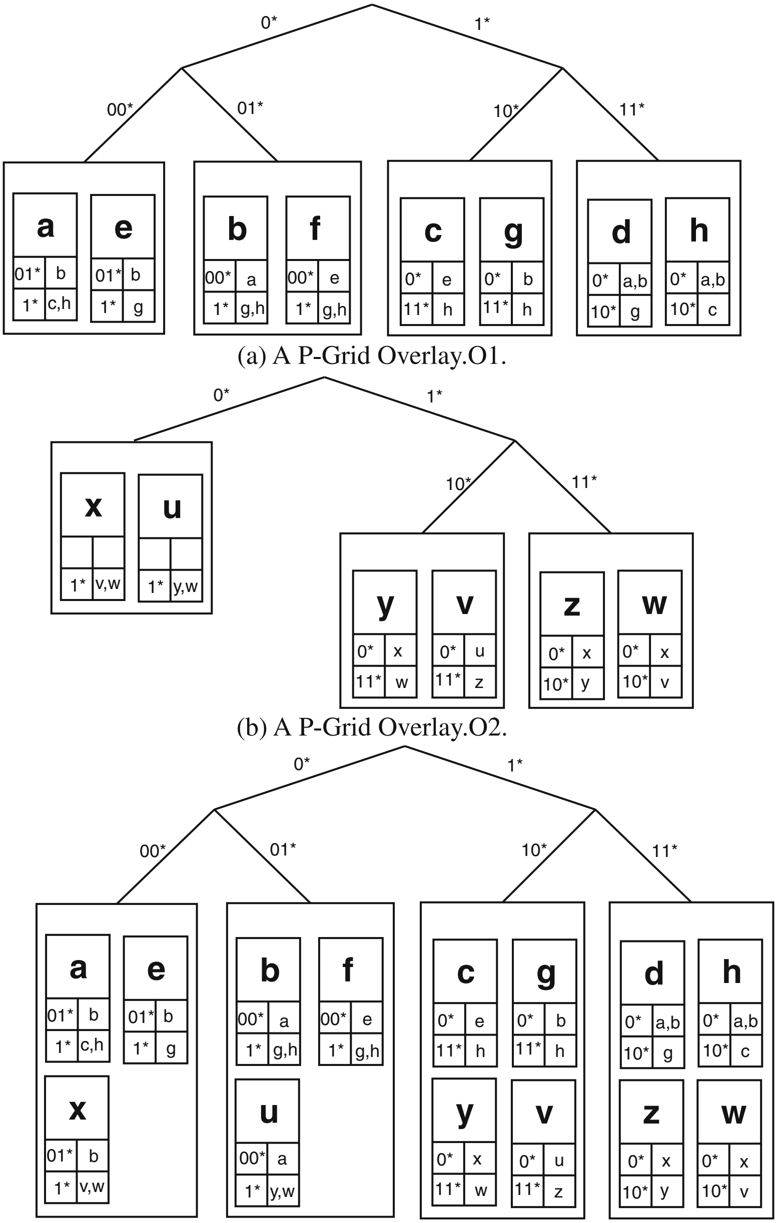
Case 3: p **and** q **do not share any prefix**

If the two peers interacting from different overlays do not share any prefix, they refer each other to peers with longer mutual match of path. This is similar to propagating the merger information, discussed shortly.

Updating routing tables

When overlays merge, apart from peers originally belonging to different overlays becoming structural replicas of the same key-partition, it is important that routing tables are also updated. Thus after merger, when *c* exchanges its view with *y*, *c* will have two routes for the prefix 11, *w* and *h*. This might lead to keys being unavailable, as a lookup from a peer initially part of *O*1 for prefix 11 routed through *c* may either endup at *h* or *w*. Since *w* does not yet have keys for *O*1, the lookup will fail. This is certainly undesireable.

The afore-mentioned problem can be remedied in various ways. A simple solution is to forward lookups to all routing entries matching the prefix. In the case mentioned, this would lead *c* to route lookups for prefix 11 to both *h* and *w*. While this method solves the problem, it suffers from the problem of increased message complexity. Another solution would be to mark lookups and routing entries with the overlay's unique identification. Lookups are then forwarded to peers in the routing table that originally belonged to the same overlay. In the afore-mentioned case, the



(a) A P-Grid Overlay.O1.

(b) A P-Grid Overlay.O2.

(c) a Possible P-Grid Overlay after O1 andO2 merge.

Fig. 7 Merger of two P-Grid overlays

lookup will contain a field marked $O1$, thus c will forward the lookup to it's routing entry that has a tag $O1$, in this case h not w. Once the merging process completes, the peers need not be distinguished according to their original overlay. This method suffers from two problems. First, in practice, it is difficult for peers to keep track of which overlay they belong to. Second, identifying the completion of the merge process in a decentralized way is problemetic. A simple solution might be to use a timeout, after which the merge is considered complete.

Propagating merger information

The merger information can be propagated by triggering new interactions between peers. This can be done by flooding the information about the new network to all entries in a peer's routing table. For example, when c and y interact, y can inform its routing table entries w and x about c. Consequently, w and x can use c's entries to find suitable peers for interaction, e.g. h for w. This flooding results in logarithmic time spreading of the information of the new overlay, though consumes extra messages.

6.2 Merging Pastry Overlays

The original Pastry overlay [28] did not handle network mergers. Merging of Pastry overlays was designed while deploying ePOST [24], a reliable peer-to-peer application, on the Internet. In Pastry, each node has a leafset, which it periodically exchanges with other nodes in it's leafset. To detect the partition and merger, the nodes employ a strategy similar to Section 3, adding failed nodes in their leafset into passive lists. The difference being that in Pastry, far away nodes are not added to the passive lists.

After a node detects a merger by being able to ping a previously failed node in its leafset, it exchanges its leafset with the other node. Since exchange of leafsets is periodic, this will have a cascading effect that will eventually merge the overlays. The merger can be executed faster if multiple nodes at different parts of the overlay detect the merger and exchange leafsets. Executing this algorithm will fix the overlay geometry, while the routing tables are updated lazily. It is worth noting that since the merger spreads by a cascading effect, it is slow compared to gossip-based Ring Unification.

6.3 Overlay Construction from Random Graph

The problem of constructing an overlay from a random graph is, in some respects, similar to merging multiple overlays after a network merger, as the nodes may get randomly connected after a partition heals. Shaker et al. [30] have presented a ring-based algorithm for nodes in arbitrary state to converge into a directed ring topology. Their approach is different from the work presented in this chapter, in the sense that they provide a non-terminating algorithm which should be used to replace all join, leave, and failure handling routines of an existing overlay. Replacing the topology maintenance algorithms of an overlay may not always be feasible, as overlays may have intricate join and leave procedures to guarantee lookup consistency [9, 18, 21]. In contrast, the algorithms presented in this chapter are terminating, and they work as a plug-in for an already existing overlay.

Montresor et al. [25] show how Chord [31] can be created by a gossip-based protocol [14]. However, their algorithm depends on an underlying membership service like Cyclon [34], Scamp [8] or Newscast [15]. Thus the underlying membership service has to first cope with network mergers (a problem worth studying in its own right), where after T-Chord [25] can form a Chord network. One needs to investigate further how these protocols can be combined, and their epochs be synchronized, such that the topology provided by T-Chord is fed back to the overlay when it has converged. Though the general performance of T-Chord has been evaluated, it is not known how it performs in the presence of network mergers when combined with various underlying membership services.

As shown below, it might happen that an initially connected graph can be split into two separate components by the Chord [31] and SSRN [30] protocols. This scenario is a counter-proof of the claim that SSRN is self-stabilizing. Consider a network which consists of two perfect rings, yet the nodes have fingers pointing to nodes in the other ring. This can easily happen in case of unreliable failure detectors [5] or networks partitions. Normally, the PS rate is higher than fixing fingers, thus due to a temporary partition, it might happen that nodes update their successor pointers, yet before they fix their fingers, the partition heals. In such a scenario, SSRN splits the connected graph into two separate partitions, thus creating a partition of the overlay, while the underlay remains connected. An example of such a scenario is shown in Fig. 8, where the filled circles are nodes that are part of one ring and the empty circles are nodes that are part of the other ring. Each node has one finger pointing to a node in the other ring. The fix-finger algorithm in Chord updates the fingers by making lookups. In this case, a lookup will always return a node in the same ring as the one making the lookup. Consequently, the finger pointing to the other ring will be lost. Similarly, the pointer jumping algorithm used by SSRN to update its fingers will also drop the finger pointing to a node in the other ring. On the contrary, the ring-unification algorithm proposed in this paper will fix such a graph and converge it to a single ring.

6.4 Data Consistency

This section gives a short discussion of the issues on the data level amid network partitions and mergers. The data level is concerned with availability and consistency of data stored in the overlay. Availability is achieved by replicating the data. When

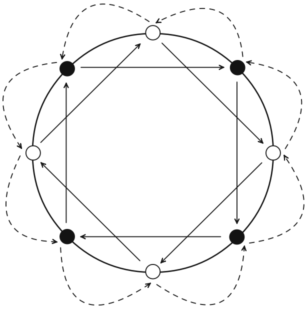


Fig. 8 A case where chord and the ring network protocol would break a connected graph into two components. Lines represent successor pointers while dashed lines represent a finger

the overlay partitions, a high replica factor and geographically spread replicas may increase availability. Similarly, proper routing while the overlays are merging guarantees availability of data during the merge process. Once the overlays merge, the data in one overlay becomes available to nodes in the other overlay.

The data level is also concerned with the consistency of the data items stored in the overlay. The solutions to this problem might depend on the application and on the semantics of the data operations, e.g. immutable key/value pairs or monotonically increasing values. It is also known that it is impossible to achieve strong (atomic) data consistency, availability,[3] and partition tolerance in overlays [9, 10].

The problem of network partitions and mergers has been studied in other distributed systems, such as in distributed databases [7] and distributed file systems [33]. These studies focus on problems created by the partition and merger on the data level. Such ideas, if combined with algorithms such as those proposed in this chapter for the routing level, can be used for handling data updates in overlays. The details of which is, nevertheless, outside the scope of this chapter.

7 Summary

The problem of partitions and mergers in structured peer-to-peer systems, when the underlying network partitions and recovers, is of crucial importance. This chapter presents algorithms for merging similar ring-based and tree-based structured overlay networks after the underlying network merges.

Though the problem of dealing with network mergers is crucial, such events happen more rarely. Hence, it might be justifiable in certain application scenarios that a slow paced algorithm runs in the background, consuming little resources, while ensuring that any potential problems with partitions will eventually be rectified. For ring-based overlays, we have shown how the algorithm can be tuned to achieve a tradeoff between the number of messages consumed and the time before the overlay

[3] By availability we mean that a get/put operation should eventually complete.

converges. Evaluations in realistic dynamic conditions have been presented, and it has been showed that with high fanout values, the algorithm can converge quickly under churn. We have also shown that the solution generates few messages even if a node falsely starts the algorithm in an already converged overlay. We have also briefly discussed solutions for tree-based ovelays.

7.1 Open Questions

We believe that dealing with partitions and mergers is a small part of a bigger, and more important, goal: making overlays that can recover from any configuration. We believe that it is desirable to make a self-stabilizing ring algorithm, which can be proved to recover from all possible states, including loopy [20] and partitioned while consuming minimum time and messages.

We believe that it is interesting to investigate whether gossip-based topology generators, such as T-man [14] and T-chord [25], can be used to handle network mergers. These services, however, make use of an underlying membership service, such as Cyclon [34], Scamp [8], or Newscast [15]. Hence, one has to first investigate how well such membership services recover from network partitions (we believe this to be interesting in itself). Thereafter, one can explore how such topology generators can be incorporated into an overlay.

Acknowledgement

This work has been funded by the European Union in the SELFMAN Project (Contract 034084) and SICS Center for Networked Systems (CNS).

References

1. K. Aberer, L. O. Alima, A. Ghodsi, S. Girdzijauskas, S. Haridi, and M. Hauswirth. The Essence of P2P: A Reference Architecture for Overlay Networks. In *Proceedings of the 5th International Conference on Peer-To-Peer Computing (P2P'05)*, pages 11–20. IEEE Computer Society, August 2005.
2. K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-Grid: a self-organizing structured P2P system. *SIGMOD Record*, 32(3):29–33, 2003.
3. L. O. Alima, A. Ghodsi, and S. Haridi. A Framework for Structured Peer-to-Peer Overlay Networks. In *Post-proceedings of Global Computing*, volume 3267 of *Lecture Notes in Computer Science (LNCS)*, pages 223–250. Springer Verlag, 2004.
4. A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: Supporting Scalable Multi-Attribute Range Queries. In *Proceedings of the ACM SIGCOMM 2004 Symposium on Communication, Architecture, and Protocols*, pages 353–366, Portland, OR, USA, March 2004. ACM Press.

5. T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, 1996.
6. A. Datta. Merging Intra-Planetary Index Structures: Decentralized Bootstrapping of Overlays. In *Proceedings of the First International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2007)*, pages 109–118, Boston, MA, USA, July 2007. IEEE Computer Society.
7. S. B. Davidson, H. Garcia-Molina, and D. Skeen. Consistency in a partitioned network: a survey. *ACM Computing Surveys*, 17(3):341–370, 1985.
8. A. J. Ganesh, A.-M. Kermarrec, and L Massoulié. SCAMP: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication. In *Proceedings of the 3rd International Workshop on Networked Group Communication (NGC'01)*, volume 2233 of *Lecture Notes in Computer Science (LNCS)*, pages 44–55, London, UK, 2001. Springer-Verlag.
9. A. Ghodsi. *Distributed k-ary System: Algorithms for Distributed Hash Tables*. PhD dissertation, KTH – Royal Institute of Technology, Stockholm, Sweden, December 2006.
10. S. Gilbert and N. A. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM Special Interest Group on Algorithms and Computation Theory News*, 33(2):51–59, 2002.
11. K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of DHT routing geometry on resilience and proximity. In *Proceedings of the ACM SIGCOMM 2003 Symposium on Communication, Architecture, and Protocols*, pages 381–394, New York, NY, USA, 2003. ACM Press.
12. N. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS'03)*, Seattle, WA, USA, March 2003. USENIX.
13. F. Jahanian C. Labovitz, A. Ahuja. Experimental Study of Internet Stability and Wide-Area Backbone Failures. Technical Report CSE-TR-382-98, University of Michigan, November 1998.
14. M. Jelasity and Ö. Babaoglu. T-man: Gossip-based overlay topology management. In *Proceedings of 3rd Workshop on Engineering Self-Organising Systems (EOSA'05)*, volume 3910 of *Lecture Notes in Computer Science (LNCS)*, pages 1–15. Springer-Verlag, 2005.
15. M. Jelasity, W. Kowalczyk, and M. van Steen. Newscast Computing. Technical Report IR–CS–006, Vrije Universiteit, November 2003.
16. M. F. Kaashoek and D. R. Karger. Koorde: A Simple Degree-optimal Distributed Hash Table. In *Proceedings of the 2nd Interational Workshop on Peer-to-Peer Systems (IPTPS'03)*, volume 2735 of *Lecture Notes in Computer Science (LNCS)*, pages 98–107, Berkeley, CA, USA, 2003. Springer-Verlag.
17. B. Leong, B. Liskov, and E. Demaine. EpiChord: Parallelizing the Chord Lookup Algorithm with Reactive Routing State Management. In *12th International Conference on Networks (ICON'04)*, Singapore, November 2004. IEEE Computer Society.
18. X. Li, J. Misra, and C. G. Plaxton. Brief Announcement: Concurrent Maintenance of Rings. In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC'04)*, page 376, New York, NY, USA, 2004. ACM Press.
19. J. Li, J. Stribling, R. Morris, and M. F. Kaashoek. Bandwidth-efficient management of DHT routing tables. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'05)*, Boston, MA, USA, May 2005. USENIX.
20. D. Liben-Nowell, H. Balakrishnan, and D. R. Karger. Observations on the Dynamic Evolution of Peer-to-Peer Networks. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS'02)*, volume 2429 of *Lecture Notes in Computer Science (LNCS)*. Springer-Verlag, 2002.
21. N. A. Lynch, D. Malkhi, and D. Ratajczak. Atomic Data Access in Distributed Hash Tables. In *Proceedings of the First Interational Workshop on Peer-to-Peer Systems (IPTPS'02)*, Lecture Notes in Computer Science (LNCS), pages 295–305, London, UK, 2002. Springer-Verlag.
22. R. Mahajan, M. Castro, and A. Rowstron. Controlling the Cost of Reliability in Peer-to-Peer Overlays. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems*

(IPTPS'03), volume 2735 of *Lecture Notes in Computer Science (LNCS)*, pages 21–32, Berkeley, CA, USA, 2003. Springer-Verlag.

23. G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed Hashing in a Small World. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS'03)*, Seattle, WA, USA, March 2003. USENIX.
24. A. Mislove, A. Post, A. Haeberlen, and P. Druschel. Experiences in building and operating ePOST, a reliable peer-to-peer application. In Willy Zwaenepoel, editor, *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems*. ACM European Chapter, April 2006.
25. A. Montresor, M. Jelasity, and Ö. Babaoglu. Chord on Demand. In *Proceedings of the 5th International Conference on Peer-To-Peer Computing (P2P'05)*. IEEE Computer Society, August 2005.
26. D. Oppenheimer, A. Ganapathi, and D. A. Patterson. Why do internet services fail, and what can be done about it? In *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*, pages 1–1, Berkeley, CA, USA, 2003. USENIX Association.
27. V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking (TON)*, 5(5):601–615, 1997.
28. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 2nd ACM/IFIP International Conference on Middleware (MIDDLEWARE'01)*, volume 2218 of *Lecture Notes in Computer Science (LNCS)*, pages 329–350, Heidelberg, Germany, November 2001. Springer-Verlag.
29. T. M. Shafaat, A. Ghodsi, and S. Haridi. Handling Network Partitions and Mergers in Structured Overlay Networks. In *Proceedings of the 7th International Conference on Peer-to-Peer Computing (P2P'07)*, pages 132–139. IEEE Computer Society, September 2007.
30. A. Shaker and D. S. Reeves. Self-Stabilizing Structured Ring Topology P2P Systems. In *Proceedings of the 5th International Conference on Peer-To-Peer Computing (P2P'05)*, pages 39–46. IEEE Computer Society, August 2005.
31. I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking (TON)*, 11(1):17–32, 2003.
32. Taiwan Earthquake on December 2006. http://www.pinr.com/report.php?ac=view_report&report_id=602, January 2008.
33. D. B. Terry, M. Theimer, K. Petersen, A. J. Demers, M. Spreitzer, and C. Hauser. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP'95)*, pages 172–183. ACM Press, December 1995.
34. S. Voulgaris, D. Gavidia, and M. van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, 13(2), 2005.

Load Balancing in Structured P2P Networks

Yingwu Zhu

Abstract In this chapter we start by addressing the importance and necessity of load balancing in structured P2P networks, due to three main reasons. First, structured P2P networks assume uniform peer capacities while peer capacities are heterogeneous in deployed P2P networks. Second, resorting to pseudo-uniformity of the hash function used to generate node IDs and data item keys leads to imbalanced overlay address space and item distribution. Lastly, placement of data items cannot be randomized in some applications (e.g., range searching). We then present an overview of load aggregation and dissemination techniques that are required by many load balancing algorithms. Two techniques are discussed including tree structure-based approach and gossip-based approach. They make different tradeoffs between estimate/aggregate accuracy and failure resilience. To address the issue of load imbalance, three main solutions are described: virtual server-based approach, power of two choices, and address-space and item balancing. While different in their designs, they all aim to improve balance on the address space and data item distribution. As a case study, the chapter discusses a virtual server-based load balancing algorithm that strives to ensure fair load distribution among nodes and minimize load balancing cost in bandwidth. Finally, the chapter concludes with future research and a summary.

1 Introduction

1.1 Motivation

The basic approach to the load balancing issue in structured P2P networks or distributed hash tables (DHTs) [8, 10, 13] is consistent hashing [15]. Nodes are

Yingwu Zhu
Seattle University, Seattle WA 98122, USA,
e-mail: zhuy@seattleu.edu

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_41, © Springer Science+Business Media, LLC 2010

associated with IDs that are generated by hashing their IPs or public keys, while data items are assigned with keys that are derived from hashing their names or contents. By node IDs and data keys respectively, nodes and data items are mapped into a one-dimensional ring (or identifier space), and data items are assigned to the nearest node in the clockwise direction. Ideally, all nodes are responsible for an equally-sized portion of the identifier space (or arc of the ring). That is, nodes are evenly distributed over the ring. When data keys are numerous and uniformly random, each node is supposed to own a same number of keys or data items. However, the consistent hashing approach can cause considerable load imbalance.

First, the consistent hashing approach causes some node responsible for a maximum arc length of $O(\frac{\log N}{N})$ with high probability due to pseudo-uniformity of the hash function, even though the average arc length is $\frac{1}{N}$, where N is the number of nodes. In other words, the approach does not evenly partition the one-dimensional identifier space among nodes, which we call *address space imbalance*. If data item keys are numerous and uniformly random, some nodes receive more than their fair share, causing load imbalance of $O(\log N)$ with high probability.

Second, the consistent hashing approach assumes uniform node capacities while the capabilities of nodes (e.g., storage and bandwidth) are heterogeneous in deployed P2P networks, differing by multiple orders of magnitude [9]. This further aggravates the problem of load imbalance.

Finally, the consistent hashing approach destroys data semantics by randomizing placement of data items which is not applicable to some applications. For example, to support range searching in a database application the items need to be in a specific order or even at specific addresses, on the ring. In such cases, the data items are likely to be unevenly distributed over the ring due to skewed distributions of data keys, worsening the issue of load imbalance.

The primary goal of P2P systems is to harness all available resources (e.g., storage, bandwidth, and CPU) in the network so that users can access all available objects “efficiently”. From the P2P system perspective, “efficiently” is interpreted as striving to ensure fair load distribution among all peer nodes. With the aforementioned load imbalance issue in DHTs, achieving load balance is of fundamental importance. As a result, various load balancing algorithms have been proposed for DHTs.

1.2 Load Definition

Each data item in the system has an associated load. The load could represent the number of bits to store the item, the bandwidth of transferring the item, the CPU cycles of serving the item, or the number of requests for the item (i.e., popularity of the item). The discussions in this chapter do not assume a particular resource. It is worth pointing out that the popularity distribution of data items might cause the problem of “hot spots”. The load balancing algorithms presented in the chapter are, by no means, intended to neglect other load balancing techniques not discussed here, such as caching and replication.

1.3 Goals

In particular, a load balancing algorithm should strive to achieve some or all of the following (often conflicting) goals:

- **Minimize load imbalance.** To provide best-quality services, each node should have same utilization which is defined as the ratio of *load* and *capacity* on a node. As nodes are heterogeneous, load should be assigned to a node proportional to its capacity.
- **Decentralization.** To ensure robust load balancing, the algorithm should avoid any centralized control; Load balancing operations such as load information aggregation and dissemination and load movement should be fully distributed.
- **Responsiveness.** Churn is the norm in P2P networks. Frequent node arrivals and departures cause rapid load migration among nodes. The algorithm should react to dynamically changing load distribution, rebalancing load in a timely manner.
- **Efficiency.** The algorithm should be resource-friendly, minimizing resource consumption in load balancing. For example, load transfer among nodes for rebalance costs bandwidth; the less the cost, the more feasible the algorithm. In other words, a resource-intensive algorithm is impractical for most environments, if not all.

2 Load Information Aggregation and Dissemination

To react to the change of load status system wide, many load balancing algorithms need to continuously monitor system status by periodically performing load information aggregation and dissemination. With collected load status, they make informed decisions on load balancing. The primary goal is to gain a summary view of the global load status and spread the summary view among nodes in a timely manner. In what follows, we briefly discuss two main techniques in the literature for load information aggregation and dissemination.

2.1 Tree Structure-Based Approach

In this approach, a tree spanning all (physical) nodes on the DHT overlay is constructed for information aggregation and dissemination and is layered atop the overlay. Each physical node is a leaf node and each subtree represents a logical group of nodes. The logical group of nodes can correspond to a set of nodes within an administrative domain [11] or a set of neighboring nodes on an arc of the ring (i.e., a region of the DHT identifier space) [14]. An internal non-leaf node, which we call *virtual node*, is planted on a DHT node in certain manner. Namely, a key is assigned to a virtual node and the virtual node is planted in a DHT node which is responsible for the key. A virtual node on a higher level of the tree corresponds to a larger set of nodes, either from a larger domain or from a longer arc of the ring. The root node corresponds to all the physical nodes on the overlay.

For load information aggregation, each leaf node reports its local load status (e.g., load, capacity, etc.) to its parent node which aggregates all information from its children and then propagates the aggregate upwards. As such, the root node gathers the load information and generates a summary view of the global load status. Then, the root node disseminates the summary view to each leaf node along the tree in a top-down fashion. The information aggregation and dissemination delay is determined by the tree height.

2.2 Gossip-Based Approach

This approach relies on gossip-style exchange of data and uses data synopsis techniques to estimate the distribution of values (e.g., load and capacity) held by the nodes. Each node maintains a list of k values, initially set to the local value held by the node. In each round, a node i randomly picks a partner j and requests the k values held by j. After receiving the k values, node i merges the $2k$ values into a list of k values in certain manner. In the simplest form, the merging operation randomly picks k values, at the risk of discarding important information. Haridasan et al. [5] proposes three different synopsis techniques, including *Concise Counting*, *Equi-Width Histograms* and *Equi-Depth Histograms*, to estimate the distribution of values, with limited information loss and consuming less storage space and communication bandwidth.

In Equi-Depth Histograms technique, each node initially divides the range $[0, x_i]$ (x_i is a local value) into k equally-sized bins, each represented by a pair $< value, counter >$. After exchanging data with a partner, each node orders all collected pairs, and computes which consecutive bins, when merged, yield the smallest combined bin (with the least counter). The identified bins are merged into a new bin, by adding their counters as the new counter and using the arithmetic mean of their values as the new value. This process repeats until the desired number of bins are left. The goal of Equi-Depth Histograms technique is to minimize disparity across all bins. After a sufficient number of gossips, the resulting bins comprises a synopsis which allows the node to estimate the distribution of values (which could be load or capacity) among the nodes.

2.3 Summary

Tree-based and gossip-based approaches make different tradeoffs between aggregate/estimate accuracy and failure resilience. Tree-based approaches produce aggregates hierarchically and under no-failure scenarios result in exact aggregates. In the presence of node churn as nodes continuously join and leave, gossip-based techniques are more resilient, though it may not be always possible to generate exact aggregates.

3 Load Balancing

When some nodes are becoming overloaded, various operations can be taken to rebalance load among nodes, including load movement/transfer and node movement. In what follows, we provide an overview of three different approaches, namely virtual server-based approach [3, 4, 10, 14], the power of two choices [2], and address-space and item balancing [6].

3.1 Virtual Server-Based Approach

Chord [10] was the first to propose the concept of virtual servers as a means of improving load balance. Like a physical node, a virtual server owns an ID, manages a routing table, and is in charge of a region of the DHT identifier space. The data items whose IDs fall into a virtual server's responsible region are stored on the virtual server. By allowing each physical node to host $\log N$ virtual servers (which means the physical node is responsible for $\log N$ non-contiguous regions of the DHT identifier space), Chord ensures that with high probability the number of objects on any node is within a constant factor of the average, on the assumption that nodes are homogeneous and data item keys are uniformly distributed. In the same vein, CFS [3] accounts for node heterogeneity by assigning to each node some number of virtual servers proportional to the node capacity. To shed load from an overloaded node, CFS dynamically turns off some of its virtual servers. Simply removing virtual servers from an overloaded node, however, can result in thrashing as this may causes another node becoming overloaded.

Leveraging Chord and CFS, recent work [4, 14] achieves load balance by transferring load from an overloaded node to a lightly loaded node in the unit of virtual servers. Virtual server transfer, as load movement, comes free as it can be simulated as a *leave* operation followed by a *join* operation, both supported by all DHTs.

3.2 Power of Two Choices

This approach [2] assumes a set of $d \geq 2$ universally agreed hash functions $h_1(), \cdots, h_d()$. It bases item insertion on sampling. To insert an item x, the approach calculates $h_1(x), \cdots, h_d(x)$ using the d hash functions, and contacts d nodes responsible for the hash values. The least loaded node among the d nodes stores the item x while the other nodes each keeps a redirection pointer for x. If a node becomes overloaded (lightly loaded), load shedding (stealing) is triggered by performing item re-insertions. For homogeneous nodes and data items, choosing $d = 2$ yields a load balance of within $\log\log N$ factor of optimal, and when $d = \Theta(\log N)$, the load balance is within a constant factor of optimal.

The power of two choices can also be applied to node insertion [16, 17]. When a new node joins, a three-step process is performed: (1) Generation of d hash values in the DHT identifier space according to some algorithm; (2) Sampling of the DHT nodes responsible for these hash values; and (3) Splitting the load of the most heavily loaded DHT node according to some algorithm (e.g., even split or split proportionally to their capacity) and giving a load share to the new node. Take Chord as the DHT example. Assume the most heavily loaded node is X which is currently storing n items, $v_1, \cdots, v_n$, sorted in increasing order of keys. The new node will be assigned an ID which is an immediate successor of $v_{\frac{n}{2}}$ on the Chord ring, thereby taking over half of X's items. In other words, the new node will become X's new predecessor node on the Chord ring by taking over half of X's load.

3.3 Address-Space and Item Balancing

Karger et al. [6] presented two load balancing protocols whose provable performance guarantees are within a constant factor of optimal. One is address-space balancing and the other is item balancing.

Address-space balancing aims to balance the distribution of the key address space to nodes in the settings where items are uniformly mapped to the DHT nodes via consistent hashing. Each node has a fixed set of $O(\log N)$ virtual servers and it chooses exactly one of the virtual servers to be active at any time. The selection of the active virtual servers is determined as follows: Each node occasionally identifies which of its $O(\log N)$ virtual servers spans the smallest address (which is defined in [6]) and activates the particular virtual server.

Item balancing, on the other hand, balances the distribution of items to nodes in the settings where items are arbitrarily mapped to the DHT nodes. Let l_i denote the load on node i in terms of the number of items and ε is a constant ($0 < \varepsilon < 1$). Each node i occasionally contacts another node j at random. If $l_j \leq \varepsilon l_i$, then the nodes performs a load balancing operation, distinguishing two cases[1]:

[1] For $l_i \leq \varepsilon l_j$, the nodes performs similar operations.

- **Case 1:** $i = j+1$, namely, node i is node j's successor. Node j increases its ID such that $\frac{l_i - l_j}{2}$ items with the smallest keys get assigned from i to j. Both nodes end up with load $\frac{l_i + l_j}{2}$.
- **Case 2:** $i \neq j+1$. If $l_{j+1} > l_i$, then we set $i := j+1$ and go to case 1. Otherwise, node j moves between nodes $i-1$ and i to capture half of node i's items, while node j's old items are handled by its former successor node $j+1$.

3.4 Discussion

Conventionally, DHTs randomize item assignments to nodes via consistent hashing. Thus, the primary goal of load balancing in DHTs is to evenly partition the address space among the nodes. Virtual server-based approach [3, 4, 10, 14, 18, 19], which has each node host multiple virtual servers, ensures with high probability that the number of items on any node is within a constant factor of the average. To handle dynamic load, Zhu et al. [14] and Godfrey et al. [4] proposed to move virtual servers, the basic unit of load, among nodes to do load balancing. The main drawbacks of the virtual server-based approach are multiplicative increases of routing states and communication bandwidth. The address-space balancing [6], avoids the drawbacks by activating only one virtual server per node at any time. Another proposal, Y_0 [19], increases the size of routing tables by at most a constant factor with $\log N$ virtual servers per node.

The power of two choices [2], which assumes each item as the basic unit of load, takes a different approach by storing every newly inserted item onto the least loaded node among a set of randomly sampled nodes. Load movement is achieved by item re-insertions. The idea can also be applied to node insertions for load balancing [16, 17] by splitting the load of the most heavily loaded node from a set of sampled nodes and giving a load share to the new node.

The item balancing [6] addresses the load balancing issue in the presence of arbitrary distributions of items. Instead of moving load (represented by virtual servers in the virtual server-based approach or items in the power of two choices approach) among nodes, it moves nodes in the address space by re-assigning their IDs to balance load.

4 Case Study: Efficient, Proximity-Aware Load Balancing [14]

4.1 Overview

The load balancing algorithm discussed in the case study is essentially a virtual server-based approach, assuming that each DHT node hosts multiple virtual servers. Load re-assignment is performed via virtual servers. The first goal is to ensure fair load distribution by assigning load to each node proportional to its capacity. The second goal is to use proximity information to guide load re-assignment and transfer, thus minimizing load movement cost and making load balancing fast and efficient. The approach conceptually consists of three steps: Load balancing information (LBI) aggregation and dissemination, proximity-aware virtual server assignment (VSA), and virtual server transfer (VST). At the heart of the algorithm is proximity-aware VSA. For ease of exposition, the discussions are based on Chord. However, all the techniques discussed here are applicable or easily adaptable to other DHTs.

4.2 LBI Aggregation and Dissemination

The primary goal of LBI aggregation and dissemination is to gather system-wide LBI and then spread the aggregate across the nodes. By the LBI aggregate, heavily loaded nodes (heavy nodes for short) and lightly loaded nodes (light nodes for short) are identified, and then the heavy node knows which virtual servers to shed (similarly, the light node knows the amount of additional load to take) to ensure fair load distribution across nodes. The LBI aggregation and dissemination is based on a distributed *k*-ary tree [12] that is built atop of the underlying DHT.

The construction of the *k*-ary tree (KT) is a recursive process of splitting the underlying DHT identifier space. Each KT node is associated with a responsible region of the DHT identifier space while the root KT node owns the whole identifier space. In the beginning of building the *k*-ary tree, there is only the KT root node. The partitioning of the DHT identifier space starts from the KT root node. A KT node X's responsible region is split into k equal parts, each of which is taken by one of X's k children. That is, X's i-th child will be responsible for the i-th fraction of X's responsible region. A KT node is hosted by a virtual server that is responsible for the KT node's key which is produced by taking the center point of the KT node's responsible region of the identifier space. The KT grows as the partitioning continues in a top-down fashion until certain termination condition is met. That is, if a KT node X's responsible region is completely covered by that of X's hosting virtual server, then X is a KT leaf node and it stops growing children. The *k*-ary tree is self-repairing, fault-tolerant, and fully distributed.

The aggregation of LBI is performed along the KT in a bottom-up manner: Each KT node periodically, at an interval T, requests LBI from its children, while each KT leaf node simply asks its hosting virtual server to report LBI. Note that it is guaranteed that a KT leaf node will be planted in each virtual server. Thus, having each KT leaf node ask its hosting virtual server to report LBI can gather the LBI of the underlying DHT. Since a DHT node hosts multiple virtual servers, in order to avoid reporting redundant LBI of a DHT node, a DHT node (say i) randomly chooses one of its virtual servers to report LBI, in the form of $< L_i, C_i, L_{i,min} >$ (where L_i, C_i, and $L_{i,min}$ stand for the total load of virtual servers, the capacity, and the minimum load of virtual servers on the node i, respectively). Each KT node (say, X) runs the routine *KT_node_report_LBI*() as outlined in Fig. 1. This process is continued along the KT in a bottom-up fashion until the KT root node is reached. As a result, the KT root node produces a system-wide LBI $< L, C, L_{min} >$, where L, C, and L_{min} represent the total load, the total capacity, and the smallest load of virtual servers in the system, respectively.

```
procedure KT_node_report_LBI(KT_node X)
if (X is a KT leaf node) then
    $< L_x, C_x, L_{x,min} > \longleftarrow$ receive $< L_i, C_i, L_{i,min} >$ from X's hosting virtual server
else
    Receive $< L_i, C_i, L_{i,min} >$s from k children /* $i = 1, ..., k$ */
    $L_x \longleftarrow \sum_{i=1}^{k} L_i$
    $C_x \longleftarrow \sum_{i=1}^{k} C_i$
    $L_{x,min} \longleftarrow$ the smallest $L_{i,min}$
end if
if (X is not a KT root node) then
    Report $< L_x, C_x, L_{x,min} >$ to X's parent /* report to the parent node */
end if
```

Fig. 1 LBI aggregation algorithm.

After LBI aggregation, the KT root node disseminates the summary $< L, C, L_{min} >$ along the KT in a top-down fashion to each KT leaf node, which in turn distributes $< L, C, L_{min} >$ to its own hosting virtual server. As a result, all DHT nodes are guaranteed to have a copy of $< L, C, L_{min} >$. Let L_i denote the sum of the loads of all virtual servers on a DHT node i, and C_i represent the capacity of a DHT node i. Recall that one of the goals of the load balancing algorithm is to ensure fair load distribution over DHT nodes by assigning the load to a DHT node in proportion to its capacity. Let T_i denote the target load of a DHT node i proportional to its capacity. We have $T_i = (\frac{L}{C} + \varepsilon)C_i$ (ε is a parameter for a tradeoff between the amount of load moved and the quality of balance achieved. Ideally, ε is 0). Therefore, a DHT node i can be defined as:

- A heavy node if $L_i > T_i$.
- A light node if $(T_i - L_i) \geq L_{min}$.
- A neutral node if $0 \leq (T_i - L_i) < L_{min}$.

Each heavy node i determines a subset of its virtual servers $\{v_{i,1}, \cdots, v_{i,m}\}$ $(m \geq 1)$ that minimizes $\sum_{k=1}^{m} L_{i,k}$ subject to the condition that $(L_i - \sum_{k=1}^{m} L_{i,k}) \leq T_i$ (where $L_{i,k}$ denotes the load of the virtual server $v_{i,k}$ on the DHT node i). This subset of virtual servers, if removed, make the heavy node i to become light. Note that this choice of virtual servers on heavy nodes would minimize the total amount of load moved for load balancing throughout the system, thereby minimizing the load movement cost. Each light node j, one the other hand, is willing to take additional load $\Delta L_j = T_j - L_j$, without becoming overloaded.

4.3 Proximity-Aware VSA

VSA is performed along the KT in a bottom-up fashion. Each heavy node i, after determining the subset of its virtual servers to shed, i.e., $\{v_{i,1}, \cdots, v_{i,m}\}$, randomly chooses one of its virtual servers to report VSA information

$$< L_{i,1}, v_{i,1}, ip\_addr(i) >, \cdots, < L_{i,m}, v_{i,m}, ip\_addr(i) >$$

to its hosted KT leaf node, which in turn propagates the information upwards along the tree. Similarly, each light node j randomly selects one of its virtual server to report VSA information $< \Delta L_j, ip\_addr(j) >$ to its hosted KT leaf node, which in turn propagates the information upwards along the tree.

During the propagation of VSA information along the KT, each KT node (say X) runs the routine $KT\_node\_VSA()$ as described in Fig. 2. It collects the VSA information into VSA_pool from either its hosting virtual server or its children (lines $2-6$). If the number of VSA information reaches a predefined *paring threshold*, X would serve as a rendezvous point for virtual server assignment, by running the subroutine $KT\_node\_rendezvous\_point()$. This subroutine uses a best-fit heuristic approach[2] to perform virtual server assignment. The successfully assigned VSA information is sent back to corresponding DHT nodes for virtual server transferring, while the unpaired VSA information is propagated to X's parent. This VSA process is continued in a bottom-up fashion along the KT until it reaches the KT root node. Then, the KT root node serves as the last rendezvous point (without a pairing threshold constraint) for virtual server assignment.

As the VSA process proceeds along the KT in a bottom-up sweep, it recursively assigns virtual servers among DHT nodes scattered in an increasingly larger contiguous portion of the DHT identifier space[3] until the whole DHT identifier space (for which the KT root node is responsible). In other words, the VSA process is *identifier space-based* in that the virtual server assignments are performed earlier among those DHT nodes which are closer to each other in the DHT identifier space. The VSA process is *proximity-oblivious* because the logical closeness in the DHT identifier space does not necessarily reflect physical closeness of DHT nodes.

The basic idea behind proximity-aware VSA is that proximity information is utilized to guide VSA such that virtual server assignments are prioritized among physically close heavy nodes and light nodes, localizing VST and thus saving bandwidth. Recall that proximity-oblivious VSA, performed along the KT in a bottom-up sweep, ensures that virtual servers are assigned earlier among DHT nodes if they are logically closer to each other in the DHT identifier space. The key to proximity-aware VSA includes that (1) it first maps the VSA information of heavy and light nodes onto the DHT overlay such that the VSA information of physically

[2] Computing an optimal re-assignment of virtual servers from heavy nodes to light nodes is NP-complete. We use this simple greedy algorithm to find an approximation solution to minimize the load to be re-assigned and moved.

[3] Note that here the location of a DHT node in the identifier space is represented by its randomly chosen virtual server.

```
procedure KT_node_VSA(KT_node $X$)
$VSA\_pool \longleftarrow \emptyset$
if ($X$ is a $KT$ leaf node) then
    $VSA\_pool \longleftarrow$ VSA information from $X$'s hosting virtual server /* receive the VSA information from its hosting virtual server */
else
    $VSA\_pool \longleftarrow$ VSA information from $k$ children
end if
if ($VSA\_pool.size \geq pairing\_threshold$ || $X$ is a $KT$ root node) then
    $KT\_node\_rendezvous\_point(X, VSA\_pool)$ /* $X$ serves as a rendezvous point*/
else
    Report $VSA\_pool$ to $X$'s parent /* propagate the VSA information to its parent */
end if
procedure KT_node_rendezvous_point(KT_node $X$, VSA information $pool$)
$light\_list \longleftarrow$ remove all $< \Delta L_j = T_j - L_j, ip\_addr(j) >$s from $pool$ /* $light\_list$ maintains the VSA information of light nodes */
$heavy\_list \longleftarrow$ remove all $< L_{i,r}, v_{i,r}, ip\_addr(i) >$s from $pool$ /* $heavy\_list$ maintains the VSA information of heavy nodes */
$pool \longleftarrow \emptyset$
Case ($heavy\_list \neq \emptyset$)
    Remove the most loaded virtual server $v_{i,r}$ from $heavy\_list$, and assign it to a DHT node $j$ in $light\_list$ such that $\Delta L_j$ is minimized and subject to the condition that $\Delta L_j \geq L_{i,r}$
    if ($v_{i,r}$ can be assigned) then
        Remove $< \Delta L_j, ip\_addr(j) >$ from $light\_list$
        if ($\Delta L_j - L_{i,r} \geq L_{min}$) then
            Insert $< \Delta L_j - L_{i,r}, ip\_addr(j) >$ into $light\_list$
        end if
        Send the assigned information $< v_{i,r}, ip\_addr(i), ip\_addr(j) >$ to DHT nodes $i$ and $j$ /* prepare for virtual server transfer */
    else
        $pool \longleftarrow pool \cup \{< L_{i,r}, v_{i,r}, ip\_addr(i) >\}$
    end if
end Case
$pool \longleftarrow pool \cup light\_list$
if ($pool.size > 0$ && $X$ is not a $KT$ root node) then
    Report $pool$ to $X$'s parent /* report un-assigned VSA information to its parent */
end if
```

Fig. 2 Virtual server assignment algorithm.

close nodes are close together in the DHT identifier space, and (2) then each virtual server in the VSA process propagates upwards the VSA information which has been mapped to it, instead of the VSA information of the DHT node it belongs to. Consequently, the bottom-up VSA process along the KT naturally guarantees that virtual servers are assigned earlier among physically closer heavy nodes and light nodes.

Leveraging on landmark clustering [7], each DHT node i measures distances to a set of m (e.g., m with a typical value of 15) distributed landmark nodes, obtaining a landmark vector $< v_{i,1}, \cdots, v_{i,m} >$. Conceptually, DHT node i is mapped into a point in a m-dimensional Cartesian space (which we call *landmark space*) by having the landmark vector as its coordinates. Two physically close DHT nodes (say, i and j) would have similar/close landmark vectors and be close to each other in the landmark space. However, the landmark space is usually of relatively high dimension while the DHT identifier space is of low dimension (e.g., 1-dimension in Chord and Pastry), so it is infeasible simply using the landmark vector as the key to map the VSA information of DHT nodes onto the DHT overlay. Space filling curves such as Hilbert curve [1] are used to generate the key from a landmark vector while preserving proximity relationships. That is, if two nodes have close landmark vectors, the resulting two keys are same or close. In other words, if two points (dictated by two landmark vectors) are close together in the m-dimensional landmark space, by space filling curves they will be mapped to two points that are also close together in the one-dimensional space, i.e., proximity is preserved by the mapping.

To summarize, proximity-aware VSA works as follows. Each heavy/light node independently generates a landmark vector which is used to derive a key by space filling curves. Each DHT heavy/light node (say i) then publishes its VSA information (e.g., $< L_{i,r}, v_{i,r}, ip\_addr(i) >$/$< \Delta L_j = T_j - L_j, ip\_addr(i) >$) onto the DHT overlay with the derived key. As such, the VSA information published by physically close nodes will be stored close together in the DHT identifier space. Given the published VSA information by all participating DHT heavy and light nodes, the proximity-aware VSA differs from the proximity-oblivious VSA in that each individual virtual server independently reports the VSA information (if any) which has been mapped into its responsible region of the DHT identifier space[4] to the KT leaf node which it hosts. In case of multiple KT leaf nodes planted in a virtual server, the virtual server reports the VSA information to only one of its KT leaf nodes to avoid sending redundant information. As a result, each KT leaf node has the VSA information of DHT heavy nodes and light nodes which are physically close together unless no VSA information is reported by its hosting virtual server. If the KT leaf node has the VSA information, it performs the following operations:

- If the number of the VSA information reaches the predefined pairing threshold, it can immediately serve as a rendezvous point for the virtual server assignments by running the routine $KT\_node\_rendezvous\_point()$, as outlined in Fig. 2.
- Otherwise, it propagates the VSA information to its KT parent node.

Then, the VSA process proceeds along the KT in a bottom-up sweep, as described in the proximity-oblivious VSA. Note that a k-ary subtree covers a contiguous portion of the DHT identifier space and the subtree's root node may serve as a rendezvous point for virtual server assignments, so this bottom-up VSA process naturally guarantees that the virtual servers are assigned among DHT nodes, recursively in a decreasing physical closeness order as the rendezvous point moves up along the KT.

4.4 VST

The VST process is quite simple and straightforward. Upon receiving the paired VSA information (say, $< v_{i,r}, ip\_addr(i), ip\_addr(j) >$), the heavy DHT node i will

[4] In the proximity-oblivious VSA, one of a DHT node's virtual servers instead is randomly chosen to report the DHT node's own VSA information.

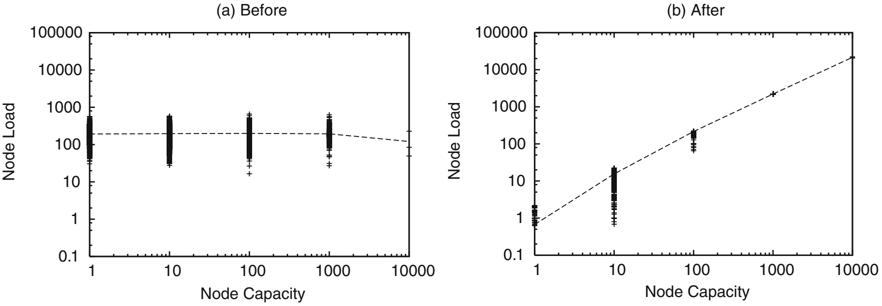


Fig. 3 Load scatterplot with respect to node capacity for Gaussian distribution of virtual server load and Gnutella-like node capacity profile. (**a**) shows scatterplot before load balancing and (**b**) shows scatterplot after load balancing

transfer its virtual server $v_{i,r}$ to the light DHT node j. Note that the VST process can be performed partly overlapping with the VSA process for fast load balancing. The transferring of a virtual server unavoidably causes the KT to restructure because the KT node which is planted in the virtual server has to migrate with the virtual server. In order to keep the KT relatively stable, a lazy migration protocol could be adopted for the KT node. Only after the transferring of a virtual server is fully completed or after the whole VSA process is fully completed will the KT node migrate. Note that the restructuring of the KT is fully distributed and inexpensive because each KT node migration only involves at most $k+1$ messages (one message with the parent and k messages with the k child nodes).

4.5 Discussion

As an example of virtual server-based approach, this load balancing algorithm strives to achieve two goals. The first goal is to balance load distribution by assigning load to each node proportional to its capacity. Figures 3, 4 show the scatterplot of node load before and after load balancing for different virtual server load distributions and node capacity profiles. The algorithm successfully assigns load to nodes proportional to their capacity. The second goal, which distinguishes this algorithm from others, is to minimize load movement cost by localizing virtual server assignments and transfer. Zhu et. al [14] have shown, via detailed simulations, that this algorithm can reduce bandwidth cost in load movement over proximity-oblivious algorithms by 11-65%, for various combinations of physical topologies of nodes comprising the DHT overlay, virtual server load distributions, and node capacity profiles.

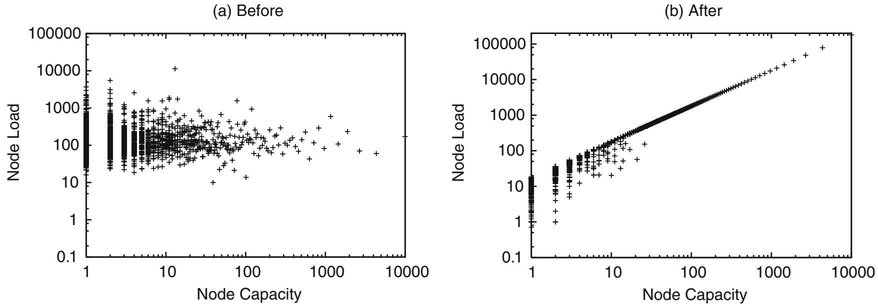


Fig. 4 Load scatterplot with respect to node capacity for Pareto distribution of virtual server load and Zipf-like node capacity profile. (**a**) shows scatterplot before load balancing and (**b**) shows scatterplot after load balancing

5 Future Research

The following research topics remain unexplored for load balancing in P2P networks.

Evaluation framework. Numerous load balancing algorithms exist for the structured P2P network, with different design goals and assumptions. One possible future research topic is to build a framework to comprehensively understand and compare the performance of the different load balancing algorithms. The framework needs to base evaluation on a list of metrics, which may include quality of load balancing (i.e., minimize load imbalance across nodes), responsiveness/convergence time (i.e., how fast does the algorithm react to load imbalance and achieve balance), cost (i.e., bandwidth cost in load movement/migration), and failure resilience. Due to heavy-tailed query distributions, high churn rates, and wide variation in node capacity found in the P2P network, it is challenging for a load balancing algorithm to fulfill the aforementioned list of metrics, which are often conflicting. A framework helps system designers understand tradeoffs in different algorithms and identify what tradeoffs and goals in algorithms are specifically critical to the structured P2P network.

Balancing multiple resources. Existing algorithms assume only one bottlenecked resource. However, a system could be constrained in more than one resource, e.g., in both bandwidth and storage. To make an algorithm capable of handling multiple resources, we need to properly quantify the load of each data item on the multiple resources. For example, we may represent the load of each data item by a load vector where each element corresponds to a resource. Existing algorithms need to be adapted or new algorithms need to be designed for balance of multiple resources.

Secure load balancing. The open and anonymous nature of P2P networks opens the door to various attacks. Few algorithms, if any, consider security against attacks. For example, a node may claim itself a heavily loaded node and shed its load to other nodes, raising the problem of free riding. On the other hand, malicious nodes can disguise themselves as lightly loaded nodes and steals away data items from heavily loaded nodes, thereby causing the problem of data availability. Worse, they can launch denial-of-service attacks and serve malicious data.

6 Summary

This chapter addressed the importance and necessity of load balancing in structured P2P networks, due to the three main reasons. First, structured P2P networks assume uniform peer capacities while peer capacities are heterogeneous in deployed P2P networks. Second, resorting to pseudo-uniformity of the hash function used to generate node IDs and data item keys leads to imbalanced overlay address space and item distribution. Finally, placement of data items cannot be randomized in some applications (e.g., range searching). We discussed tree structure-based approach and gossip-based approach, two main techniques in load information aggregation and dissemination, which are used by many load balancing algorithms to monitor and react to continuously changing system status. We also presented three load balancing algorithms, namely virtual server-based approach, power of two choices, and address-space and item balancing. While different in their designs, either moving loads (expressed by virtual servers or items) or moving nodes, they all aim to improve balance on the address space and data item distribution in DHTs. As a case study, we discussed an efficient, proximity-aware load balancing algorithm that is essentially a virtual server-based approach, not only achieving fair load distribution over nodes but also minimizing load balancing cost in bandwidth. We finally presented three possible research directions on load balancing.

References

1. T. Asano, D. Ranjan, T. Roos, E. Welzl, and P. Widmaier, “Space filling curves and their use in geometric data structure,” *Theoretical Computer Science*, vol. 181, pp. 3–15, July 1997.
2. J. Byers, J. Considine, and M. Mitzenmacher, “Simple load balancing for distributed hash tables,” in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, Feb. 2003.
3. F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, “Wide-area cooperative storage with CFS,” in *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP ’01)*, (Banff, Canada), pp. 202–215, Oct. 2001.
4. B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, “Load balancing in dynamic structured P2P systems,” in *Proceedings of IEEE INFOCOM*, pp. 2253–2262, Mar. 2004.
5. M. Haridasan and R. van Renesse, “Gossip-based distribution estimation in peer-to-peer networks,” in *Proceedings of IPTPS*, (Tampa Bay, FL), Feb. 2008.
6. D. R. Karger and M. Ruhl, “Simple efficient load balancing algorithms for peer-to-peer systems,” in *Proceedings of IPTPS*, Feb. 2004.

7. S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, “Topologically-aware overlay construction and server selection,” in *Proceedings of IEEE INFOCOM*, (New York, NY), pp. 1190–1199, June 2002.
8. A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,” in *Proceedings of the 18th IFIP/ACM International Conference on Distributed System Platforms (Middleware)*, (Heidelberg, Germany), pp. 329–350, Nov. 2001.
9. S. Saroiu, P. K. Gummadi, and S. D. Gribble, “A measurement study of peer-to-peer file sharing systems,” in *Proceedings of Multimedia Computing and Networking(MMCN)*, (San Jose, CA), Jan. 2002.
10. I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” in *Proceedings of ACM SIGCOMM*, (San Diego, CA), pp. 149–160, Aug. 2001.
11. P. Yalagandula and M. Dahlin, “A scalable distributed information management system,” in *Proceedings of ACM SIGCOMM*, pp. 379–390, Aug. 2004.
12. Z. Zhang, S. Shi, and J. Zhu, “SOMO: self-organized metadata overlay for resource management in p2p DHT,” in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, Feb. 2003.
13. B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, “Tapestry: An infrastructure for fault-tolerance wide-area location and routing,” Tech. Rep. UCB/CSD-01-1141, Computer Science Division, University of California, Berkeley, Apr. 2001.
14. Y. Zhu and Y. Hu, “Efficient, proximity-aware load balancing for DHT-based P2P systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 4, pp. 349–361, 2005.
15. D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, “Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web,” in *Proceedings of the 29th ACM Symposium on Theory of Computing (STOC)*, pp. 654–663, May 1997.
16. J. Ledlie and M. Seltzer, “Distributed, secure load balancing with skew, heterogeneity, and churn,” in *Proceedings of IEEE INFOCOM*, Mar. 2005.
17. X. Wang and D. Loguinov, “Load-balancing performance of consistent hashing: asymptotic analysis of random node join,” *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 892–905, 2007.
18. D. Wu, Y. Tian, and K. Ng, “On the effectiveness of migration-based load balancing strategies in DHT systems,” in *Proceedings of IEEE ICCCN*, pp. 405–410, Oct. 2006.
19. P. Godfrey and I. Stoica, “Heterogeneity and load balance in distributed hash tables,” in *Proceedings of IEEE INFOCOM*, Mar. 2005.

Acyclic Preference-Based Systems

A Self-Stabilizing Approach for P2P Systems

Fabien Mathieu

Abstract Acyclic preference-based systems recently appeared as an elegant way to model many P2P unstructured or hybrid systems. The strength of these systems is a self-stabilizing property that allows to provide analytical results in addition to empirical validation. In this chapter, we give the keys to understand and use acyclic preference-based. After a brief introduction to the roots and notation of the model, the main convergence theorem is exposed and a description of P2P-relevant preferences is given. Ensues an explanation of the combinatory techniques used to refine the convergence study, and a statistical description, based on mean field assumptions, of the stable configurations of acyclic preference-based systems. For some preferences, we observe that the stable configuration is a small-world.

1 Introduction

During the last decade, the increase of Internet users' resources (storage, network access, CPU, ...) and the advances in multimedia coding have made the download of large files, including audio and video content, feasible with only the help of peer-to-peer (P2P) exchanges. As a result, P2P file-sharing networks like KaZaA, eMule [10, 11] or BitTorrent [8] are now widely popular.

Following this success, P2P technology proposes more and more functionalities, streaming being one of the most noticeable. Live streaming applications like CoolStreaming [55], PPLive [21], SopCast [48], TVants [53], or UUSee [54] become more and more popular, as do Video-on-Demand applications [3, 27, 31].

All these P2P applications rely on the same paradigm: if you want to serve a large amount of users with little or no server capacity, you have to take the resources where they are present, that is from the users themselves.

Fabien Mathieu
Orange Labs, 38–40 rue du général Leclerc, 92794 Issy-les-Moulineaux, France,
e-mail: fabien.mathieu@orange-ftgroup.com

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_42, © Springer Science+Business Media, LLC 2010

In other words, the key to make a P2P system work is to answer the following question: *who gets what from whom?*

While the answer is generally straightforward for classical server/client architectures, it becomes a major challenge for P2P networks, which are decentralized by nature. Of course, many other questions can be sources of trouble, such as availability, fairness, robustness ... The heterogeneous and dynamic nature of the peers also add to the difficulty.

There are roughly two basic approaches to answer the *who gets what from whom?* question: structured or unstructured (so-called *hybrid* approaches try to combine both of them).

The aim of the structured approaches is to fit the peers within a pre-determined structure. The structure is a guide that tells the peers how to behave. Many structured solutions have been proposed in the context of distributed hash tables (DHTs) [19, 40, 43, 47, 50, 56], which are used to locate content in a P2P system. Structured solutions have also been proposed to solve the P2P live streaming issue [5, 14].

The structures themselves are generally graphs, like trees, interior-node-disjoint trees [5], de Brujin graphs [14, 19], regular graphs, ... They can be built on top of a topological structure (ring [40, 50], d-torus [43], hypercube [40, 47, 50, 56], ...).

The main strength of structured approaches is that the behavior of the system is determined by a known structure, which can be designed to ensure optimal or quasi-optimal performance. However, maintaining a structure when there is churn (arrivals and departures of peers) can be costly. Moreover, most structures assume that peers have similar capacities, and all but a few proposals [51] have difficulties to handle heterogeneous peers.

On the other hand, the idea behind unstructured approaches is to let the peers take decisions based on their own view of the system: there is no explicit structure, but an implicit one resulting in a sum of local choices. Note that the term *self-structured* may be more appropriate than *unstructured*. Self-structured approaches are mostly employed in content distribution (e.g., filesharing [17, 42], streaming [4]), and are hybrided in some DHTs[19, 40].

By design, self-structured approaches should be robust to churn and heterogeneity. But, as their structure is not known *a priori*, it can be difficult to predict their performance, and the validation of a self-structured system is often empirical only. However, some progress has been made in order to give some theoretical grounds [4, 42] to self-structured systems.

The aim of this chapter is to present and give insight into one of the models for self-structured systems: *preference-based systems* (p.b.s.). P.b.s. have been introduced to model the dynamics of collaborations between peers acting independently according to some personal preferences.

1.1 Stable Marriage: A Brief Overview

The roots of p.b.s. come from Gale and Shapley's *stable marriage theory*, which has been adapted to handle P2P networks' specific characteristics.

The most famous (often considered as *recreational mathematics*) version of the problem appeared in 1962 in the seminal paper [20]. The problem is to try to match some men and women in a way such that no adultery can happen. Formally, an instance of the Stable Marriage problem involves two sets M and W (men and women), and assumes that each member of each gender has strict preferences over some members of the opposite gender. A configuration of the instance is a matching between M and W. It is unstable if there is a pair (m,w) not in the given configuration such that each one prefers being matched with the other rather than being like it is in the current configuration. Instability means possible adultery because both m and w have interest in eloping together. The goal of the theory is to find and describe the stable configurations of the instance. It has been proved that if there is no preference tie, a stable configuration always exists, which can be computed using a proposal/disposal algorithm.

An extension of the problem, known as *college admissions*, allows participant from one set to have more than one mate from the other set. Its applications are, for instance, affecting graduating medical students to their first hospital appointment, or young assistant teachers to universities [24, 45]. Like the stable marriage problem, a stable configuration always exists.

Then, the case of interest for P2P modeling is when only one set of participants is evolved. This variant is known as the *Roommates* problem when only simple matchings are allowed, and b-matching otherwise (where b is the standard notation used to describe matching quotas) [12, 23, 25]. It has applications in many areas, for instance in the pairwise kidneys exchange program [46]. The main issue with this variant is that the corresponding systems might admit no stable configuration, whatsoever. Note that centralized algorithms give a stable configuration if one exists [6, 22, 23, 52].

1.2 Going P2P

The interest in the matching theories for P2P systems is rather intuitive: given a set of peers, each one having a list of neighbors for potential connections, the theory can give some insight into these connections under a scheme where each peer tries to maximize its own benefits.

As the stable marriage theory and its b-matching variant have been widely studied for almost fifty years, one could think modeling P2P with it should be straightforward. In practice, it offers a steady theoretical background, but P2P networks have specificities that force a new perspective on the problem. Thus the need of introducing preference-based systems.

The details of differences between standard b-matching and preference-based systems are the following:

- most of the time, preferences in b-matching studies are supposed to be arbitrary. On the other hand, P2P preferences are mostly based on objective values.
- b-matching mainly cares about the existence of stable configurations, and how to compute them. Most P2P preferences fall under a powerful existence/uniqueness/self-stabilization theorem which makes such questions irrelevant.
- dynamics and performance of configurations are fundamental for P2P, while not extensively studied in b-matching research.

1.3 Outline

This chapter is structured as follows: in Section 2,we give the basics of preference-based systems. This includes the notation, the main convergence theorem for acyclic p.b.s., and a description of P2P-relevant preference classes. Section 3 is then devoted to deepen the convergence theorem in order to give tighter convergence bounds depending on the preference class. Section 4 describes the stable configurations of acyclic systems. Lastly, Section 5 concludes the chapter.

2 Acyclic Preference-Based Systems

This section sets the main framework: notation and definitions are presented in Section 2.1; the main theorem for acyclic preferences is then stated in Section 2.2, and lastly Section 2.3 gives a description of the set of acyclic preferences.

2.1 Notation and Definition

We present here the standard notation used for handling preference-based systems [15, 17, 33, 37, 38]. A large part of it is based on the literature on b-matching [6, 9].

An instance of the problem is composed of a set of peers, with an acceptance graph, preferences and quotas. A configuration is a valid set of effective collaborations, and the dynamics are modeled by a process called *initiative*, where peers try to improve their collaborations in a selfish, decentralized, manner.

2.1.1 Preference-Based Systems

A preference-based system consists of a set P of n peers (or nodes), whose possible interactions are described by an acceptance graph G, a mark matrix m and a quota vector b.

The acceptance graph $G = (V,E)$ is an undirected graph. It describes compatibilities: a peer i and a peer j are capable of collaborating (we say that i is acceptable for j, and vice versa) if, and only if (iff) $\{i, j\} \in E$. For instance, the acceptance graph can represent the peers' knowledge of other peers, the existence of common interests between peers, or a pre-existing overlay graph. Although some of the results we provide are true for any kind of acceptance graph, we mainly consider Erdös-Rényi graphs $\mathscr{G}(n,p)$ in this chapter (each possible edge exists with probability p independently of the others; hence the expected degree is $d = p(n-1)$). The advantage of using $\mathscr{G}(n,p)$ graphs is that theoretical analysis is made easier.

The mark matrix m defines how peers value each other. It is used for determining the preferences. For $\{i, j\} \in E$, $m(i, j)$ denotes the mark of j for i. If $m(i, j) < m(i,k)$, then we say that i prefers j to k. We assume that there is no preference tie, i.e., that all marks within a row are distinct.[1] The sign of the inequality is arbitrary, and has no impact on the model. By default, we assume lower marks are preferred (which is suitable for distance or latency marks), but for marks like bandwidths, it may be the opposite. The matrix m can have various shapes (cf Section 2.3), which affect the resulting preferences. Note that a peer is not necessarily aware of the full mark matrix. In fact, it may not even be aware of all the values of its corresponding row in the matrix. However, we assume that a peer can always find out the value of a given neighbor by contacting it.

Lastly, the quota vector b limits the collaborations: a peer i cannot have more than $b(i)$ simultaneous collaborations. Without loss of generality, one can assume that $b(i)$ is not greater than the degree of i in the acceptance graph. For convenience, we often assume that b is a constant in this chapter, although the model allows any kind of integer distribution.

2.1.2 Configurations

The state of collaborations in a preference-based system is called a configuration. Formally, a configuration C is a subset of E. The neighbors $C(i)$ of i in C are called *mates* of i. If $c(i)$ denotes the number of mates for peer i, the existence of quotas implies $c(i) \leq b(i)$ for all $i \in P$. If $c(i) < b(i)$, we say that i is *undermated.*

The evolution of a configuration comes from the resolution of *blocking edges*. A blocking edge is an edge e between two neighbors that are not matched together ($e \in E \setminus C$), such that both could gain by collaborating (even at the price of possibly dropping an existing collaboration). Blocking edges are the equivalent of possible

[1] Ties in the preferences raise many issues ([24–26, 35, 36, 44]) that will not be discussed in this chapter. Note that ties can be easily avoided if we assume that the peers possess distinct ids that are used to break any tie that may happen.

adulteries in the original marriage theory. Formally, an edge $e = \{i, j\} \in E \setminus C$ is blocking iff the two following conditions are verified:

- $c(i) < b(i)$ (i is under-mated), or $\exists k \in C(i), m(i, j) < m(i, k)$ (i prefers j to one of its mate);
- $c(j) < b(j)$ or $\exists k \in C(j), m(j, i) < m(j, k)$ (symmetrical to the previous condition).

A peer that belongs to at least one blocking edge is *eligible*. A configuration with no blocking edge (or equivalently no eligible peer) is *stable*.

2.1.3 Initiatives

In a preference-based system, the resolution of blocking edges is local: peers independently try to find and resolve a blocking edge they belong to. This scheme is called *initiative*, and we assume that it is atomic. Initiatives are the elementary steps of the system evolution. An initiative is *active* if it succeeds in resolving a blocking edge.

An initiative scheme is formally described as a (possibly random or time varying) selection function from V to V that gives for any peer i a neighbor to try. The selection can be restricted to blocking neighbors if the peer knows the blocking edges it belongs to. Otherwise the choice is made among all neighbors non adjacent to i in C. Typical initiative schemes are best mate (best blocking neighbor selection), decremental (round-robin-like selection), or plain random. Both best mate and decremental initiatives require a complete knowledge of the neighbors' ranking, whereas random initiative only requires to evaluate a peer's value on the fly. Note, that hybrid initiative schemes can be designed in order to increase the performance of a preference-based system (cf [38], or Section 3.2).

Starting from an initial configuration, the sequence of initiatives performed by peers during time will entirely define the system evolution. Standard sequences are for instance round-robin, which models a peer mono-periodic activity (each peer performs a periodic initiative, the time period being the same), or uniformly random, which models an homogeneous Poisson activity (each peer follows a Poisson process, the intensity being the same). We can also consider adversarial sequences, which are built in order to exhibit worst cases. By convention, the time unit (t.u.) is the time needed to perform a sequence of n initiatives (in other words, the time cost of each atomic initiative is $\frac{1}{n}$ t.u.). This is a convenient scaling, because t t.u. corresponds to an average of t initiatives per peer.

Another time measure for the initiative is the round. A round is a sub-sequence where each peer that was eligible at the beginning of the round either takes an initiative or ceases to be eligible during the round. For a round-robin sequence, a time unit is a round, but this is not true in the general case.

2.1.4 Nodes and Edges' Stability

The initiative process gives a new definition of stability: a configuration C is *stable* if the only configuration reachable from C using any sequence of any initiative scheme is C itself. This definition is equivalent to the one given in Section 2.1.2 (no blocking edge). The advantage of the initiative approach of stability is that it can be defined at a local level: an edge e of a configuration C is *stable* iff e exists in all reachable configurations. In other words, a stable edge is a collaboration that initiatives cannot break.

Whenever a node is adjacent to a stable edge, its degree of mating freedom is lessened. The *free quota* of a node i, denoted $b'(i)$, is the difference between $b(i)$ and the stable edges adjacent to i. Free quotas form a configuration-dependent vector that can only decrease with time. They allow to define stability for a node: a node i is *stable* (or inactive), if $b'(i) = 0$ or if all the non-stable neighbors of i share a stable edge with it.

The last definitions which are useful for stability are warmness and hotness. For a given configuration C, an un-matched edge $e = \{i, j\}$ of the acceptance graph ($e \in E \setminus C$) is *warm* iff:

- j belongs to the $b'(i)$ best non-stable neighbors of i,
- i belongs to the $b'(j)$ best non-stable neighbors of i.

Hotness is a more restrictive condition: an un-matched edge $e = \{i, j\}$ between two non-stable neighbors is hot with respect to C iff i is the best non-stable neighbor of j and vice versa.

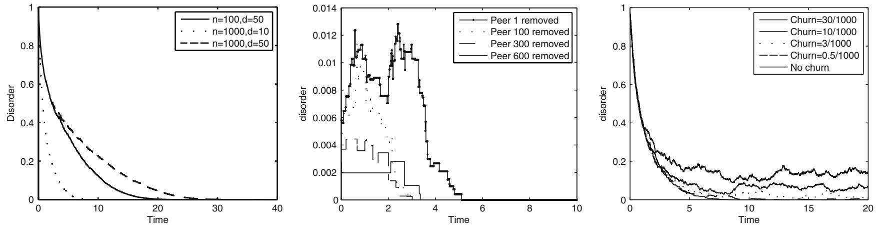
By extension, we call warm node (resp. hot node) a peer adjacent to a warm edge (resp. hot edge). Intuitively, warm and hot edges are *ready-to-be-stabilized* edges. They are a central concept in acyclic preference-based systems, because of the following Lemma 1, which is used to demonstrate many properties of the acyclic p.b.s.:

Lemma 1 ([16, 33]). *If C is a non-stable configuration of a given acyclic p.b.s., there is at least one hot edge with respect to C.*

2.2 Acyclicity: Main Convergence Theorem

A *preference cycle*, or *Kieschnick* cycle, is a cycle of $k \geq 3$ peers $i_1, \ldots, i_k$, such that each one prefers its successor to its predecessor: i_1 prefers i_2 to i_k, i_2 prefers i_3 to $i_1, \ldots, i_k$ prefers i_1 to i_{k-1} (or equivalently $m(i_1, i_2) < m(i_1, i_k)$, $m(i_2, i_3) < m(i_2, i_1)$, $\ldots, m(i_k, i_1) < m(i_k, i_{k-1})$).

In this chapter, we are interested in acyclic preference-based systems, that is p.b.s. which contain no preference cycle. Note that the acyclicity property is entirely defined by the acceptable values of the mark matrix, and is independent from quotas or initiatives. By convention, we call *acyclic* a mark matrix that produces an



(a). Static p.b.s.

(b) Atomic alteration

(c) Churn

Fig. 1 The self-stabilization property for three basic scenarios

acyclic p.b.s. The classification of the acyclic mark matrices, and their link with P2P systems will be deepened in Section 2.3.

Acyclic p.b.s. are characterized by a property that makes them unique among other p.b.s.:

Theorem 1 ([16]). *An acyclic preference-based system always admits a unique stable configuration, which is self-stabilizing under initiatives.*

Sketch of proof: The complete proof, available in [16], is in two steps: first, showing the self-stabilization, which proves the existence of a stable configuration, then the uniqueness.

The self-stabilization is a consequence of the irreversibility of the configuration evolution: given any sequence of initiatives, the corresponding sequence of configuration is irreversible. In other words, if the configuration evolves from C_1 to $C_2 \neq C_1$ at some time t, a consequence of the acyclicity is that $C \neq C_1$ for all configurations C that occur after t. Because the number of possible configurations is finite, any p.b.s. eventually reaches a configuration from which it cannot evolve, i.e., a stable configuration.

The uniqueness is proved by contradiction: If we consider two distinct stable configurations A and B, and a peer i with different mates in A and B, we can construct a preference cycle starting from i. □

Before pursuing further, let us illustrate the practical importance of Theorem 1 with an example, taken from [18]: in Fig. 1, we can see the evolution of a p.b.s. configuration with time. In details, we observe the distance between the actual and stable configurations (for details on the parameters and distance used, see [18]). Three distinct scenarios are considered:

Static case Fig. (1a) In a first set of simulations, a p.b.s. is set starting from the empty configuration, $C_\emptyset$. We observe the convergence towards the stable configuration stated by Theorem 1;

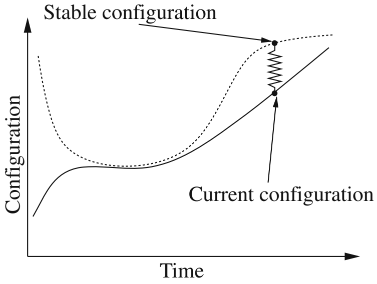


Fig. 2 The *spring* metaphor: because of the self-stabilizing property, the current configuration is continuously attracted to the stable configuration, as if there was a spring between them. This is true even if the stable configuration evolves during time

Atomic alteration Fig. (1b) In the second set of simulations, a p.b.s. is in its stable configuration, then a peer is removed from the system. This alteration changes the stable configuration. The system then converges towards its new stable configuration. We can see that the new stable configuration stays close to the old one, and that the system converges back quickly. Note that some peers induce more disorder than others when removed. This comes from a sort of *domino* effect (for more details, see [18]).

Continuous churn Fig. (1c) Finally, we consider continuous churn: peers are removed or introduced in the system according to a *churn* intensity. We observe that the system cannot cope with the evolving stable configuration, especially for high churn intensity. However, the self-stabilization manages to keep the system close to the stable configuration (the distance is roughly proportional to churn intensity). Note that if the churn stops at some time, then the system converges like in the other experiments.

This simple example suggests that the configuration of a p.b.s. should be always *close* to the stable configuration, because of the self-stabilization. This is the *spring* effect (cf Fig. 2).

Thus the Theorem 1 justifies a two-step approach for analyzing the performance of acyclic preference-based systems:

How fast is the self-stabilizing process? The proof of the theorem only gives the number of possible configurations as a bound, which is far too high to be of a practical use: this number, also known as the number of involutions of a set of size n, has a factorial-like asymptotic behavior [7]. Thus there is a need for tighter bounds in order to evaluate the *force constant* of the *spring*.

What is the stable configuration? If the self-stabilization is fast enough compared to the characteristic time of the system and the possible churn intensity, then the stable configuration will give a good approximation of the effective configurations of the system. Therefore the generic properties of stable configurations can be used to estimate the performance of the system.

These two questions will be addressed in the next sections. But first let us take a closer look at the shape of the acyclic preferences.

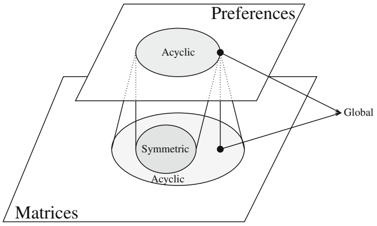


Fig. 3 Matrices and corresponding preferences: symmetric and global matrices form two distinct subsets of the acyclic matrices, but symmetric matrices can describe all acyclic preferences, including global preferences

2.3 Acyclic Classes

The set of the acyclic matrices is only a small subset of all possible mark matrices. In fact, if you pick up a mark matrix with random values, it is very likely that the corresponding preferences contain a cycle. For instance, if you consider any subset of 3 peers among n, with random marks between them, the probability that they form a preference cycle is $\frac{1}{4}$; thus the probability of having no preference cycle of length 3 is $(\frac{3}{4})^{\binom{n}{3}}$, which is very low for n large

On the other hand, many matrices with specific properties are acylic. For instance ([16]):

Global matrices A mark matrix m is global if all its rows are identical with respect to its acceptable entries (in other words, $m(i,k) = m(j,k)$ for all i, j that accept k). The corresponding preferences follow a total order among the nodes, and are thus called node-based (or just global).

Symmetric matrices A matrix m that verifies $m(i, j) = m(j,i)$ for all acceptable pairs $\{i, j\}$ is acyclic. Note that the corresponding preferences, which are edge-based, form the whole set of the acyclic preferences (i.e., any acyclic preference can be described by a symmetric mark matrix [1, 16]). The links between acyclic, global and symmetric matrices and the corresponding preferences are summarized by Fig. 3.

Linear combination of global and symmetric matrices Such matrix are acyclic as long as the linear combination produces no ties [15] (but as stated in Section 2.1.1, ties are easy to remove).

Let us remark that it is always possible to merge acyclic preferences together, if we use a representation by symmetric matrices and a linear combination.

2.3.1 P2P-Relevant Cases

Now that we have an idea of the possible acyclic p.b.s., let us see how standard P2P measurements fit into our framework.

Capacity A peer in a P2P network possesses many node-based values that are relevant for preferences: access bandwidth, storage or CPU capacity, average uptime,...For instance, the filesharing protocol BitTorrent [8] uses a Tit-for-Tat policy that makes a peer upload preferentially to the neighbors from whom it downloads the most. This can be seen as a preference system where the mark of a peer is its upload bandwidth (divided by its quota, i.e., its number of simultaneous upload connections).

Proximity All values that correspond to a sort of distance (in the physical network or in a virtual space) or similarity measure are edge-based by nature. For instance, many P2P systems try to minimize the latencies, such as the DHT Pastry [47] or online real-time gaming applications [34]. Similarly, massively multiplayer online games (MMOG) require connecting together players with nearby coordinates in a virtual space [13, 28, 29]. Some authors also propose to connect participants of a filesharing system according to their common interests [32, 49], which is a symmetric measure. The co-uptime, which is the average shared uptime, can also be a relevant measure for all collaborative applications.

Complementarity The measure of the difference between the resources of peers can also be valuable. For instance, a peer in a distributed storage application may want to find neighbors that are online when it is not, in order to keep its data available even when it is not connected. This is complementary uptime. Similarly, in a filesharing system like BitTorrent, where all participants want a same content which is divided into pieces, it may be convenient to collaborate with the peers possessing the most complementary set of file pieces. As a special case of a linear combination of a global and a symmetric value, complementary preferences are acyclic [16].

2.3.2 Considered Preferences

All acyclic preferences do not have the same self-stabilization and stable configuration. In order to exhibit generic and specific properties of these systems, we focus here on four classes of acyclic preferences:

Global As we said before, a node-based (or global) matrix is equivalent to a total order among the nodes. Therefore we do not need to explicit the mark matrix m for node-based preferences, and we can use an ordered node labeling instead. We arbitrarily choose $1,\dots,N$ as labels, 1 been the best (if 1 is ranked first for all nodes that accept 1, and so on. . .).

Geometric the nodes are associated to n points picked uniformly at random on a t-dimensional torus ($t \geq 1$). The (symmetric) marks are the distances between those points. These preferences allow a theoretical analysis of proximity-based performance.

Meridian latencies we considered random subsets of n nodes taken from the 2500 nodes dataset of the Meridian Project [41]. The marks are the (symmetric) latencies between those nodes. We do not perform analysis for those marks, but use them in Section 4.6.2 for validating the geometric approach.

Random acyclic preferences Each edge receives a random uniformly distributed value. The name is justified because as stated before, all acyclic preferences can be described edge-based marks. Hence uniformly distributed (symmetric) random marks are a convenient way to perform a uniform sampling of the acyclic preferences [1, 16].

3 Self-Stabilization Speed

In this section, we want to respond to the question *How fast is the self-stabilizing process?*. The answer, which is a good indicator of the strength of the self-stabilization process, is given theoretically when possible, and through simulations otherwise: while theoretical results can only be given for specific parameters (for instance, only *best mate* initiatives are investigated, and most of the tight convergence bounds only concern the case $b = 1$), simulations allow to investigate a broader set of situations.

3.1 Upper Bounds

We first propose to give upper bounds for the convergence time with best mate initiative. For the case $b = 1$, we show upper bounds, linear in rounds and exponential in number of initiatives. Both bounds are tight for some adversarial sequence. Then we propose a weaker round-bound for $b > 1$.

3.1.1 Convergence (Rounds)

For $b = 1$, the maximum time (in round) is given by Theorem 2:

Theorem 2 ([37]). *Starting from any configuration, an adversarial sequence takes at most $\lfloor \frac{n}{2} \rfloor$ rounds to converge. The bound is tight for a round robin sequence (and thus for any adversarial sequence).*

Proof. The proof relies on Lemma 1 (the existence of hot edges in any non-stable configuration). The best-mate initiative of a hot node stabilizes the corresponding hot edge, so after each round (until stabilization), at least one stable edge is formed. As $b = 1$, the stable configuration has at most $\lfloor \frac{n}{2} \rfloor$ edges, so the stabilization cannot last more than $\lfloor \frac{n}{2} \rfloor$ rounds. This bound is tight because it is reached for global preferences and complete acceptance graph, if we use a round robin sequence of initiatives with the pattern $n, (n-1), \ldots, 2, 1$, starting from the empty configuration $C_\emptyset$.

3.1.2 Convergence (Initiatives)

We now want to measure the convergence in term of active initiatives. The bound now depends whether the initiative sequence is round-robin or fully adversarial.

- For a round robin sequence, the convergence is given by Theorem 3:

Theorem 3 ([37]). *Starting from any configuration, a round robin sequence takes at most* $\sum_{k=0}^{\lfloor \frac{n}{2} \rfloor -1} n-(2k+1) \approx \frac{n^2}{4}$ *active initiatives to converge. This bound is tight.*

Proof. The reasoning is the same that for Theorem 3. As long as the current configuration is not stable, there exists at least two hot nodes, so after at most $n-1$ active initiatives, the round robin initiative is forced to match a strong pair. The remaining non stable nodes are less than $n-2$, and if the configuration is not stable yet, at least two of them are hot, so after at most $n-3$ active initiatives, a new stable edge is formed...

If we continue this process, we see that the number of active initiatives cannot be more than

$$\sum_{k=0}^{\lfloor \frac{n}{2} \rfloor -1} n-(2k+1) \approx \frac{n^2}{4}.$$

Like for Theorem 3, one can see that this bound is reached for global preferences and complete acceptance graph, if we consider a round robin sequence with the pattern $n, (n-1), \dots, 2, 1$, starting from $C_\emptyset$.

- For an adversarial sequence, the convergence is given by Theorem 4:

Theorem 4 ([37]). *Starting from any configuration, an adversarial sequence takes at most* $2^{n-1}-1$ *active initiatives to converge. Reciprocally, there is a sequence that needs* $\Theta(\lambda^n)$ *active initiatives to converge, with* $\lambda \approx 1.6826$*, thus proving that the complexity of the adversarial daemon stand somewhere between these two bounds.*

Sketch of proof: Like for previous proofs, the $2^{n-1}-1$ bound relies on Lemma 1. The key is to focus on two hot nodes, and to consider the moment where they stabilize. Before that moment, they do not take initiatives, and after that they are inactive. This gives a recursion on the number of nodes that are allowed to take initiatives, and the bound follow.

The $\Theta(\lambda^n)$ bound is obtained for node-based preferences with complete graph, by using a sequence that selects the worst active node, that is the active node with highest label. The study of the configuration evolution for this sequence gives the bound. □

Remark 1. In all previous convergence bounds, global preferences are used for reaching the bounds. From a convergence point of view, it seems that global preferences are the *worst* possible, and it is indeed the case most of the time. More generally, global preferences often display a non-typical behavior compared to other acyclic preferences.

3.1.3 *b*-Matching Generalization

If $b > 1$, Theorem 2 can be adapted and give a $\lfloor \frac{nb}{2} \rfloor$ bound for the number of rounds (the argument is the same: at least one hot edge is stabilized per round). However, this is not a tight bound. For instance, if we take $b = n - 1$ (which is a limit case where there is virtually no quota, because a node can collaborate with all other nodes simultaneously), each node can stabilize all its neighbors after $n - 1$ initiatives, so we have a bound of $n - 1$ rounds, which is far less than $\lfloor \frac{n(n-1)}{2} \rfloor$.

The reason is that if $b > 1$, the system can have warm edges in addition of hot edges. Warm edges stabilize just like hot edges, but they are harder to count, which makes difficult to get a tight bound. However, we still can give a few results for global preferences:

Theorem 5 ([38]). *For node-based preferences, the convergence time is bounded by $\frac{b}{b+1}n$ rounds for the complete acceptance graph, and by n otherwise.*

Note that simulations (cf Section 3.3) show that $\frac{b}{b+1}n$ is a rather accurate bound that reveals well the actual behavior: the few first quotas are the most time-consuming. On the other hand, a complete acceptance graph seems to be the worst case, so the n bound of the general case is clearly an overestimate, not to say the $\lfloor \frac{nb}{2} \rfloor$ bound.

Sketch of proof: For the complete acceptance graph, the proof relies on the fact that the stable configuration is made of cliques of size $b + 1$ ([18]), and that b rounds are enough to stabilize one clique. For an arbitrary acceptance graph, we can only tell that b rounds are enough to stabilize the b best non-stable nodes, which gives the n bound. □

3.2 Expected Convergence Time

Now, we focus on the convergence expected in practice under realistic initiative behaviors, namely the poisson and round-robin sequences. The results presented here are for $b = 1$ and best-mate initiative. We give upper bounds for the expected convergence time of p.b.s., measured in time units.

3.2.1 Generic Upper Bound

The following theorem gives an upper bound for the expected convergence time which is true for any acyclic p.b.s. (with $b = 1$ and best mate initiative).

Theorem 6 ([38]). *The expected convergence time is bounded by $\frac{n}{4}$ for a poisson sequence, and by $\frac{n+1}{6}$ for a round-robin sequence.*

Sketch of proof: Like almost all proofs for estimating the convergence time, Theorem 6 is based on Lemma 1: any non-stable configuration admits at least to hot nodes, and the expected time between two consecutive hot node initiatives is thus less than $\frac{1}{2}$ for a poisson sequence, or $\frac{1}{3}(1 + \frac{1}{n})$ for a round-robin sequence. Multiplying this $\frac{n}{2}$, which is the maximum number of edges to stabilize, gives the result. □

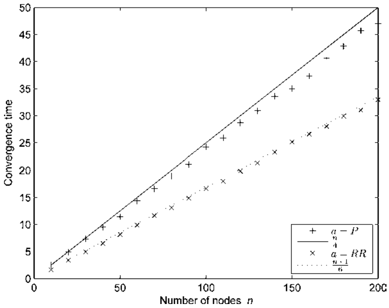


Fig. 4 Expected convergence time for node-based preferences with complete acceptance graph. Each point represents the average over 100 simulations

Remark 2. For node-based preferences with complete acceptance graph, a non-stable configuration admits exactly two hot nodes (the two best non-stable nodes). Thus we can hope that Theorem 6 gives good bounds for such systems. The bounds are in fact very accurate, as shown by Fig. 4: global preferences are the worst possible case, like already observed in Section 3.1.

Remark 3. The tightness of the $\frac{n}{4}$ for a round-robin sequence is consistent with the tightness of the $\lfloor \frac{n}{2} \rfloor$ bound from Theorem 2: the former is a bound for the expected convergence time while the latter is an absolute bound for all possible sequences.

3.2.2 Node-Based Preferences

For node-based preferences and an Erdös-Rényi acceptance graph ($\mathcal{G}(n,p)$, we can provide better bounds:

Theorem 7 ([38]). *For node-based preferences and expected degree $d = p(n-1)$, the expected convergence time is $O(d + \log(n))$ for a poisson sequence and $O(d)$ for a round-robin sequence.*

Sketch of proof: The key of the proof is to *count* the hot nodes in an unstable configuration. Combinatory techniques show that it is of the same order of magnitude of $\frac{n}{d}$. Then we deduce an average time between two hot node selection of the same order of magnitude than $\frac{d}{n}$, which give the $O(d)$ behavior for poisson and round-robin sequences. Lastly, a specific care must be taken regarding the end of convergence, when almost all possible edges are stabilized. In this *endgame* phase, almost all non-stable nodes are hot. During that phase, the non-stable nodes will be stabilized in $0(1)$ for a round-robin sequence (1 time unit stabilizes all existing hot nodes). In contrast, for a poisson sequence, a standard balls ans bins calculus shows that $O(\log(n))$ time units are needed to resolve all the hot nodes. □

3.2.3 Random Acyclic Preferences

For random acyclic preferences, we have a similar result:

Theorem 8 ([38]). *For random acyclic preferences and expected degree $d = p(n-1)$, the expected convergence time is $O(\log(d)+\log(n))$ for a poisson sequence and $O(\log(d))$ for a round-robin sequence.*

Sketch of proof: Using combinatory techniques, we prove that about one half of the non-stable nodes are hot. This result is used to prove that an *endgame* configuration is reached in $O(\log(d))$. Then the endgame add an extra $O(\log(n))$ behavior for a poisson sequence. □

Remark 4. We have $d < n$, so $O(\log(d)+\log(n))$ can be simplified in $O(\log(n))$. The "improper" notation $O(\log(d)+\log(n))$ is only motivated because in practice, the impact of d is greater than the impact of n (cf Section 3.3).

3.2.4 Other Acyclic Preferences

For other preferences, the key to give the expected convergence time would still be to estimate the distribution of hot nodes. It may be difficult for preferences based on real measurements, like the Meridian latencies. Similarly, it may be hard to give the distribution for geometric preferences because of the inter-nodes correlations.

However, we proposed in [38] to use a *peer value* distribution to determine if the convergence should be closer to the node-based case or to the random acyclic case. The idea is to observe if there is correlations between the rankings. The more we see *good nodes*, that is nodes that have good rankings in all the preferences of their neighbors, the more the convergence behavior is likely to be node-based ($O(d)$ or $O(d+\log(n))$). On the other hand, the absence of good nodes tends to indicate a random acyclic convergence ($O(\log(d))$ or $O(\log(d)+\log(n))$).

This rule of thumb is verified for the preferences studied in this chapter. More generally, it may apply for any class of acyclic preferences (for instance the linear combination of two or more simple preferences), and for arbitrary acceptance graphs and quotas. Of course, its validity should always be checked for a specific p.b.s., for instance by means of simulations.

3.3 Simulations

The previous results provide a good theoretical ground for p.b.s., but they are not sufficient to describe all possible p.b.s.: for instance, only the best mate initiative is studied and the bounds for $b > 1$ are rather weak. This is why we propose the use of simulations. We begin by giving the results for the best-mate initiative, then we give a brief overview of other possible initiatives. All technical details on how the simulations were performed can be found in [38].

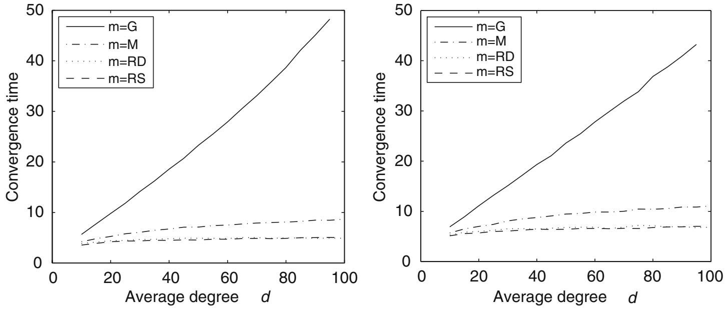
3.3.1 Best-Mate Initiative

In the following, we show how each of the main parameters of the system affects the self-stabilization process under best-mate initiative. Quotas higher than 1 may be used to show how the results seen for $b = 1$ extend to $b \geq 1$.

Neighborhood size

We first consider systems where n is fixed and $d = p(n-1)$ (the average neighborhood size) varies. Results for $n = 100$ and $b = 4$ are shown in Fig. 5 (each point represents the average over 100 samples). Depending on the preferences, two distinct behaviors can be observed:

- For global preferences ($m = G$) the convergence time grows linearly; note that the round-robin sequence is slightly slower than the poisson sequence for d close to n, but faster otherwise;
- other preferences: the convergence time grows logarithmically; round-robin sequences are slightly faster than poisson preferences; the two fastest preferences (random acyclic and geometric) are hard to distinguish.



(a) Round-robin sequence

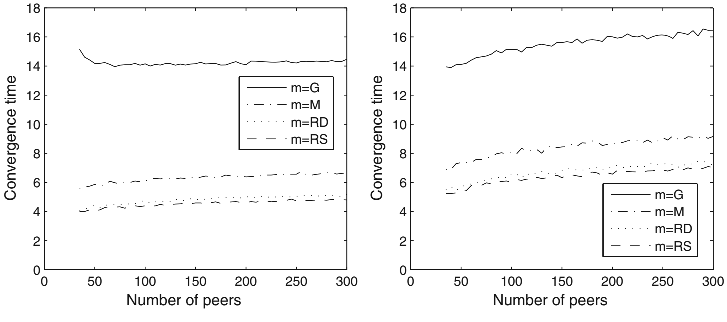
(b) Poisson sequence

Fig. 5 Convergence time as a function of the expected degree d

System size

We now consider systems where the size n of the system varies between 35 and 300 and the expected degree is set to 30. Results are shown in Fig. 6 (each point represents the average over 100 samples; $b = 4$). The trend is that if d is fixed, n seems to have little effect on the convergence time. In details, we can observe that:

- the curves slightly increase with n for poisson sequences ($a = P$);
- for round-robin sequences, they are almost flat, except for global preferences ($m = G$) where the curve decreases at the start (n close to d);
- poisson sequences takes longer than round-robin sequences, except for the case mentioned above (global preferences, n close to d);
- like in Fig. 5, random and geometric preferences converge both faster than Meridian preferences, and the global preferences scenarios are last.



(a) Round-robin sequence (b) Poisson sequence

Fig. 6 Convergence time as a function of the number of nodes n (with fixed degree d)

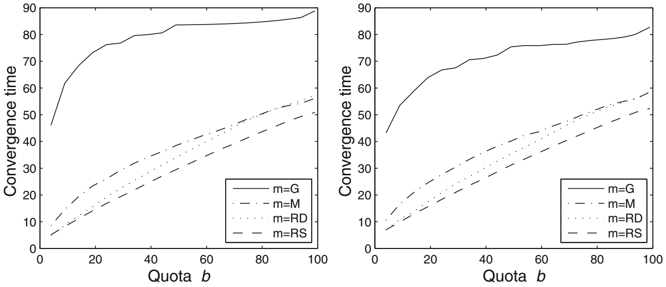
Quotas

Lastly, we study the effect of quotas by considering different values of b for systems with $n = 100$ and complete acceptance graph. Results are shown in Fig. 7 (each point represents the average over 100 samples). Like for the previous simulations, the behavior depends on the preferences used:

- Node-based ($m = G$): the convergence time increases with b; most of the variations happen for small values of b; poisson sequences are slightly faster;
- other preferences: the growth is linear for random acyclic and geometric preferences, and almost linear for Meridian latencies, with a higher slope at the start; round-robin sequences are slightly faster;
- random acyclic and geometric preferences are indistinguishable for small values of b, but geometric becomes closer to Meridian as b increases.

Interpretation

The three experiments above give some hints about the design of systems based on measures like bandwidth, distances or latencies. The main interest of these simulations is that the bounds proved in Section3.2 for $b = 1$ are also validated for $b > 1$.



(a) Round-robin sequence

(b) Poisson sequence

Fig. 7 Convergence time as a function of the quota *b*

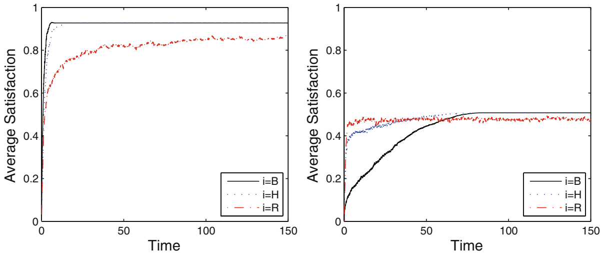
We should retain as a rule of thumb that for node-based systems, the convergence time grows linearly with the average degree of the acceptance graph, so *d* may be the main parameter to consider. The quota *b* is less important, especially past the first values.

On the opposite, in systems based on more symmetric measures (such systems can be classified by analyzing the *good nodes* distribution), the convergence time is roughly proportional to the quota, so *b* should be the main parameter to consider. The logarithmic influence of *d* is less important.

Note that the simulations show that for some specific parameters, a poisson sequence can be faster than a round-robin sequence, despite the endgame advantage for the round-robin sequences. This is caused by a so-called *node-saturation* effect, which happens for node-based preferences when *d* is close to *n* and *b* greater than three. More details about this can be found in [38].

3.3.2 Random and Hybrid Initiatives

Although most of this section is devoted to the best mate initiative, we would like to examine the other initiatives before ending it. Random-mate initiatives need less overhead than best-mate initiatives, because the nodes do not need to keep track of the rankings and mates of their neighbors. The counterpart is that random initiatives are less effective with respect to the convergence time: a random-mate initiative on a hot node may fail to stabilize an edge, unlike a best-mate initiative. However, the time for complete convergence is not the only measure that matters. The time needed for reaching “good enough” configurations is important too. Figure 8 monitors the evolution of a satisfaction measure (cf [38] for a detailed definition of the measure) for different preferences and initiatives. For Meridian preferences (geometric or random acyclic preferences yield similar results), the best-mate initiative clearly outperforms the random-mate initiative with respect to the average satisfaction (Fig. 8a). On the other hand, for node-based preferences (Fig. 8b), the random-mate initiative only takes a few time units to produce a high satisfaction, whereas the best-mate initiative needs more than 50 t.u. to reach the same levels.



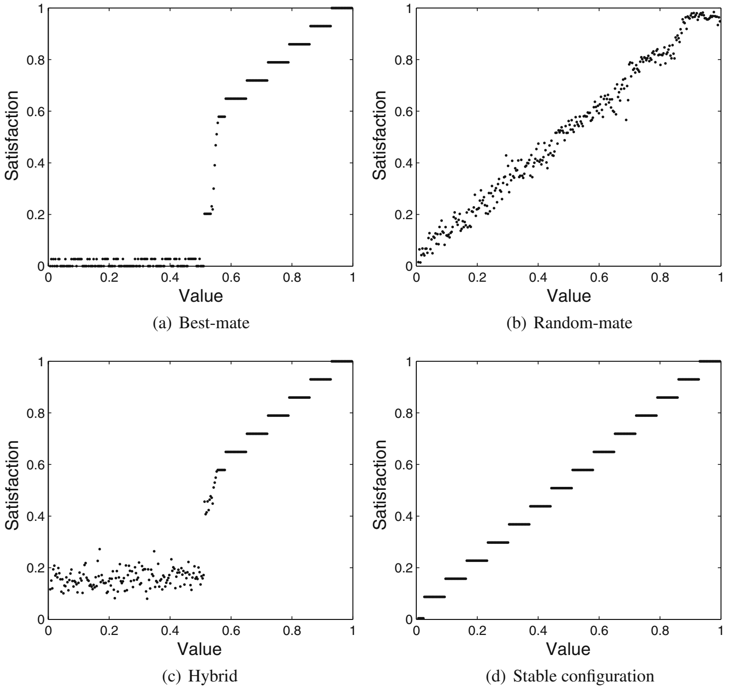
(a) Meridian preferences

(b) Node-based preferences

Fig. 8 Evolution of satisfaction ($n = 200$, $b = 3$, complete acceptance graph and round-robin sequence)

In order to understand this phenomenon, we need a detailed understanding of the convergence process for node-based preferences. Figure 9 shows snapshots of a converging node-based preference system with different initiatives. The y-coordinate indicates the satisfaction of the nodes, which are sorted by increasing value. Figure 9d shows the stable configuration, which should serve as reference. With a best-mate initiative (Fig. 9a), the convergence is abrupt and goes from the best nodes (right) to the worst (left); good nodes stabilize quickly all their connections; bad nodes are mostly single, or have one non-stable connection. The reason is that bad nodes continuously try to connect to the best not stabilized nodes. Those nodes stabilize quickly (they are hot), breaking the bad-nodes connections in the process. Therefore bad nodes cannot keep their mates until full convergence. In contrast, the random-mate initiative induces a sort of uniform convergence for bad and good nodes alike (Fig. 9b): as bad nodes are not continuously trying to connect to good nodes, their connections break less often.

Hybrid initiatives can be used to merge the advantages of best and random initiatives. For instance, we proposed in [38] that a node performs a random initiative while it has not fulfilled its quota, and switches to best-mate when it has established enough collaborations. Figure 9 gives an idea of the resulting convergence process: good nodes still converge quickly, but bad nodes keep a minimal satisfaction because of the random part of the hybrid initiative. Going back to Fig. 8, we see that the performance of hybrid initiative is good for all preferences, with respect to both convergence and transitory satisfaction, so it could be a good solution for systems where the exact nature of the marks is unknown or versatile.



(a) Best-mate

(b) Random-mate

(c) Hybrid

(d) Stable configuration

Fig. 9 **a–c**: Satisfaction of nodes 100 t.u. after the empty configuration $C_\emptyset$. **d**: Stable configuration. Parameters are: $n = 300$, complete acceptance graph, $b = 20$, global preferences, round-robin sequence

4 Stable Configuration Description

Now we want to answer the second question of interest for acyclic preference-based systems: assuming that the self-stabilization ensures that the configuration of a system is stable, or almost stable, *what are the properties of the stable configuration?*

For $b = 1$, we propose a generic mean-field approach. For node-based preferences, we show the existence of a fluid limit for which we give an explicit solution: the stable mate of a peer follows an exponentially decreasing distribution centered around that peer (*stratification* effect [18]). For random acyclic and geometric preferences, the fluid limit indicates a power-law distribution. The validity of the fluid limit can be validated by comparing to an exact solution for node-based preferences, and by simulations otherwise.

Lastly, we extend the results to multiple matchings. Although the corresponding fluid limits have no explicit solution, the same asymptotical behavior than for $b = 1$ is observed (exponential for peer-based preferences, power-law for other classes). As an interesting side-effect, the stable configuration for distance-based classes with enough quotas is a small world.

4.1 Specific Notation

Because we want to focus on a specific configuration (the stable one), we need additional notation to express precisely how mated peers are ranked. If j is acceptable for i, $r_i(j)$ denotes the rank of j in i's list (1 being the best). r_i is called the *acceptable* ranking of i. If i has more than k acceptable neighbors, $r_i^{-1}(k)$ is the k^{th} node in i's acceptable ranking. Similarly, for $j \neq i$, $R_i(j)$ denotes the rank of j in the complete graph (the acceptability condition is omitted.[2]) R_i is called the *complete* ranking of i. For $K < n$, $R_i^{-1}(K)$ is the K^{th} node in i's complete ranking.

All stable mating probabilities that are discussed in this section are designed by D. Subscripts and arguments are used to precise the meaning of D whenever needed. For instance: $D_{R_i}(K)$ denotes the probability that i has a stable mate with complete rank K; $D_{n,d}(i,j)$ is the probability that $i \leftrightarrow j$, knowing there is n nodes and that the expected degree of the acceptance graph is d; for $c \leq b(i,)$, $D_{r_i,c}(k)$ is the probability that the c^{th} stable mate of i has relative rank k...

The complementary cumulative distribution function (CCDF) of D is denoted S, and the scaled version of D and S are denoted $\mathscr{D}$ and $\mathscr{S}$.

4.2 Acyclic Formulas

We first consider the case $b = 1$ (simple matching). The results will be extended to multiple matchings in Section 4.5. We give a generic formula that describes the complete rank of the mate $C(i)$ of a peer i in a stable configuration C.

4.2.1 Generic Formula

Let $D_{R_i}(K)$ be the probability that $R_i(C(i)) = K$ (the probability that the mate of i, if any, has rank K). The CCDF of D is $S_{R_i}(K) := 1 - \sum_{L=1}^{K-1}$, which is the probability that i's mate has a rank greater than K ($R_i(C(i)) \geq K$) or has no mate (short notation: $R_i(C(i)) \nless K$). Following the approach proposed in [18], we first give a generic exact formula that describes D_{R_i}, then we propose a simplified mean-field approximation.

For solving $D_{R_i}(K)$, one can observe that i is mated with its K^{th} peer $j = R_i^{-1}(K)$ iff:

- $\{i, j\}$ is an edge of the acceptance graph; this happens with probability p as G is supposed to be a $\mathscr{G}(n,p)$ graph.
- i is not mated with a node better than j ($R_i(C(i)) \nless K$);
- j is not mated with a node better than i ($R_j(C(j)) \nless R_j(i)$).

[2] We assume for simplicity that m is complete and not limited to the edges of G. For all considered preferences but random acyclic, the completion is straightforward. For random acyclic preferences, we assume that dummy random values are assigned to non-acceptable edges.

This leads to the following exact formula:

$$\begin{aligned} D_{R_i}(K) &= p\mathbb{P}(R_i(C(i)) \nless K)\times \\ &\quad \times\mathbb{P}(R_j(C(j)) \nless R_j(i)|R_i(C(i)) \nless K) \\ &= pS_{R_i}(K)\mathbb{P}(R_j(C(j)) \nless R_j(i)|R_i(C(i)) \nless K) \end{aligned} \tag{1}$$

4.2.2 Mean-Field Approximation

Solving (1) is difficult to handle, mainly because of possible correlations between $R_j(C(j)) \nless R_j(i)$ and $R_i(C(i)) \nless K$. The solution is to adopt a mean field assumption:

Assumption 1 *The events* node i is not with a node better than j *and* node j is not with a node better than i *are independent.*

This assumption has been proposed in [18] to solve (1) in the case of node-based preferences. It is reasonable when n is large and p is small. Then (1) can be approximated by

$$D_{R_i}(K) = pS_{R_i}(K)S_{R_j}(R_j(i)). \tag{2}$$

We propose now solve Equation 2 for specific preferences.

4.3 Node-Based Preferences

We assume here that the preferences comes from marks on nodes. This is equivalent to assume a total order among the nodes. Therefore we do not need to explicit the mark matrix m, and we can use an ordered node labeling instead. We arbitrarily choose $1,\dots,n$ as labels, 1 been the best (i.e., 1 is ranked first for all nodes that accept 1, and so on...).

Because the nodes' label express their complete ranks, we can directly consider $D(i,j)$, the probability that node i is mated with node j. Node j has rank j for i if $j<i$, and $j-1$ if $j>i$, because a rank does not rank itself. This gives the relation between D and D_R:

$$D(i,j) = \begin{cases} D_{R_i}(j) \text{ if } j<i, \\ 0 \text{ if } j=i \text{ (mating is not reflexive)}, \\ D_{R_i}(j-1) \text{ if } j>i. \end{cases} \tag{3}$$

Using the CCDF $S(i,j) := 1-\sum_{k=1}^{j-1}D(i,k)$, we get the node-based version of Equation (2):

$$D(i,j) = \begin{cases} 0 \text{ if } i=j, \\ pS(i,j)S(j,i) \text{ otherwise.} \end{cases} \tag{4}$$

This equation was originally proposed in [18], where it was also shown that it gives a very good approximation of the empirical distribution. It can be numerically solved by using a double iteration.

4.3.1 Fluid Limit

For node-based preferences, we propose to prove that, under a constant degree scaling, D admits a fluid limit. This limit gives a complete description of D that can be applied to all values of n and p, while Equation (4) needs to be solved for each set of parameters.

4.3.2 Constant Degree Scaling

In order to compare the distributions for arbitrary values of n, we need a scaled version of D, where a peer i is represented by a scaled ranking $0 \leq \alpha < 1$. In details, we associate to each i the number $\alpha(i) = \frac{i-1}{n}$, and to each real number α the node $i(\alpha) = \lfloor n\alpha \rfloor + 1$. The scaled version of D, denoted $\mathscr{D}$, is then defined by

$$\mathscr{D}_n(\alpha,\beta) = nD(\lfloor n\alpha \rfloor + 1, \lfloor n\beta \rfloor + 1).$$

$\mathscr{D}_n$ is a piecewise constant function. Its set of function values is the set of the $(nD(i,j))$ values. The factor n in its definition allows to express $D(i,j)$ as an integral of $\mathscr{D}$:

$$D(i,j) = \int_{\frac{j-1}{n}}^{\frac{j}{n}} \mathscr{D}_n(\frac{i-1}{n},x)\,\mathrm{d}x = \int_{\frac{i-1}{n}}^{\frac{i}{n}} \mathscr{D}_n(x,\frac{j-1}{n})\,\mathrm{d}x$$

The scaling of the CCDF is defined by

$$\mathscr{S}_n(\alpha,\beta) = 1 - \int_0^{\beta} \mathscr{D}_n(\alpha,x)\,\mathrm{d}x, \tag{5}$$

and the relation between S and $\mathscr{S}$ is

$$S(i,j) = \mathscr{S}(\frac{i-1}{n},\frac{j-1}{n}). \tag{6}$$

4.3.3 Convergence Theorem

We now want to show the existence of a continuous limit for $\mathscr{D}$. The problem is the existence of an intrinsic discontinuity for $\alpha \approx \beta$, because $D(i,i) = 0$. However, this discontinuity is just a reminder of the fact that a node cannot mate with itself, so we propose to make $\mathscr{D}$ more "continuous" by introducing

$$\tilde{\mathscr{D}}(\alpha,\beta) = \begin{cases} \mathscr{D}(\alpha,\beta) \text{ if } \lfloor n\alpha \rfloor \neq \lfloor n\beta \rfloor, \\ np(S(\lfloor n\alpha \rfloor + 1, \lfloor n\alpha \rfloor + 1))^2 \text{ otherwise.} \end{cases}$$

The fluid limit of $\tilde{\mathscr{D}}$ is then given by the following theorem:

Theorem 9. *Let $d > 0$ be a constant. If $n \to \infty$ with $p = \frac{d}{n}$, the function $\tilde{\mathscr{D}}_{n,d}$ uniformly converge towards*

$$\mathscr{D}_\infty(\alpha,\beta) = \frac{de^{d(|\beta-\alpha|)}}{(1 - e^{-d\min(\alpha,\beta)} + e^{d|\beta-\alpha|})^2}. \tag{7}$$

This result indicates that asymptotically, the average degree in the acceptance graph completely defines the mating distribution. The consequence is that we can explicitly describe the so-called *stratification* effect [18]: the mating distribution is exponentially decreasing with $|\beta - \alpha|$, with intensity d. In other words, a peer with scaled rank α tends to mate with a mate of same scaled rank, with a standard deviation of the same order than $\frac{1}{d}$.

Sketch of proof: The complete proof of Theorem 9 is available in [39]. It uses four distinct steps:

- prove that the $\tilde{\mathscr{D}}_N$ functions are uniformly Cauchy on $[0,1]^2$;
- use the Cauchy convergence to show that $\mathscr{S}_N$ and $\tilde{\mathscr{D}}_N$ have limits $\mathscr{S}_\infty$ and $\mathscr{D}_\infty$;
- give a PDE verified by $\mathscr{S}_\infty$;
- solve the PDE, and use the solution to get $\mathscr{D}_\infty$.

□

Note that the existence of a fluid limit was proposed as a conjecture in [18], and proved for $\alpha = 0$ (but the expression of the fluid limit in the general case was not provided).

Theorem 9 gives two immediate corollaries:

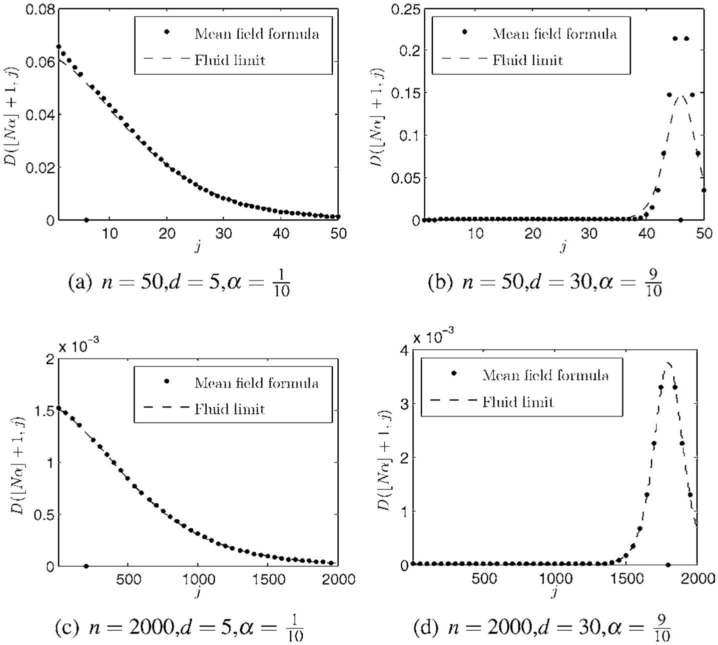
- using the CCDF of $\mathscr{D}_\infty$, the probability that a node of scaled rank α has no mate is $\frac{1}{1+e^{-d\alpha}(e^{-d}-1)}$;
- for $i \neq j$ (discrete case), a good approximation for $D(i,j)$ is

$$D(i,j) \approx \frac{pe^{p(|j-i|)}}{(1 - e^{-p\min(i,j)} + e^{p|j-i|})^2}. \tag{8}$$

4.3.4 Validation

We compared the fluid limit approximation (8) to the mean-field equation (4), which is known to be accurate [18].

Representative results are shown in Fig. 10. n was set to 50 or 2000, and d to 5 or 30. Because D is 2-dimensional, we arbitrarily set the scaled rank α to 0.1 or 0.9 (but the convergence validation holds for any α).



(a) $n = 50, d = 5, \alpha = \frac{1}{10}$

(b) $n = 50, d = 30, \alpha = \frac{9}{10}$

(c) $n = 2000, d = 5, \alpha = \frac{1}{10}$

(d) $n = 2000, d = 30, \alpha = \frac{9}{10}$

Fig. 10 Validation of the fluid limit for node-based preferences

We observe a gap for $j = \lfloor n\alpha \rfloor + 1$, because the mean field formula sets D to 0 whereas the fluid limit uses a continuous extension.

Besides this gap, $n = 50$ (Fig. 10a, b) shows some difference between the mean field and the fluid limit. The error is especially noticeable for $d = 30$ Fig. (10b). However, for $n = 2000$ (Fig. 10c, d), there is practically no error.

These results are consistent with the complete proof of Theorem 9 [39], which shows that the convergence is $O(\frac{d^2}{n}e^{8d})$.

Remark 5. For the record, if $b = 1$, there exists an exact recursive formula for the node-based stable configuration, which gives the same fluid limit than the mean field approach. A complete description of the formula, including its construction, its PDE counterpart and resolution, can be found in [39]. However, the issue with that exact formula is that it cannot be generalized for other preferences or for $b > 1$. This is why we prefer not to present that formula here and focus on the mean field formulas.

4.4 Acyclic and Distance-Based Preferences

We now consider geometric and random acyclic preferences. Following the approach used for node-based preferences, we first focus on the complete rank distribution. Under the mean field assumption, we propose a recursive formula for D, then we solve the corresponding fluid limit. The results are extended to the distance and acceptable ranking distributions.

4.4.1 Complete Rank Distribution

Assumption 1 is not enough for solving (2) in the case of geometric or random acyclic preferences. Therefore, we propose this additional assumption:

Assumption 2 *For geometric and random acyclic preferences, the following approximations hold:*

- *$D_{R_i}(K)$ is independent of i (and therefore denoted $D_R(K)$);*
- *the complete ranking is symmetric: $R_i(j) = R_j(i)$.*

The first approximation just states that in average, all nodes have the same mate distribution, while the second one tells that $R_i(j)$ is a good approximation of $R_j(i)$. These approximations were motivated by the uniform distributions used for shaping the preferences. In particular, they do not apply for node-based preferences, where the mate distribution is strongly affected by a node's mark. Under these assumptions, we get

$$D_R(K) = pS_R^2(K), \text{ with } S_R(K) = 1 - \sum_{L=1}^{K-1} D_R(L). \tag{9}$$

This equation gives an immediate recursion for S_R:

$$S_R(K) = \begin{cases} 1 \text{ if } K = 1, \\ S_R(K-1) - pS_R^2(K-1) \text{ otherwise.} \end{cases} \tag{10}$$

D_R is then directly given by $D_R(K) = S_R(K) - S_R(K+1)$.

Fluid limit

We now give the fluid limit of D_R. The scaled version of D_R is defined like for node-based preferences, except that there now only one parameter. For $0 \leq \alpha < 1$, we define $\mathscr{D}_R(\alpha) := (n-1)D_R(\lfloor(n-1)\alpha\rfloor\rfloor + 1)$. The scaling factor is now $n-1$ because it is the upper bound for K (while n was the upper bound for i, j in Section 4.3). D_R can be expressed as an integral of $\mathscr{D}_R$:

$$D_R(K) = \int_{\frac{K-1}{n-1}}^{\frac{K}{n-1}} \mathscr{D}_R(x)\,\mathrm{d}x.$$

The scaled CCDF, $\mathscr{S}_R$, is then naturally defined as:

$$\mathscr{S}_R(\alpha) = 1 - \int_0^{\alpha} \mathscr{D}_R(x)\,\mathrm{d}x.$$

Theorem 10. *We assume that $d = p(n-1)$ is a positive constant. As $n \to \infty$, $\mathscr{S}_R$ uniformly converges towards*

$$\mathscr{S}_\infty(\alpha) = \frac{1}{d\alpha + 1}. \tag{11}$$

In particular, the probability that a node has no mate in the stable configuration is $\mathscr{S}_R(1) = \frac{1}{d+1}$, and a good approximation for $S_R(K)$ is

$$S_R(K) = \frac{1}{p(K-1)+1}. \tag{12}$$

Sketch of proof: The proof is a simpler version of the proof of Theorem 9, which can be found in [39]. First we prove that the $\mathscr{D}_R$ functions are uniformly Cauchy, which is simpler here because there is only one variable and no need for a continuous extension. This proves the uniform convergence towards $\mathscr{S}_\infty$. Then we deduce from (10) a differential equation verified by $\mathscr{S}_\infty$:

$$-\dot{\mathscr{S}}_\infty(\alpha) = d\mathscr{S}_R^2(\alpha), \tag{13}$$

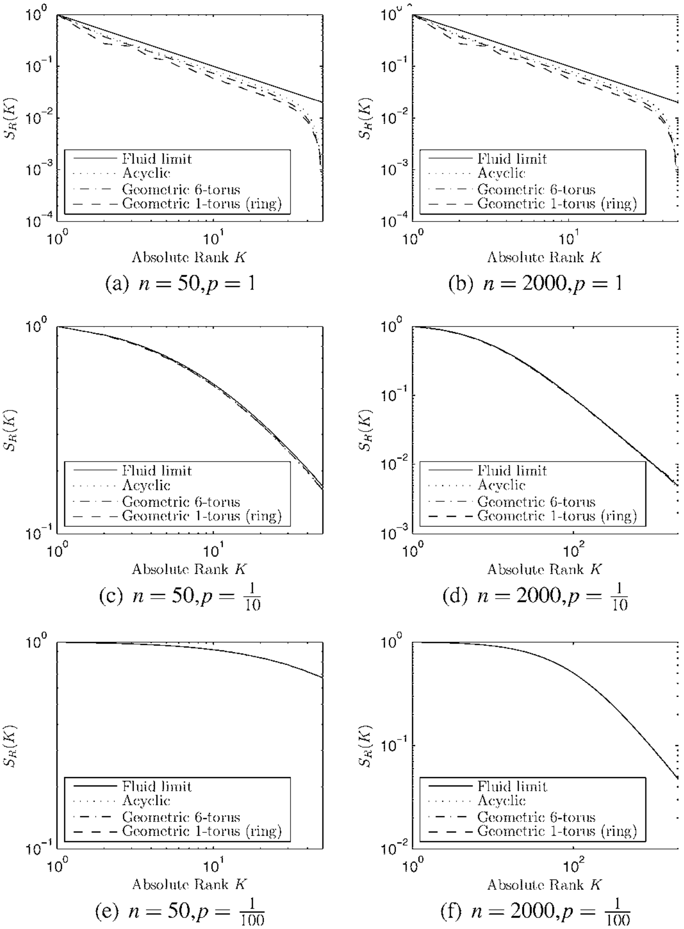
with the boundary condition $\mathscr{S}_R(0) = 1$. The resolution of (13) gives (11), which completes the proof. □

Validation

In order to validate the mean field formula (9), we considered random acyclic instances, and geometric preferences in a 1-dimensional torus and in a 6-dimensional torus. Figure 11 shows the accuracy of the formula for representative parameters. n was set to 50 or 2000. We used 3 values of p: 1, $\frac{1}{10}$ and $\frac{1}{100}$. For each set of parameters, the empirical distribution was calculated over 100 instances.

For $p = 1$ (Fig. 11a, b), the mean-field assumptions hardly hold. As a consequence, the curves depend of the type of preferences, and the fluid limit is not accurate. This is especially visible if K is close to the boundaries (that is 1 or n). In particular, the non-mate probability is clearly over-estimated. However, the fluid limit manages to give the $O(\frac{1}{K})$ behavior that is common to all considered preferences. From that point of view, the fluid limit performs better than the recursive Equation (10), which gives $S_R(K) = \delta_K^1$ for $p = 1$.

For $p = \frac{1}{10}$ (Fig. 11c, d), the curves are nearly indistinguishable. We verify that all types of preferences (acyclic or geometric) tend to have the same behavior and that Theorem 10 gives precise approximations.



(a) $n = 50, p = 1$

(b) $n = 2000, p = 1$

(c) $n = 50, p = \frac{1}{10}$

(d) $n = 2000, p = \frac{1}{10}$

(e) $n = 50, p = \frac{1}{100}$

(f) $n = 2000, p = \frac{1}{100}$

Fig. 11 Empirical validation of the fluid limit for random acyclic and geometric preferences

For $p = \frac{1}{100}$ (Fig. 11e, f), the curves are indistinguishable.

We conclude that fluid-limit based on the mean-field formula is very effective for computing the complete ranking distribution, even if n is not very large and p is not very small.

4.4.2 Distance Distribution

For geometric preferences, the actual distance between a node and its mate may be a more valuable performance indicator than the ranking. We call $S_X(x)$ the probability that the distance between a node i and its mate $C(i)$ is not less than x (in other words, the distance is greater than x or i is unmated). Under the fluid limit, we get a good estimate of S_X:

$$S_X(x) = \frac{1}{dB_t(x)+1}, \tag{14}$$

where B_t is the size of a ball of radius x in the t-torus.

Proof. In the fluid limit, a ball of radius x contains $nB_t(x)$ nodes, because it occupies a ratio $B_t(x)$ of the torus. Therefore the farest node in a x-ball centered at a node i should have a complete rank $nB_t(x)$ for i, while being at a distance x from i. We deduce that $S_X(x) = S_R(nB_t(x))$. Equation (12) concludes.

The value of $B_t(x)$ depends on the dimension t and on the norm used. If we consider the maximum norm, then $B_t(x) = \min((2x)^t, 1)$. For other norms, the formula may be more complicated because the ball may partially overlap itself in the torus. Should we have chosen $\mathbb{R}^t$ (with uniform point distribution) instead of the t-torus, $B_t(x)$ would just have been the size of a ball of radius x.

Figure 12 shows S_X for $t = 1$ and $t = 3$, with the taxicab norm. With this norm, we have $B_1(x) = \min(2x, 1)$ and

$$B_3(x) = \begin{cases} \frac{4}{3}x^3 \text{ if } 0 \le x \le \frac{1}{2}, \\ \frac{4}{3}x^3 - 4(x - \frac{1}{2})^3) \text{ if } \frac{1}{2} \le x \le 1, \\ 1 - \frac{4}{3}(\frac{3}{2} - x)^3 \text{ if } 1 \le x \le \frac{3}{2}, \\ 1 \text{ if } x \ge \frac{3}{2}. \end{cases}$$

We used $n = 2000$ and $p = \frac{1}{100}$, and the fluid limit and empirical distribution of S_X were indistinguishable.

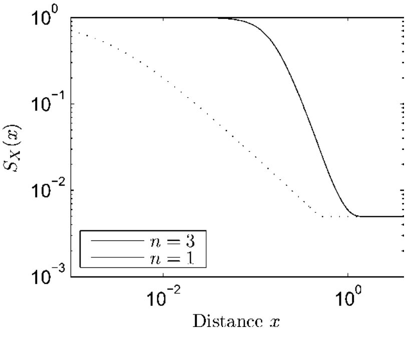


Fig. 12 Distance distribution ($n = 2000$, $p = \frac{1}{100}$)

4.4.3 Acceptable Rank Distribution

Now we want to investigate the probability that the mate of a node has an acceptable rank k. We call $D_r(k)$ this probability. Like for the other distributions, we introduce the CCDF $S_r(k) := 1 - \sum_{l=1}^{k-1} D_r(k)$.

Following the complete ranking method, we consider the conditions for a node i to be mated with its k^{th} best neighbor $j = r_i^{-1}(k)$:

- i must have k neighbors or more,
- it must not be mated with someone better than j,
- j must not be mated with someone better than i.

With the acceptance ranking, there is intrinsic correlations between these events that complicates things. Despite of that, assuming that these events are independent allows us to give a first, non-accurate, recursive formula:

$$D_r(k) = S_r(k)\frac{1 - I_{1-p}(n-k+1,k)}{k+1}, \tag{15}$$

where I_x is the regularized incomplete beta function.

Proof. i has k neighbors or more with probability $1 - I_{1-p}(n-k+1,k)$. The probability that i is not with better than j is $S_r(k)$. For the reciprocal, we can use $K = \frac{k}{p}$ as a (very rough) approximation of the complete rank; then Equation (12) gives the probability $\frac{1}{k+1}$. Formula (15) follows.

The results are shown in Fig. 13. One can observe that Equation (15) is not accurate for $D_r(1)$, which provokes a gap between the empirical distribution and the formula.

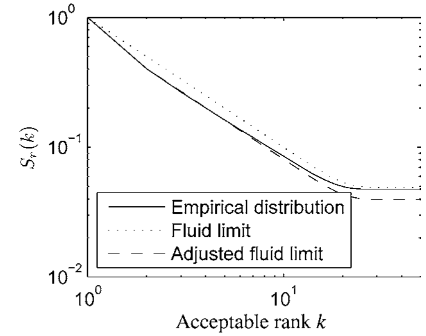


Fig. 13 Acceptable ranking CCDF ($n = 2000, p = \frac{1}{100}$)

In an attempt to adjust the formula, we propose a more accurate estimation of $D_r(1)$: under the normalized fluid limit, the scaled rank of the first neighbor j of a given peer i follows the distribution $de^{-d\alpha}$. j and i are mate if j is mated with someone with a scaled rank greater or equal to α, which happens with probability $\frac{1}{d\alpha+1}$. Thus we have

$$\begin{aligned} D_r(1) &= \int_0^\infty \frac{d\mathrm{e}^{-d\alpha}}{d\alpha+1}\mathrm{d}\alpha \\ &= \int_1^\infty \frac{\mathrm{e}^{-t+1}}{t}\mathrm{d}t \\ &= \mathrm{e}E_1(1) \approx 0.596 \end{aligned} \tag{16}$$

The accuracy of $D_r(1) = \mathrm{e}E_1(1)$ (E_1 denotes the exponential integral) is verified in Fig. 14. If we use this value for adjusting the fluid limit, we get a better estimation of S_r for small values of k (cf Fig. 13). However, this adjustment introduces a

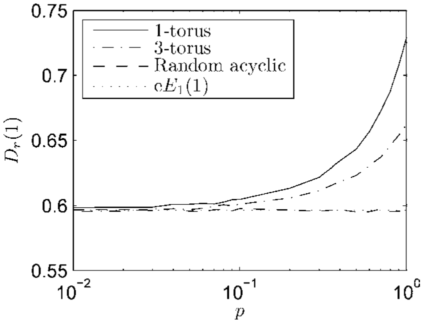


Fig. 14 $D_r(1)$ as a function of p ($n = 2000$)

gap for larger values of k. In a further version, we will aim at unifying these two estimates, which will require a better understanding of the correlations that occur when considering the acceptable rank.

4.5 b-Matching Generalization

We now extend our results to the case of multiple matchings. For simplicity, we still assume that the quota vector b is a scalar, i.e., that all nodes share the same number of authorized collaborations. For distance and acyclic preferences, we focus on the complete rank, although distance and acceptable ranking could be derived using the same techniques than for $b = 1$.

4.5.1 Mean Field Formulas

A peer can now have up to b mates. For $1 \leq c \leq b$, D_c denotes the distribution of the complete ranking of the c^{th} best mate, and S_c denotes the corresponding CCDF. Like we did for $b = 1$, we can give the conditions for a node $j = R_i^{-1}(K)$ to be the c^{th} mate of a node i:

- $\{i, j\}$ is an acceptable edge,
- the $(c-1)$th mate of i (if $c > 1$) is better than j, but the cth (if any) is not,
- the bth mate of j (if any) is not better than i.

By extending Assumption 1, we obtain a generic mean field formula for multiple matchings:

$$D_{R_i,c}(K) = \begin{cases} pS_{R_i,1}(K)S_{R_j,b}(R_j(i)) \text{ if } c = 1, \text{ otherwise} \\ p(S_{R_i,c}(K) - S_{R_i,c-1}(K))S_{R_j,b}(R_j(i)). \end{cases} \tag{17}$$

Like for the simple matching case, this formula can be adapted to specific preferences.

We first consider node-based preferences. $D_c(i,j)$ being the probability that the cth mate of i is j, we have the following system, which can be solved by a double iteration on i and j (cf [18])

$$D_c(i,j) = \begin{cases} 0 \text{ if } i=j, \\ pS_1(i,j)S_b(j,i) \text{ if } i \neq j,\, c=1, \\ p(S_c(i,j)-S_{c-1}(i,j))S_b(j,i) \text{ if } i \neq j,\, c>1. \end{cases} \tag{18}$$

Then, for acyclic and distance-based preferences, we also extend Assumption 2 (homogeneity of the distributions and symmetry of the complete ranking). This gives the following system:

$$D_{R,c}(K) = \begin{cases} pS_{R,1}(K)S_{R,b}(K) \text{ if } c=1, \\ p(S_{R,c}(K)-S_{R,c-1}(K))S_{R,b}(K) \text{ if } c>1. \end{cases} \tag{19}$$

Using $S_{R,c}(1)=1$ and $D_{R_c}(K)=S_{R,c}(K)-S_{R,c}(K+1)$, Equation (19) immediately gives an iterative computation of $S_{R,c}$.

Figure 15 shows $S_c(i,j)$ (node-based) and $S_{R,c}$ (acyclic/geometric) as obtained by (18) and (19). The parameters are $b \in \{2,3,4\}$, $n=2000$, $p=\frac{1}{100}$, and $i=1001$ (for $S_c(i,j)$). We verified for each set of parameters that the curves coincide with the empirical distribution. S and S_R (CCDF for $b=1$) are also plotted for serving as landmarks. We see that the curves have a behavior that is similar than for the simple matching case: for node-based preferences, it seems that the distributions $D_c(i,.)$ are still exponentially decreasing, even if the multiplicity of matchings creates offsets between the distribution peaks and i. For acyclic and geometric preferences, we still observe a kind of power law behavior.

4.5.2 Fluid Limits

Fluid limits also exist for $b>1$. We will not present the proofs here, because they are essentially the same than for the simple matching fluid limits, only more complex to write because of the multiple distribution involved. Therefore we just give the equations verified by those limits.

For node-based preferences, the scaled limit $\mathscr{S}$ of the CCDF verifies:

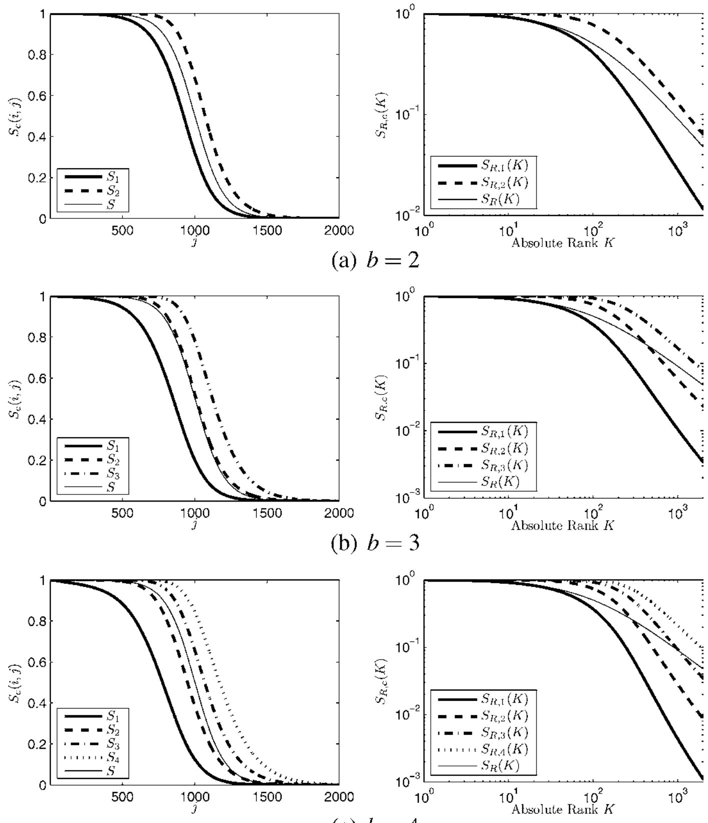
$$\partial_y \mathscr{S}_c(\alpha,\beta) = \begin{cases} -d\mathscr{S}_1(\alpha,\beta)\mathscr{S}_b(\beta,\alpha) \text{ for } c=1, \text{ otherwise} \\ -d\left(\mathscr{S}_c(\alpha,\beta)-\mathscr{S}_{c-1}(\alpha,\beta)\right)\mathscr{S}_b(\beta,\alpha), \end{cases} \tag{20}$$

with border conditions $\mathscr{S}_c(\alpha,0)=1$.

Similarly, for acyclic and distance-based preferences, the scaled limit $\mathscr{S}_R$ of the CCDF verifies

$$\dot{\mathscr{S}}_{R,c} = \begin{cases} -d\mathscr{S}_{R,1}\mathscr{S}_{R,b} \text{ if } c=1, \\ -d(\mathscr{S}_{R,c}-\mathscr{S}_{R,c-1})\mathscr{S}_{R,b} \text{ if } c>1, \end{cases} \tag{21}$$

with the boundary condition $\mathscr{S}_{R,c}(0)=1$.



(a) $b = 2$

(b) $b = 3$

(c) $b = 4$

Fig. 15 Complete rank CCDFs for $b > 1$. Node-based (resp. acyclic/geometric) distributions are on the left (resp. right) side. $n = 2000$, $p = \frac{1}{100}$, and $i = 1001$ (for $S_c(i, j)$)

There is no simple explicit solution for Equations (20) and (21). However, (18) and (19) can still be used as a difference equation to approximate a numerical solution. The reason for which we give these limits is that we think that they can give us valuable information about the asymptotical behavior of the distribution (exponentially decreasing or power law), even if this work is still to be done.

4.6 Some Basic Applications

We now propose to give two simple examples of the interest of having a good description of the stable configuration: the stratification effect for node-based preferences, and the small-world effect for geometric preferences.

4.6.1 Stratification Trade-off

As we have seen, for node-based preferences, the mates of a given peer i have, in average, the same rank than i. This is the stratification effect ([18]), which guarantees some fairness in the stable configuration: the expected gain of a node tends to be the value offered by this node, measured in terms of ranking. However, we also observed that the exponential decreases of the $\mathscr{D}_c(i,.)$ functions provokes a standard deviation of the same order that $\frac{1}{d}$, where d is the average degree in the acceptance graph. This gives the following stratification trade-off:

- if d is too small, the standard deviation is high. In particular, if the mark matrix is non uniformly distributed, there can be a big difference between the expected gain and gift, measured with the marks. This issue has been highlighted in [18] for explaining a possible workaround of BitTorrent's Tit-for-Tat policy;
- on the other hand, a high d will enforce the fairness. However, increasing the size of the acceptance graph degree has a cost for the nodes: convergence time, memory usage, overlay management,... Also, the absence of long-range mates makes the diameter of the stable configuration high, which can be problematic if messages are to be spread using stable edges.

Note that there are similar trade-offs between cost and quality for b, which is the maximal degree in the stable configuration.

This suggests that most node-based preference systems (this includes the systems based on the sharing of an access bandwidth, a storage or CPU capacity, an expected uptime,...) should admit an optimal pair (d,b) with respect to the stable collaborations properties, whose values depend on the weight put on performance measures like diameter, fairness homogeneity, convergence time or overhead cost.

4.6.2 Small-World Effect and Preference Dimension

A small world is a sparse graph with a low average shortest path length (ASPL) and a high clustering coefficient. In details:

- *sparse graph* means that the average degree is $O(\log(n))$ or even $O(1)$,
- *low ASPL* means $O(\log(n))$,
- *high clustering coefficient* means that two nodes sharing an edge are likely to have a common neighbor. The clustering coefficient is a probability, that must be compared to a reference value, for instance the clustering coefficient of a random graph with same number of nodes and edges.

In [30], Kleinberg proved that a n-dimensional grid can be turned into a small world by adding long-range edges that follow a $\Omega(\frac{1}{x^n})$ distribution.

For multiple matchings, the stable configuration in geometric preferences is likely to have a high clustering coefficient, because most of the stable edges link close nodes. Moreover, the power-law rank distribution tells that long-range edges exist. So the stable configuration is likely to be eligible as a small-world.

Type of preferences	ASPL	Clustering Coefficient
1-torus	7.4	0.055
2-torus	6.7	0.043
Meridian	6.1	0.031
3-torus	5.9	0.033
4-torus	5.1	0.027
Random Acyclic	5	0.0043

Table 1 ASPL and clustering ($n = 2000$, $p = \frac{1}{10}$, $b = 10$)

In Table 1, we give the ASPL and clustering coefficient for some preferences, using the parameters $N = 2000, p = \frac{1}{10}, b = 10$. The reference clustering is here $\frac{b}{n-1} \approx 0.005$. We verify that for the t-tori, the stable configurations are small-worlds. This seems to be a specific property of geometric preferences: like previously observed in [16], the stable configurations of random acyclic preferences are not small-worlds, because of their clustering coefficient (they behave like an incomplete b-regular graph). Similarly, node-based preferences have a high clustering but the corresponding ASPL is too high ($\Theta(n)$).

The small-world property of a distance-based stable configuration only results from how the nodes rank each other: the actual distances are only used for sorting the nodes, but the values are never involved in the configuration construction. Nevertheless, they reveal valuable insight about the topology of the underlying set of distances.

For instance, we calculated the ASPL and clustering obtained by using the Meridian Project's real-world latencies, which are known to produce small-worlds configuration [16]. One can observe that the results are very close to the one obtained with 3-torus, suggesting that somehow, 3 may be seen as sort of dimension for the latency space. Considering the recent eager for estimating the Internet dimension (see for instance [2]), this unexpected result is appealing.

5 Conclusion

We gave a large overview of the acyclic preference-based systems, with an emphasis on relevant preferences such as node-based or geometric preferences. Regarding the self-stabilization property of such systems, we gave techniques for estimating the convergence time, but we also proposed the use of simulation in order to clearly separate the importance of each parameter. Then we gave a statistical description of the stable configurations obtained from node-based, geometric and random acyclic preferences. Starting from a generic formula for the rank distribution, we introduced mean-field and fluid limit techniques in order to give explicit formulas. We proved and validated by means of simulations that the distributions are exponentially decreasing for node-based preferences (stratification effect), and power-law otherwise. As an interesting consequence, for distance-based preferences, the stable configurations behave similarly to Kleinberg's grids, and are small-world graphs.

All these results should provide a general guideline for designers of new systems that use preference-based interactions. Of course, not all questions about the acyclic preference-based systems (or more generally preference-based systems) are covered in this chapter. However, this is a handbook, and we hope that the methods deployed here can give leads to anyone wishing to further pursue this area. For instance, how to use hot and warm nodes for evaluating the convergence speed; how to employ mean field and fluid limit approaches to describe the stable configuration; how to balance between theory and simulation in order to have both strong theoretical and practical results.

References

1. Abraham, D.J., Levavi, A., Manlove, D., O'Malley, G.: The stable roommates problem with globally-ranked pairs. In: WINE, pp. 431–444 (2007)
2. Abrahao, B., Kleinberg, R.: On the internet delay space dimensionality. In: Proceedings of the 2008 Internet Measurement (2008)
3. Babelgum: Babelgum. URL `http://www.babelgum.com/`
4. Bonald, T., Massoulié, L., Mathieu, F., Perino, D., Twigg, A.: Epidemic live streaming: optimal performance trade-offs. In: Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS) (2008)
5. Castro, M., Druschel, P., Kermarrec, A.M., Nandi, A., Rowstron, A., Singh, A.: Splitstream: high-bandwidth multicast in cooperative environments. In: SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, pp. 298–313. ACM, New York, NY, USA (2003). DOI http://doi.acm.org/10.1145/945445.945474
6. Cechlárová, K., Fleiner, T.: On a generalization of the stable roommates problem. ACM Transactions on Algorithms **1**(1), 143–156 (2005)
7. Chowla, S.: The asymptotic behavior of solutions of difference equations. In: Proceedings of the International Congress of Mathematicians, Vol. I, 377, American Mathmatical Society. (1950)
8. Cohen, B.: Incentives build robustness in bittorrent. In: P2PECON (2003)
9. Diamantoudi, E., Miyagawa, E., Xue, L.: Random paths to stability in the roommate problem. Games and Economic Behavior **48**(1), 18–28 (2004)

10. Http://www.edonkey2000.com/index.html
11. Http://www.emule-project.net/
12. Fleiner, T.: The stable b-matching polytope. Mathematical Social Science **46(2)**, 149–158 (2003)
13. Frey, D., Royan, J., Piegay, R., Kermarrec, A., Anceaume, E., Fessant, F.L.: Solipsis: A decentralized architecture for virtual environments. In: International Workshop on Massively Multiuser Virtual Environments. IEEE (2008)
14. Gai, A., Viennot, L.: Incentive, resilience and load balancing in multicasting through clustered de bruijn overlay network (prefixstream). In: Proceedings of the 14th IEEE International Conference on Networks (ICON), vol. 2, pp. 1–6. IEEE Computer Society (2006)
15. Gai, A.T., Lebedev, D., Mathieu, F., de Montgolfier, F., Reynier, J., Viennot, L.: Acyclic preference systems in p2p networks. In: Proceedings of the 13th International Parallel Processing Conference (Euro-Par), pp. 825–834 (2007)
16. Gai, A.T., Lebedev, D., Mathieu, F., de Montgolfier, F., Reynier, J., Viennot, L.: Acyclic preference systems in p2p networks. In: Euro-Par, pp. 825–834 (2007)
17. Gai, A.T., Mathieu, F., de Montgolfier, F., Reynier, J.: Stratification in p2p networks: Application to bittorrent. In: Proceedings of the 27th IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 30–39 (2007)
18. Gai, A.T., Mathieu, F., de Montgolfier, F., Reynier, J.: Stratification in p2p networks: Application to bittorrent. In: ICDCS, p. 30 (2007)
19. Gai, A.T., Viennot, L.: Broose: A practical distributed hashtable based on the de-bruijn topology. In: P2P '04: Proceedings of the Fourth International Conference on Peer-to-Peer Computing, pp. 167–164. IEEE Computer Society, Washington, DC, USA (2004). DOI http://dx.doi.org/10.1109/P2P.2004.10
20. Gale, D., Shapley, L.: College admissions and the stability of marriage. American Mathematical Monthly **69**, 9–15 (1962)
21. Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K.W.: Insights into pplive: A measurement study of a large-scale p2p iptv system. In: In Proc. of IPTV Workshop, International World Wide Web Conference (2006)
22. Irving, R.: An efficient algorithm for the stable roommates problem. Journal of Algorithms **6**, 577–595 (1985)
23. Irving, R., Leather, P., Gusfield, D.: An efficient algorithm for the "optimal" stable marriage. J. ACM **34**(3), 532–543 (1987)
24. Irving, R.W., Manlove, D., Scott, S.: The hospitals/residents problem with ties. In: SWAT '00: Proceedings of the 7th Scandinavian Workshop on Algorithm Theory, pp. 259–271. Springer-Verlag, London, UK (2000)
25. Irving, R.W., Manlove, D.F.: The stable roommates problem with ties. Journal Algorithms **43**(1), 85–105 (2002). DOI http://dx.doi.org/10.1006/jagm.2002.1219
26. Iwama, K., Miyazaki, S., Manlove, D., Morita, Y.: Stable marriage with incomplete lists and ties. In: ICALP, pp. 443–452 (1999)
27. Joost: Joost. URL `http://www.joost.com/`
28. Kawahara, Y., Aoyama, T., Morikawa, H.: A peer-to-peer message exchange scheme for large-scale networked virtual environments. Telecommunication Systems **25**(3) (2004)
29. Keller, J., Simon, G.: Solipsis: a massively multi-participant virtual world. In: Intern. Conf. on Parallel and Distributed Techniques and Applications (2003)
30. Kleinberg, J.: The small-world phenomenon: an algorithm perspective. In: STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing, pp. 163–170. ACM, New York, NY, USA (2000). DOI http://doi.acm.org/10.1145/335305.335325
31. Kontiki: Kontiki. URL `http://www.kontiki.com/`
32. Le Fessant, F., Handurukande, S., Kermarrec, A.M., Massoulié, L.: Clustering in peer-to-peer file sharing workloads. In: IPTPS (2004)
33. Lebedev, D., Mathieu, F., Viennot, L., Gai, A.T., Reynier, J., de Montgolfier, F.: On using matching theory to understand p2p network design. In: Proceedings of the International Network Optimization Conference, pp. 1–6 (2007)

34. Lin, Y.J., Guo, K., Paul, S.: Sync-ms: synchronized messaging service for real-time multi-player distributed games. In: Proc. of the 10th IEEE International Conference on Network Protocols (2002)
35. Manlove, D.: The structure of stable marriage with indifference. Discrete Applied Mathematics **122**, 167–181 (2002)
36. Manlove, D.F., Irving, R.W., Iwama, K., Miyazaki, S., Morita, Y.: Hard variants of stable marriage. Theoretical Computer Science **276**(1-2), 261–279 (2002)
37. Mathieu, F.: Upper bounds for stabilization in acyclic preference-based systems. In: SSS, pp. 372–382 (2007)
38. Mathieu, F.: Self-stabilization in preference-based systems. Peer-to-Peer Networking and Applications **1**(2), 104–121 (2008)
39. Mathieu, F., Postelnicu, G., Reynier, J.: The stable configuration in acyclic preference-based systems. Tech. Rep. RR-6628, INRIA (2008)
40. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: IPTPS, pp. 53–65 (2002)
41. Meridian Project: Http://www.cs.cornell.edu/People/egs/meridian/
42. Qiu, D., Srikant, R.: Modeling and performance analysis of bittorrent-like peer-to-peer networks. In: Proceedings of ACM SIGCOMM, pp. 367–378. ACM, New York, NY, USA (2004). DOI http://doi.acm.org/10.1145/1015467.1015508
43. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 161–172. ACM, New York, NY, USA (2001). DOI http://doi.acm.org/10.1145/383059.383072
44. Ronn, E.: On the complexity of stable matchings with and without ties. Ph.D. thesis, Yale University (1986)
45. Roth, A.E.: The evolution of the labor market for medical interns and residents: A case study in game theory. Journal of Political Economy **92**(6), 991–1016 (1984)
46. Roth, A.E., Sonmez, T., Utku Unver, M.: Pairwise kidney exchange. Journal of Economic Theory **125**(2), 151–188 (2005). Available at http://ideas.repec.org/a/eee/jetheo/v125y2005i2p151-188.html
47. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pp. 329–350 (2001)
48. SopCast: Sopcast. URL `http://www.sopcast.com/`
49. Sripanidkulchai, K., Maggs, B., Zhang, H.: Efficient content location using interest-based locality in peer-to-peer systems. In: INFOCOM (2003)
50. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 149–160. ACM, New York, NY, USA (2001). DOI http://doi.acm.org/10.1145/383059.383071
51. Sung, Y.W., Bishop, M., Rao, S.: Enabling contribution awareness in an overlay broadcasting system. SIGCOMM Computer Communication Review **36**(4), 411–422 (2006). DOI http://doi.acm.org/10.1145/1151659.1159960
52. Tan, J.J.M.: A necessary and sufficient condition for the existence of a complete stable matching. Journal of Algorithms **12**(1), 154–178 (1991). DOI http://dx.doi.org/10.1016/0196-6774(91)90028-W
53. TVants: Tvants. URL `http://tvants.en.softonic.com/`
54. UUsee: Uusee inc. URL `http://www.uusee.com/`
55. Zhang, X., Liu, J., Li, B., Yum, T.: Coolstreaming/donet : A data-driven overlay network for peer-to-peer live media streaming. In: INFOCOM (2005)
56. Zhao, B.Y., Kubiatowicz, J., , Joseph, A.D.: Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. UCB/CSD-01-1141, U. C. Berkeley (2001)

Part XI
Measurement and P2P Traffic Characteristics

The Behavior of Free Riders in BitTorrent Networks

Manaf Zghaibeh, Kostas G. Anagnostakis, and Fotios C. Harmantzis

Abstract In this chapter we report on the results of two large-scale measurement studies on BitTorrent. We focus on identifying the strategic behavior of users in response to the incentives that are embedded in the design of BitTorrent. Our results regarding the first study that was performed in year 2005 illustrate a gap between what system designers and researchers expect from users in reaction to the provided incentives, and how users react to them. In particular, we divide users in BitTorrent into four classes based on their interactions with the system. We also provide a measurement of free riders' volume in the BitTorrent environment.
On the other hand, the results of the second study that we conducted in year 2007 show that free riders' population has significantly changed comparing to our first measurement study. We categorize free riders based on the behavior they exhibit in multiple-torrent system into three types: cheaters, strategic and lucky peers. Furthermore, we refute the findings of other studies that suggested that free riders belong to a specific bandwidth capacity class. We also argue that the Tit-for-Tat mechanism reacts successfully against inter-class bandwidth capacity strategic peers. Finally, we present our vision to a modified approach of the optimistic unchoke policy in BitTorrent.

Manaf Zghaibeh
Stevens Institute of Technology, Hoboken, NJ 07030, USA,
e-mail: manaf.Zghaibeh@stevens.edu

Kostas G. Anagnostakis
Institute for Infocomm Research, Heng Mui Keng Terrace 119613, Singapore,
e-mail: kostas@i2r.a-star.edu.sg

Fotios C. Harmantzis
Stevens Institute of Technology, Hoboken, NJ 07030, USA,
e-mail: fotios.harmantzis@stevens.edu

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_43, © Springer Science+Business Media, LLC 2010

1 Introduction

Many Peer-to-Peer (P2P) content distribution systems are susceptible to threats and attacks. Some of these attacks are related to the persistent vulnerabilities in the design of those systems, the innovated methods the attackers use, or both. In some cases, such threats and attacks may lead to performance limitations that could devastate the P2P system, i.e., MojoNation [5]. *BitTorrent* [2], our topic in this chapter, is also a subject to attacks that substantially affect its performance even that they have not degraded the system as a whole yet. Accounting about 53% of all P2P traffic in 2004 [3], BitTorrent emerged in the last few years as one of the most important applications in the Internet. It attracted millions of users mainly because of its relatively high download speeds compared to other P2P file sharing applications.

The most significant threat that has been addressed in the BitTorrent environment is the free riding threat. In P2P, a *free rider* is a user who obtains resources from the network without contributing back to it, either selfishly or unintentionally. This behavior leads to a tangible level of unfairness to the remaining participants who contribute. Free riding has been the major threat that jeopardized the performance of most of the P2P content distribution protocols. Significant amount of research has been done on detecting, measuring, and proposing solutions to that problem. However, some of those remedies were not able to overcome the inherited design limitations of some of those systems.

In general, P2P content distribution mechanisms rely on the cooperation among their participants in order to realize the benefits of their systems. However, although cooperation is a virtue and highly appreciated in P2P networks, it might look as naivety to some users, not necessary, or costly to others. System designers have considered to build mechanisms that address the last two concerns: necessity and cost. First, when designing a P2P mechanism, the proposed mechanism has to ensure that cooperation is necessary to advance the position of the user in the sharing process. Second, the mechanism must take into account reducing the cost of sharing, and increasing the cost of selfish behavior. Yet, with regard to the naivety concern, spreading cooperation as a virtue in P2P system is not a norm that system designers were able to handle, at least not now. In other words, there will always be selfish and malicious users who will always rummage around for methods to circumvent the system and spread their values, i.e., selfishness.

For BitTorrent, its specification unambiguously indicates that its mechanism reduces the *cost* of sharing by redistributing the load from one single node to multiple nodes who participate in the sharing process [10]. The specification also suggests that undertaking a cooperative behavior is *necessary* for peers to enhance their positions, especially that the mechanism employs a reciprocal exchange process, i.e., *Tit-for-Tat* [7]. Furthermore, the design of BitTorrent emphasizes that its incentive mechanism is robust against selfishness and malicious behavior [10]. However, recent studies have argued and suggested that even that BitTorrent is renowned for having a reputation of fairness and robustness, its incentives are not sufficient enough to deter users from thwarting the system [8, 13, 15, 16, 18].

In this chapter, we analyze and investigate the behavior of users in BitTorrent. The chapter is composed of the results of two large-scale measurement studies that we performed on BitTorrent. The first part of the chapter includes the findings of the first measurement study that was conducted in years 2005–2006 [22]. In that study, we show how users in BitTorrent recognize the incentives provided in the system. We also question whether those incentives are sufficient enough to drive users towards the desired level of cooperation. Moreover, we classify the users of BitTorrent into four clusters based on their contribution to the system: major contributors, users with balanced download and upload behavior, users who downloaded at least twice as they uploaded, and free riders.

In the second part of the chapter we present the findings of the second measurement analysis study which we performed on BitTorrent in year 2007 [23]. In this most recent study we revisit the free riding problem from another point of view. We isolate and characterize free riders into three classes: lucky peers, strategic peers, and cheaters. We try to determine whether free riders in BitTorrent belong to a specific bandwidth capacity category, since there has been some detailed research suggesting that peers who exploit the system in BitTorrent are high bandwidth capacity peers [13], while another study argued that they are low bandwidth capacity peers [8].

Moreover, some researchers indicated incompetency in the role of Tit-for-Tat when it performs between inter-class bandwidth capacity peers, i.e., between high bandwidth capacity peers and low bandwidth capacity peers. Therefore, we question this matter; we take further steps in analyzing the performance of BitTorrent in order to determine whether the employed Tit-for-Tat is to blame for its shortcomings and inefficiency, if exist. Finally, we present our modified approach to the optimistic unchoke policy that is meant to reduce the effects of free riding.

2 BitTorrent Background

BitTorrent [2] is a content distribution system that lowers the cost of sharing by redistributing the load from one peer to multiple peers. This is achieved through many steps in the sharing process. When a user wishes to share a content that he owns using the BitTorrent system, he has first to create a meta-data file using a BitTorrent client. The process of creating the meta-data file (its size is usually less than 100 KBytes) logically divides the content into small equal size pieces (24 KBytes-4 MBytes), where each piece will be transferred as single chunk during the sharing process. The meta-data file, which is called a *torrent*, includes several information about the content to be shared such as the name of the content, its size, the hashes of the pieces, and the address of the *tracker*. The tracker is a central node that is responsible for helping peers find each other. It keeps a list that contains the addresses of all peers who participate in distributing the content and their positions in the download and upload process. After creating the .torrent file, the user uploads it to a public server (website) where it will be available to others.

If a user wants to obtain the content, he first downloads the .torrent file form the website. Then he initializes the download process using the BitTorrent client. The client reads the information embedded in the meta-data file and contacts the tracker notifying it with the name of the content. The tracker responds with a list that contains a partial random set of the addresses of peers who are joined in the *swarm*.[1] It is the responsibility of the peer to contact the members of the list and start downloading the pieces of the content. After downloading each piece, the peer verifies its integrity by checking the hash information in the torrent file (SHA-1 stamps). Please note that the user who announced his content for sharing must keep at least one client offering the complete content to be downloaded by the peers in the swarm. However, this is only required until the user uploads all the pieces of the content to the remote peers. Finally, BitTorrent defines two types of peers: *Seeder* and *Leecher* peers. A seeder is a peer who owns a complete copy of the content and shares it, while a leecher is a peer who does not own any piece or has partial copy of the content, i.e., not all the pieces.

To facilitate the distribution process, BitTorrent implements a rate-based Tit-for-Tat incentive mechanism to reciprocate services among peers. The Tit-for-Tat mechanism is known to be a highly effective strategy for the iterated prisoner dilemma [7]. The two-player multi-round game starts when the first player cooperates at the beginning of the game, then he responds based on the previous action of his opponent. If the second player keeps cooperating, the first player will always cooperate. If the second player defects, the first player will retaliate. This strategy is fully deployed in BitTorrent using the *choke*, *unchoke* and *optimistic unchoke* policies. Although a peer in BitTorrent might be connected to many other peers in the swarm, he typically uploads to a small number of them, based on those policies. The peer unchokes other peers, whom he had received data from recently, by uploading to them. The unchoke process starts the reciprocal exchange of data between two peers, which is mainly intended to restrain free riding behavior among peers. On the other hand, when the peer does not receive an acceptable level of service from his opponent, or when his opponent defects from the reciprocal exchange, the local peer retaliates by choking him and refraining from uploading to him. Moreover, the peer always optimistically unchokes one randomly selected peer at a time, regardless of his cooperation level, in an attempt to provoke him into cooperation. This policy is important in order to discover new peers with better rates comparing to what the peer is being connected to, and replacing them with faster ones if necessary. Finally, in the Tit-for-Tat, peers forgive quickly, i.e., there is no harboring in the game.

3 Previous Work

The incentive techniques of BitTorrent have been discussed in many publications. Feldman and Chuang [11] argue that the Tit-for-Tat mechanism employed in

[1] A swarm is a group of peers who are joined in a single torrent.

BitTorrent helped increase the cooperation level among peers. Similarly, Andrade et al. [6] emphasize that the reciprocity algorithm in BitTorrent makes the system appealing to be used and that the design of BitTorrent increases cooperation among peers. The paper also suggests that in some cases and due to the large number of seeders, BitTorrent fails in reducing free riding, since there is no specific mechanism embarked to limit their gains. On the other hand, Jun and Ahamad [15] argue that BitTorrent lacks fairness: It does not punish free riders effectively, neither it does reward users who contribute properly.

The unchoke mechanism is also discussed in another bulk of the literature. Cohen [10], the creator of BitTorrent, presents the unchoke mechanism as the only efficient method used in BitTorrent to maximize the download rates of the peers. Legout et al. [16] evaluate the unchoke mechanism in BitTorrent and question its efficiency in providing reasonable reciprocation in balancing upload and download rates. The paper concludes that the unchoke mechanism provides fairness to leechers in connecting to others and a reasonable level of reciprocation.

Piatek et al. [18] found that altruistic behavior exhibited by some peers is the dominant factor in expediting BitTorrent transfers. The authors argue that the Tit-for-Tat mechanism has little to do with the altruism of the peers and the performance of BitTorrent, and that BitTorrent is still working today because most of the peers are not trying to subvert the system by tweaking their clients.

Qiu and Shroff [20] analyze the Tit-for-Tat mechanism deployed in BitTorrent. The study concludes that the system converges to a Nash equilibrium when peers utilize their pre-set upload capacities and try to maximize their downloads. Another study also targeted the Tit-for-Tat mechanism and found that the robustness of BitTorrent is not related to it. Halesand and Patarin [14] argue that the presence of altruistic peers in swarms is the major factor behind the robustness of BitTorrent. In some cases when swarms are being highly infested with selfish peers, the authors argue that BitTorrent does not reach to expectations and that such swarms tend to die fast.

Liogkas et al. [17] tested the robustness of BitTorrent against selfish peers. The authors deploy three kinds of exploits that try to abuse the system in order to achieve higher download rates than other peers. They conclude that BitTorrent proved to be robust against their attacks even that their strategic clients were able to exploit the system. Moreover, the study claims that the optimistic unchoke policy fortifies the robustness of the system by giving leechers chances to connect to fast leechers or seeders.

Guo et al. [13] state that BitTorrent lacks fairness where peers with high bandwidth capacities tend to exploit the system and download much more than they upload. Moreover, the authors also claim that BitTorrent provides fluctuating download services and poor availability services. Bharambe et al. [8] suggest that the Tit-for-Tat mechanism is not efficient enough in deterring unfairness, and relates this inadequacy to the heterogeneity of peers' bandwidth capacities. The authors also argue that low bandwidth capacity peers can download more than they upload when high bandwidth capacity peers are present in swarms.

Our work is different from the existing studies on BitTorrent. It provides a measurement of the volume of free riders in a real world BitTorrent trace. It classifies users in BitTorrent based on the behavior they exhibit into different clusters. Moreover, this study also characterizes and identifies free riders based on two attributes, first, the behavior they reflect in multiple-torrent file condition, and second, their bandwidth capacity. Finally, we consider this study as an in-depth analysis of the Tit-for-Tat mechanism when it functions between inter-class bandwidth capacity peers, to the best of our knowledge no other study has considered this angle of analysis for the Tit-for-Tat.

4 Methodology

In this section we describe the methodology we followed in our work. We begin by justifying our choice of using the measurement method, then we describe the tools that we employed to collect BitTorrent samples and our approach in luring peers to keep them interacting with our system and tools.

Studies on P2P had been performed using modeling [12, 13, 20], simulation [5, 21, 24, 25], and measurement [4, 9, 19] methods. In this chapter, we believe that our goals can be more accurately pursued using measurement analysis. Our choice is based on the need to capture how BitTorrent actually performs under realistic scenarios. Furthermore, we aim in this study to identify free riders and measure their volume in BitTorrent, and investigate how real users interact with the system. This can be efficiently deduced from data collected from actual peers.

4.1 First Measurement Study

Table 1 First measurement parameters

Number of trackers	2
Number of torrents	700
Average content size	630 MByte
Number of peers	65,063
Average number of peers per torrent	93
Number of ctorrent clients	4

In the first measurement study we utilized the following methods: (i) A modified BitTorrent client that is able to aggressively request new peers from the main tracker and try to connect to as many peers as possible to track their download progress and their connection time. The tool is developed using a robust C-language BitTorrent client called *ctorrent*. We modified the ctorrent client into *ctorrent-bigbro* to be able to only connect to the remote peers and monitor their progress without being involved in any download or upload activity, to avoid accidentally accessing illegal content. The ctorrent-bigbro provided us with the download volumes of remote peers and their connection times only. However, since the remote peer does not report its uploaded volume, we calculated this attribute from the IP-ID field which is embedded within each packet sent by the remote peer.

(ii) The other method we used for getting detailed information about the performance of BitTorrent peers was obtaining the information directly from trackers. Trackers provided us with full logs of the activities of all peers, including their download/upload volume and their connection time.

In the first measurement we traced around 1000 torrents, involving more than a hundred thousand peers. After weeding out the uninformative torrents we ended up with 700 torrents that gathered more than 65,000 peers.[2] Table 1 lists some of the measurement parameters.

4.2 Second Measurement Study

In collecting our data for the second measurement we also used two methods: a passive and a proactive one. (i) For the passive method we ran multiple trackers each on an independent platform. The trackers provided us with complete logs of the activities of all peers. The platforms we used to run the trackers were robust enough to perform the tracking critical functions. Each platform was dedicated solely for the purpose of running its specific tracker. Furthermore, each platform was adequately provided with a network interface that was capable of handling the requests of large number of peers. During our sampling, we did not have any discontinuity in our traces due to congestion problems at the trackers.

(ii) The proactive method we used in the second measurement study was deploying a number of BitTorrent clients. Each of those clients was a *BitTornado* [1] client that had a function of participating in the download and upload processes of the offered torrent. As we will discuss later, we classify peers based on their connection speeds into two types: high bandwidth capacity peers and low bandwidth capacity peers. Accordingly, our clients played the roles of strategic and non-strategic peers who joined the swarms as low bandwidth and high bandwidth capacity peers. The purpose of this method is to help us better understand how different peers in

[2] Because of the discrepancy of the IP-ID method that we used in measuring the uploaded volume by the remote peers, we ignored the samples in which the total uploaded volume did not match or was not close to the total downloaded volume by all peers.

Table 2 Second measurement parameters

Number of trackers	4
Number of contents	139
Content type	Number of contents
3-part content	58
4-part content	41
5-part content	40
Content type	Number of torrents
3-part content	174
4-part content	164
5-part content	200
Average part size	800 MByte
Number of peers	312,047
Average number of peers per torrent	580
Number of BitTornado clients	32

the swarm who belong to different bandwidth capacities interact with our special clients. That entitles us perceive how the Tit-for-Tat performs between peers in inter-class bandwidth capacity classes.

The trackers hosted different sets of unique torrents that we created using the BitTornado clients. Each content we offered consisted of three to five parts, with each part having a unique torrent associated with it. Using this approach in offering contents leads to better isolate peers and classify them based on their behaviors. Moreover, following this approach attracted peers to stay connected to our trackers for longer times which assisted us studying their behaviors more comprehensively. For example, a peer who appeared to be a free rider in one of his downloads could have possibly achieved that based on either a selfish behavior, i.e., *strategic* peer, or malicious behavior, i.e., *cheater*, or he could have just been a *lucky* peer. However, using the multiple-torrent content method at least diminished the incomprehension in identifying lucky peers from strategic or cheater peers. In other words, a peer who cheated in obtaining the first part of the content will most likely cheat in obtaining the remaining parts; this argument also applies on strategic peers. Yet, a peer who got lucky downloading a part of the content without contributing back to the system, will not necessarily get lucky obtaining the remaining parts of the content.

We ran four independent trackers that hosted 139 contents.[3] The contents were arranged as follows: 58 3-part contents, 41 4-part contents, and 40 5-part contents, a

[3] All the contents we offered were not intellectually protected.

total of 538 torrents. The sizes of shared parts varied between 700 and 1050 MBytes with an average of 800 MBytes per part. The total number of peers joined our system reached 312,047 peers, with an average of 580 peers per torrent. Finally, for the second method, we used 8 BitTornado clients per tracker. Table 2 summarizes the parameters of our measurement.

5 Results and Discussion

In this section we present the findings of our two measurement studies that we performed on BitTorrent.

5.1 First Measurement Study

We analyze our data in an attempt to understand how users in BitTorrent perceive and respond to the incentives provided; and whether the incentives are strong enough to drive users to a satisfactory level of cooperation.

We investigate how users follow different strategies by monitoring their download and upload behaviors and deducing their cooperation levels. The strategies are set by fine-tuning the parameters of the BitTorrent client. Some of these parameter changes could boost the download progress of users and advance their gain compared to others who probably are not aware of the existence of these options or their significance. Such behavior induces free riding. There was no clear understanding about the magnitude of free riding in BitTorrent before this measurement study. Most of the work on BitTorrent prior to this study is likely to focus on the Tit-for-Tat mechanism and particularly whether it is an effective technique to deter free riding.

As mentioned in Section 2, BitTorrent uses the Tit-for-Tat strategy that is known to provide a strong incentive for cooperation. It reciprocates services to nodes that cooperate and periodically chokes nodes that choose not to cooperate. Moreover, BitTorrent relies on the optimistic unchoke policy in order to discover new peers and to encourage other peers to cooperate. However, although this policy is used to incite peers who do not provide reciprocal services to cooperate, we suspect that it also furnishes opportunities for peers to free ride.

In each of the torrents in this study we isolated four different clusters of BitTorrent users. Each cluster is characterized by different level of cooperation exhibited by its members.

The first cluster we isolated includes users who profited from the system as much as they benefited it: The volume downloaded by those users almost equals the volume they uploaded. Contemplating in this behavior, it seems that the members of this cluster, who comprise the majority of users in our samples, did not follow any definite approach in downloading or uploading their files. Or in other words, they just simply ran the BitTorrent client on its default without changing any of its settings.

The second cluster we isolated in our samples contains peers who we classify as the major contributor leechers in the system. Their high cooperation level is clearly seen in the considerable volume they uploaded which is more than twice than the volume they downloaded. This behavior gives the impression that the members in this cluster were either altruistic or, as the members in the first cluster, did not follow specific strategy in their interaction with other peers. However, they are differentiated from the peers in the first cluster by their higher bandwidth capacities. Nonetheless, we suggest that members in clusters 1 and 2 are the honest peers who responded to the incentives provided since they exhibited medium to high levels of cooperation. However, it is not clear whether their reaction to the incentives was driven by altruism, carelessness, or by the lack of knowledge of the possibility in exploiting the resources of the system and free riding.

The third cluster in our samples includes free riders: Users who completed their downloads without contributing to the system. The maximum upload speed we observed from those users was less than 2 KB/s. On the other hand, they were fast enough downloading with speeds reached up to 300 KB/s. Moreover, as an expected selfish conduct, those peers left the system as soon as they completed their downloads.

The last group contains users who downloaded at least as twice as they uploaded. They are also characterized by disconnecting immediately after they completed their downloads. They remarkably benefited from the system as well. However, their limited contribution to the system makes them distinguishable from free riders. This behavior is explicable if we assume that those peers are fairly acquainted to BitTorrent mechanisms, or at least, aware of the parameters put available in most of BitTorrent GUI clients, thus, allowing them to follow advanced sharing strategies compared to peers in clusters 1 and 2. Based on this classification, it is obvious that peers in cluster 2 are structuring an enjoyable downloading source to peers in this cluster and to free riders.

In general, using the operational parameters in BitTorrent helps users to strategize themselves to augment their gain on the account of other users. However, we suspect that the use of such parameters has counter effects on the functionality of the optimistic unchoke mechanism. For example, a peer can easily change the maximum number of incoming connections that are allowed, which would increase the probability of having incoming connections that periodically optimistically unchoke him. This presumably makes the optimistic unchoke mechanism in BitTorrent susceptible to parameters tweaking. Nevertheless, as long as the population of such peers is limited in BitTorrent, the system will still be capable of performing well and will still be an attractive choice for users. Yet, if the population of such peers significantly increased, we believe that the optimistic unchoke would be incompetent in handling such scenario.

On the other hand, the behavior of free riders is possibly related to either more advanced strategies compared to those adopted by peers in cluster 4, or to cheating. With some modifications to a BitTorrent client, users will be able to apply several threat scenarios that allow them to completely free ride without contributing to the system. Some threat scenarios are very well known now: Serving false pieces, connecting directly to seeders, and using multiple identities. Therefore, since the upload volumes of such peers is very limited, we are inclined to believe that this behavior is closer to cheating than parameters tweaking.

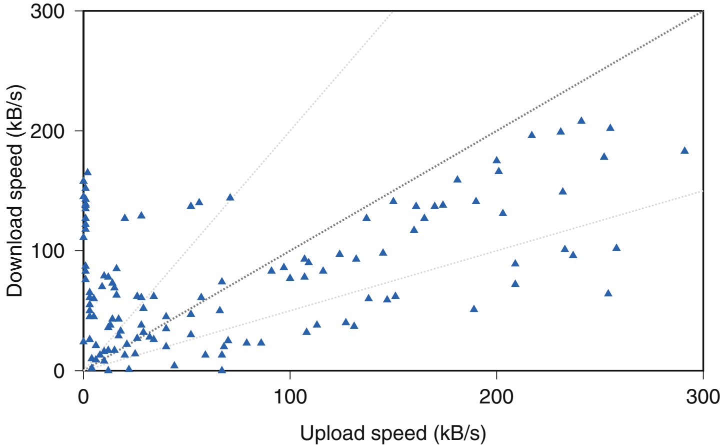


Fig. 1 Upload speed vs. download speed for a sample torrent

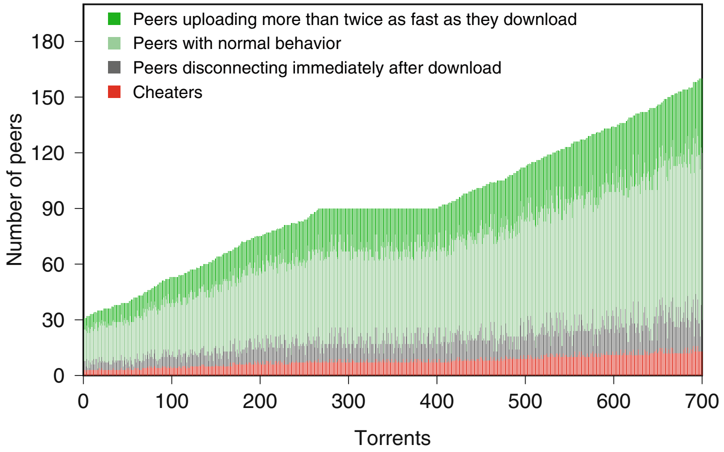


Fig. 2 Distribution of peers joined in 700 torrents

Figure 1 demonstrates the distribution of peers joined in a torrent based on their download and upload speeds. The figure shows that free riders are positioned on the download speed axis, users in cluster 4 are located between the download speed axis and the $y = 2x$ line, where the $x = 2y$ line separates users in clusters 1 and 2.

To get a comprehensive view of the clusters we identified above, Fig. 2 illustrates the fraction of each cluster in each torrent in our samples. The figure shows that the majority of users belong to cluster 1 and cluster 2. Furthermore, we notice that at least 50% of users in all our samples uploaded as much as they downloaded, i.e., cluster 1; where users in cluster 2 who have been taken advantage from formed about 17%. On the other hand, cluster 4, users who disconnected immediately after they completed their downloads, comprised about 10% of the total number of users in a torrent, while free riders population reached about 10%.

5.2 Second Measurement Study

We investigate the free riding problem more meticulously by categorizing free riders in view of their upload volumes to the rest of the network. Based on their types, we try to determine whether free riding is a behavior that is associated with a particular class of bandwidth capacities, i.e., high bandwidth capacity peers or low bandwidth capacity peers. Finally, we examine the robustness of the Tit-for-Tat mechanism against different bandwidth capacity strategic and non-strategic clients.

5.2.1 Free Riders: Who Are They?

In the previous subsection we limited our analysis regarding free riders to their volumes and speeds. In this part of the analysis and with the most recent data that we collected, we try to take the analysis steps forward to comprehend the behavior of free riders and identify them more clearly.

We suggested that the *free rider* term is used to recognize peers who exploit the design limitation of the P2P system intentionally or unintentionally and download without contributing back to it, or delivering minimum upload volumes comparing to the volumes they download.

Free Riders Volume: Previously we found that the free riders percentage in our 2005 samples reached up to 10% of the total population of the peers who joined the torrents. The maximum upload speed we observed from those peers was less than 2 KB/s, while their download speed reached in some torrents 300 KB/s. However, in our most recent samples which we collected in 2007 [23], we find that those numbers have noticeably changed comparing to what they were before. Figure 3 illustrates free riders percentage in our data samples in ascending order. Each point on the x axis represents a torrent from our samples, while the y axis signifies free riders percentage (only shown up to 20% for resolution purposes). In all our samples, free riders were present. Their volumes varied between a minimum percentage of 2.8, and surprisingly, a maximum percentage of 16.8 of the total population of peers.

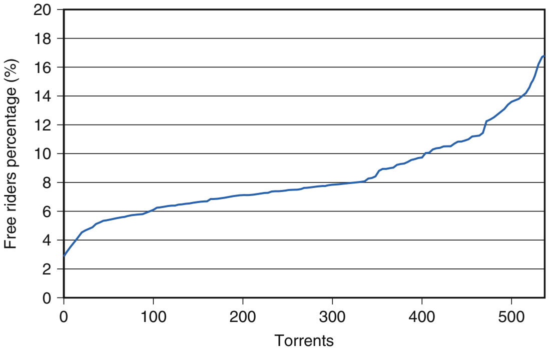


Fig. 3 The percentage of free riders in 538 torrents

The Cumulative Distribution Function (CDF) of the percentage of free riders clearly describes their presence in our samples, Fig. 4. The figure indicates that in about 20% of the torrents the percentage of free riders varied between 10.8 and 16.8, while in 75% of the torrents their percentage was between 2.8 and 9.

Dissecting Free Riders: In the next step of our analysis on free riding we dissect the samples with respect to the convention we suggested previously to determine their types, i.e., strategic, cheaters, or lucky peers. Our goal in this phase of the study is to categorize free riders based on their interaction with other peers in multiple-torrent contents. That will help us later to understand their strategies in case they follow any, or realize any deficiency in BitTorrent design, if exists.

In Fig. 5 we notice the categorization of free riders. In our convention here we presume that a *cheater* is a free rider with a zero upload volume, and that a *strategic* peer is a free rider who scarcely uploaded to other peers. We indicate that this perception is related to the levels of participation of free riders in the multiple-torrent contents: 3-part, 4-part, or 5-part contents. For example, a cheater would exhibit the same 0 KByte upload volume in at least two of the multiple-torrent contents which he participated in, while a free rider who reflected the same limited upload behavior in at least two torrents of multiple-torrent content would be classified based on our norm, as a strategic peer. Finally, if a peer showed a free riding behavior in one torrent and normal cooperative behavior in other torrents, we would classify him as a *lucky* user. Please note that the criterion we use here to sort free riders is only based on the volume they upload in multiple-torrent contents, not on their bandwidth capacities.

As Fig. 5 shows, the majority of free riders in our samples are strategic peers. Their individual upload contribution was less than 5% of their individual download volume. On the other hand, cheaters who exhibited unique 0 KByte upload volumes comprised about 8% of the total population of free riders. Interestingly, lucky peers who free rode in a single torrent of a multiple-torrent content and behaved cooperatively in other parts comprised about 22% of free riders population. Finally, the remaining 24% of free riders represents those who only participated in a single torrent.

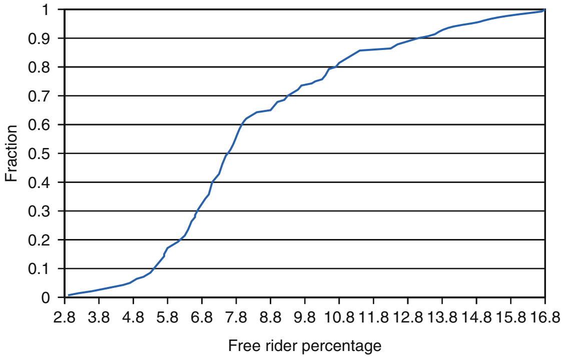


Fig. 4 CDF of the percentage of free riders

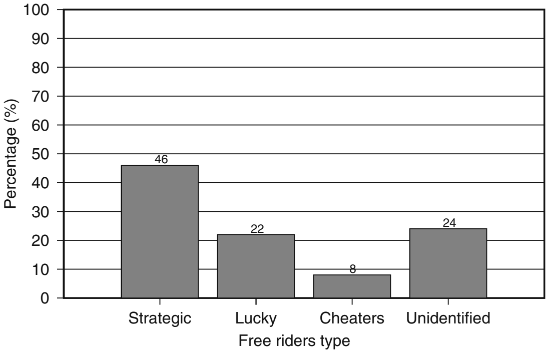


Fig. 5 Free riders type

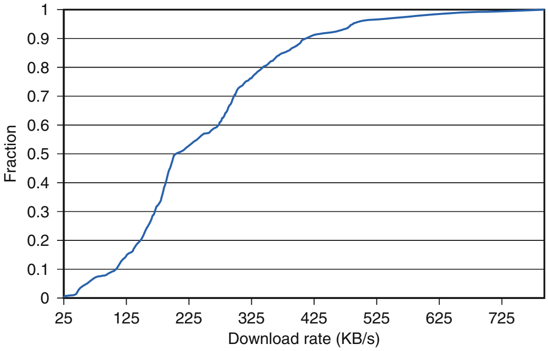


Fig. 6 CDF of free riders download rate

Free Riders Download Rate: Before we conclude the analysis on identifying free riders in our samples, we present in Fig. 6 their download speeds. The figure demonstrates the Cumulative Distribution Function (CDF) for download rates of all free riders in our samples. As we stated before, in the 2005 study we observed that free riders speeds cropped up to 300 KB/s. However, and as Fig. 6 shows, the average download rate reached up to 790 KB/s as an upper bound for some free riders.

5.2.2 Who Exploits BitTorrent?

The next step in our analysis study is to determine the nature of peers who take advantage of BitTorrent's *unfairness*. Those peers either free ride or flourish, in the sense that they benefit from the system without providing satisfactory levels of contribution to assist other peers. Many measurement, analytical, and modeling studies have labeled this trend as the dark unfair side of BitTorrent [6, 15, 16]. Moreover, some of those studies have related this theme to the heterogeneity in peers' bandwidth capacities, denoting that peers within a class of specific bandwidth capacity are taking advantage of the system, as well as of peers who do not belong to the same class [8, 13]. However, researchers disagreed on which class of bandwidth capacity is exploiting the system. Two major arguments exist here:

- High bandwidth capacity peers exploit the system: Guo et al. [13] argue that peers with high bandwidth capacities tend to download more than they upload, although the authors mention that the performance of the peer in such systems fluctuates widely with the population of the peers.
- Low bandwidth capacity peers exploit the system: Bharambe et al. [8] contradict [13] and state that low bandwidth capacity peers tend to download more than they upload.

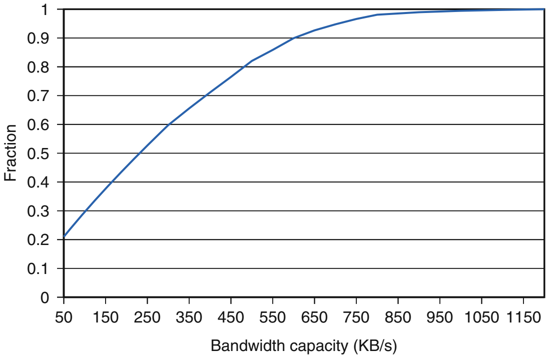


Fig. 7 CDF of the bandwidth capacities of peers in our samples

However, in our samples, neither of these two arguments apply. We did not find among the peers who participated in the offered torrents a trend that supports either of the two previous statements. Figure 7 shows the Cumulative Distribution Function (CDF) of the bandwidth capacities of the peers in all our samples. It indicates that about 23% of those peers have bandwidth capacities within the range of 0 and 50 KB/s, while the majority of peers have bandwidth capacities between 100 and 650 KB/s. Figure 8 shows that peers who exploited the system in our samples diversified in their bandwidth capacities. Based on their upload and download volumes, 27% of peers who downloaded at least twice the volume they uploaded (including free riders: strategic, lucky, and cheaters) had bandwidth capacities less than 50 KB/s, while the majority of those peers belonged to a class of bandwidth capacities between 100 and 800 KB/s. However, these two ratios almost evenly split between the two types of bandwidths when their percentages are normalized based on the population of peers in our samples.

Therefore, our finding opposes the arguments about the bandwidth capacities of peers who take advantage of the system in BitTorrent. We argue in this stage of the study that the gain which those peers accomplished by taking advantage of others is not related to their bandwidth capacities, yet, to another cause as we will discuss later. We find that peers who flourished and exploited the system on the account of others belong to different classes of bandwidth capacities. That is, peers with small bandwidth capacities were able to exploit the system as much as high bandwidth capacity peers did, with respect to their population.

However, it is important to clarify that measurement analysis studies usually differ in their conditions. For example, in the 2007 study the population of peers with high bandwidth capacities has increased compared to what it was in the 2005 study. This could be related to the advance of the Internet backbone speed and the connection speeds of the users. Moreover, it is clear that BitTorrent users have become more acquainted with the system and the protocol, which is due to the increased popularity of the system.

5.2.3 Is the Tit-for-Tat Robust?

Many questions arise on how peers with different bandwidth capacities are able to significantly gain from the system, and who to blame about this cooperation inadequacy in the system. Accordingly, many answers emerged trying to justify the behaviors of peers and clarify the shortcomings of BitTorrent. However, studies which attempted to answer the first question usually targeted the altruism and the egoism of peers [18], while making the *Tit-for-Tat* incentive mechanism in BitTorrent responsible for the unfairness became the major justification for the second question [8, 14].

Tit-for-Tat is the successful outcome of the iterated prisoner dilemma that is used in BitTorrent [7]. Moreover, the *optimistic unchoke* is the first move in the Tit-for-Tat game. A peer optimistically unchokes his opponent in order to provoke him into cooperative exchange of data blocks. Based on the reaction of the opponent to this move, the peer will cooperate with his opponent if he reciprocates the service, or, he will choke him if he declines the cooperation.

In general, we do not expect optimality in the performance of Tit-for-Tat. Even if there is an imperfection in its role, the question here is whether this imperfection is acceptable or whether it is beyond the tolerable limits in which it is fundamentally

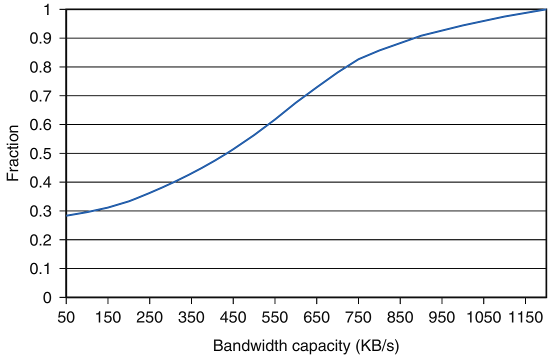


Fig. 8 CDF of bandwidth capacities of peers who downloaded at least as twice as they uploaded

harming the performance of BitTorrent. Therefore, it is essential to understand when the Tit-for-Tat succeeds and when it fails. Theoretically, the mechanism *succeeds* when the reciprocation of data blocks between two peers takes place. In addition to that, after the first move in the Tit-for-Tat in BitTorrent, i.e., the optimistic unchoke, if the remote peer declines the reciprocation of services, then the mechanism chokes him *successfully*. Therefore, conceptually, the Tit-for-Tat attained its goal: It rewarded cooperative peers by reciprocating services with them and punished the uncooperative ones by choking them.

On the other hand, the optimistic unchoke has been the main component to blame for the incompetency of BitTorrent. The rationale behind this assumption is based on losing the first move to uncooperative peer, i.e., uploading to the remote peer before discovering that he is uncooperative. Finally, there has been also some doubt about the ability of the Tit-for-Tat incentive mechanism to work in heterogeneity environment where the bandwidth capacities of peers differ, which was related to the unfairness that we discussed previously.

In this phase of the analysis we try to understand how the Tit-for-Tat performs in the wild. Our goal is to determine whether the Tit-for-Tat is to blame for the volume of free riding and the unbalanced contribution of peers that we observed previously. To pursue this target, we investigate this problem from a peer's level. We deploy the regular BitTorrent clients that we mentioned previously. The objective of this method is to monitor how peers with different bandwidth capacities interact with our BitTornado clients that have different participation strategies. The deployed BitTornado clients have only one advantage over other clients: they obtain from the trackers complete lists that contain the identities of all peers in the swarm, while regular clients receive partial lists. We configure the BitTornado clients to adapt the following behaviors and bandwidth capacities:

Low Bandwidth Strategic Clients (LBS): we set the download capacities of those clients to 20 KB/s, we limited their number of uploads to one, and their maximum upload rate to 1 KB/s. As Fig. 9 shows, peers who uploaded to our clients were both low bandwidth peers and high bandwidth peers; however, the major contributors were low bandwidth capacity peers with an average contribution level of 64%, while high bandwidth capacity peers contributed an average of 36%.

Low Bandwidth Non-strategic Clients (LBNS): we set the download capacities of those clients to 20 KB/s, and limited their maximum upload rate to 8 KB/s. In this arrangement, the performance of our clients was nearly balanced between uploading and downloading. Figure 9 shows that they were able to download the same volume from high bandwidth capacity peers and low bandwidth capacity peers.

High Bandwidth Strategic Clients (HBS): we limited the number of uploads to one, and the maximum upload rate to 1 KB/s. Our clients connected to low bandwidth capacity peers and high bandwidth capacity peers as well. The major uploaders to those clients in this case were high bandwidth capacity peers with an average upload volume of 71%. On the other hand, the contribution of low bandwidth capacity peers was limited to 23%.

High Bandwidth Non-strategic Clients (HBNS): we set their maximum upload rate to 300 KB/s. The contribution of high bandwidth capacity peers to our clients reached an average of 63%, while low bandwidth capacity peers uploaded an average of 37%.

To further investigate this point, we look into the choking messages our clients have received from the members of their swarms. Figure 9 gives us a better insight on how remote peers interacted with our clients and whether the Tit-for-Tat mechanism performed successfully with different types of peers related to different bandwidth capacities. As the figure demonstrates, LBS clients have been choked mostly by the high bandwidth capacity peers. On the contrary, HBS clients were choked mostly by the low bandwidth peers. Interestingly, neither low bandwidth capacity peers nor high bandwidth capacity peers discriminated against the non-strategic clients. The volumes of the choking messages that were received by our two types non-strategic clients were relatively close to each other and thus did not suggest a selective behavior from remote peers.

To conclude this part of the analysis, we argue that the Tit-for-Tat mechanism succeeds indeed in rewarding cooperative peers and punishing uncooperative ones, regardless of their bandwidth capacities. We also argue that the Tit-for-Tat does not discriminate peers based on their bandwidth capacities. The mechanism successfully reciprocates a fair exchange among peers with different bandwidth capacities. However, the mechanism reacts effectively against inter-class bandwidth capacity strategic peers. That is, BitTorrent successfully chokes high bandwidth capacity peers from LBS users and low bandwidth capacity peers from HBS users, when the Tit-for-Tat detects unbalanced reciprocation exchange. On another venue, we emphasize our previous interpretation that what drives successful exploitation of resources in BitTorrent is directly related to the first move in the Tit-for-Tat, i.e., the optimistic unchoke. For example, consider that the number of pieces of a shared content is 6,000 data chunks and the number of peers who join the swarm during the sharing process is about 600 peers. For a strategic peer who is only relying on the first move of the Tit-for-Tat and utilizing a single slot of his upload link, i.e., 1 KB/s, connecting to five remote peers concurrently would be a sufficient arrangement for him to free ride and to exploit the system. The strategic peer would reciprocate a

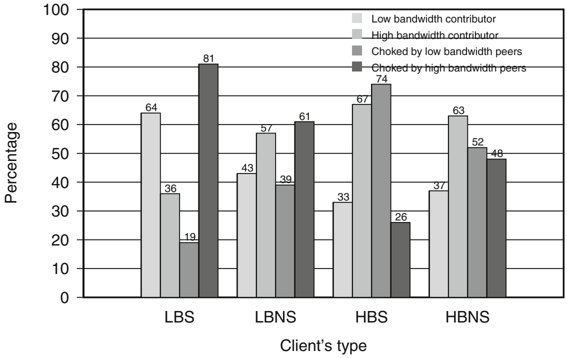


Fig. 9 Peers contribution levels and choking volume

service with one of the five remote peers using his single upload slot, while waiting for the other four peers to optimistically unchoke him. After optimistically unchoking him, if those four remote peers discovered that this strategic peer is not willing to reciprocate services, they would choke him. However, the strategic peer will be continuously probing other remote peers trying to connect to them and take advantage of the optimistic unchoke. Furthermore, the lack of memory in BitTorrent helps the strategic peer to re-exploit remote peers repeatedly. That is, in a worst case scenario, if the strategic peer was able to obtain 10 pieces from each remote peer using the optimistic unchoke, he would be entirely able to free ride, even without the need to use his single upload slot.

5.2.4 The Population of Peers and the Optimistic Unchoke

The size of the swarm also has a direct relation with the degree of exploitation in the system. In principle, a peer would punish his opponent when he does not reciprocate services with him; the punishment is achieved by choking the opponent. However, since this punishment is not perpetual, the choked peer could possibly be optimistically unchoked by the same peer who choked him before. Logically, if the size of the swarm is small, then the selfish peer will not find many peers to optimistically unchoke him, yet, his chances to be repeatedly optimistically unchoked by the peer who had previously choked him will increase. While if the size of the swarm is large, then the probability that the peer will optimistically unchoke one of his previously choked peers decreases because there are many other peers available in the swarm to choose from. However, in this case, the chance that the selfish peer will find more peers to optimistically unchoke him will increase. We study this conjecture by monitoring the download progress of our strategic clients in swarms that have different sizes, Fig. 10. The figure shows that strategic peers were able to finish their downloads in small populated swarms faster than they did in large swarms. Our explanation to this surprising finding revolves on the notion that our strategic peers in the heavily populated swarms were faced by tough competition from other remote peers who offered better services than they did. Thus, those peers were matched through the Tit-for-Tat with other members of the swarm leaving our clients in continuous search for peers to unchoke them. However, in small swarms, the competition was not strong and our clients were able to connect to other remote peers easily.[4]

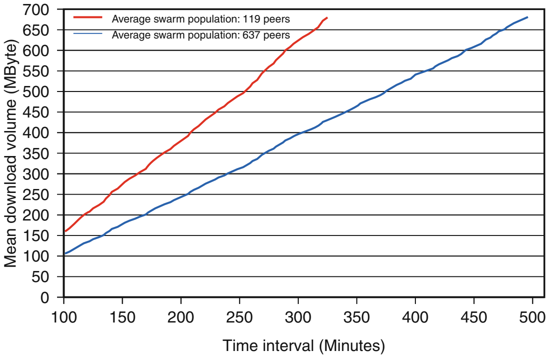


Fig. 10 Average download volume for strategic peers in different swarms

5.2.5 Modified Approach to BitTorrent

Although we argued that free riding in BitTorrent has a profound relation with the optimistic unchoke policy, however, bypassing the implementation of the policy would have devastating effects on the system because of its essential role in discovering the fast peers. We also suggested in the previous subsection that the no-harboring characteristic of the Tit-for-Tat makes the system susceptible to exploitation by selfish peers. Therefore, we propose a simple modification on the design of BitTorrent that requires the peer to remember the disappointing outcomes of the optimistic unchoke policy he performed on remote peers. This method is a protective policy to guard the peer from interacting with selfish clients frequently. However, the memory will be adaptive based on the behavior of the remote peers who are remembered.[5] For example, a remote peer who is recorded in the memory of a local peer will be tested after a period of time t by optimistically unchoking him. If he shows a cooperative behavior, the local peer will essentially continue interacting with him and remove him from the memory. On the other hand, if the remote peer reveals the same selfish behavior, then the local peer will keep him in his memory, yet, will try him again if needed after $2t$. Every time the remote peer reflects a selfish behavior, his waiting time in the memory table will be doubled.

[4] The percentage of strategic clients was 7% in small and large swarms.

[5] An exponential backoff rule.

We experiment this proposed policy in a 200-peer network. Each peer maintains a history table to save his interactions with the selfish peers. In this experiment, our goal is to test for the proposed policy, therefore, we monitor how successful the selfish peers were in obtaining the pieces of the file via the optimistic unchoke policy. The number of selfish peers in this experiment is 30 and the number of the pieces of the file is 800.

Figure 11 shows the preliminary results of our experiment. It demonstrates the number of downloaded blocks by selfish peers in our configuration. The figure signifies the original implementation of the optimistic unchoke policy and our proposed approach. The *x* axis in the figure represents the time slots of the experiment while the *y* axis stands for the mean number of pieces obtained by the selfish clients through the optimistic unchoke policy only. We find significant improvement in the performance of the system under the memory backoff approach. In this preliminary experiment, the memory backoff approach succeeds in reducing the effects of free riding that is caused by optimistically unchoking selfish peers repeatedly. As we notice in the figure, the number of blocks obtained by the optimistic unchoke policy in the original implementation of BitTorrent is much more than the number of blocks obtained via the memory backoff approach. Moreover, this approach is sufficient enough to initiate exchanges between new comers and other peers in the system.

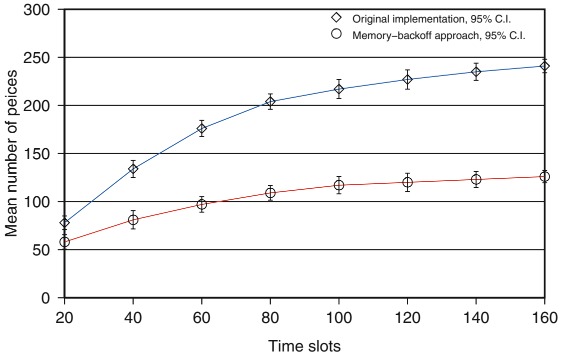


Fig. 11 Pieces obtained via the optimistic unchoke

6 Conclusions

In this chapter we presented the results of two major measurement analysis studies of BitTorrent file sharing system.

Our findings support that peers in BitTorrent might remarkably enjoy higher download speeds comparing to other P2P file-sharing systems. It appears that the incentive mechanisms adopted in BitTorrent are succeeding in promoting cooperation among peers until now. However, the level of cooperation in BitTorrent does not seem to be as satisfactory as expected.

Our results in the first measurement study show that while the majority of peers in our samples contributed to the system as much as they benefited from it, 17% of the peers were the main contributors to the system, and more than 10% of peers downloaded as twice as they uploaded to others. Moreover, the results show that the percentage of free riders in our samples reached up to 10% of the total population joined in a torrent. This is an indication that BitTorrent may not be providing a strong enough incentive to reduce free riding since there is no explicit mechanism in its design to punish or at least discourage free riding.

However, the results of the most recent measurement study that we performed on BitTorrent show that the volume of free riders is increasing in the BitTorrent environment. It reached up to 16.8% of the total population of peers joined in the torrents. We argue that this increase in their population is related to the advance in the connection speeds of users and their increased knowledge about the strategic options of BitTorrent clients. Using our multiple-torrent contents methodology, we classified free riders based on their upload volumes into three types: cheaters, strategic, and lucky peers. Our results show that strategic peers comprised the majority of free riders uploading up to 5% of their download volumes, while cheaters who exhibited 0 KBytes upload volume were 8% of the total population of free riders.

In our analysis we also refute what other studies had come across relating peers who exploit the system in BitTorrent to specific bandwidth capacities. We argue based on our finding that high bandwidth capacity peers exploit the system as well as low bandwidth capacity peers. Simply put, we do not relate the exploitation of resources in BitTorrent to bandwidth capacity. However, we relate this problem to the type of the peer, i.e., strategic or regular peer.

On the other phase of the analysis we used four types of BitTorrent clients to interact with other peers in the swarms: low bandwidth strategic clients LBS, low bandwidth non-strategic clients LBNS, high bandwidth strategic clients HBS, and high bandwidth non-strategic clients HBNS. Our results indicate that the Tit-for-Tat performs well in spreading cooperation amongst peers in BitTorrent regardless of their bandwidth capacity. It punishes uncooperative peers by choking them and rewards cooperative peers by reciprocating fair services with them. We also argue that the Tit-for-Tat reacts effectively against inter-class bandwidth capacity strategic peers. The mechanism prevents LBS peers from being served by high bandwidth capacity peers, and similarly, chokes HBS from being served by low bandwidth capacity peers. However, we emphasize our previous finding that the optimistic unchoke represents a critical shortcoming of the Tit-for-Tat mechanism, and thus BitTorrent. Furthermore, one of the surprising outcomes we found in this study is that our strategic clients in small swarms were able to finish their downloads faster than those in large swarms. We argued that this finding has to do with the competition the selfish or strategic peers face in heavily populated torrents which hinders their efforts in connecting to remote peers. Finally, based on our analysis on the optimistic unchoke, we proposed a modified approach to its role in BitTorrent. Our approach relies on requiring the peer to *remember* the identities of the remote peers who denied him the service after he optimistically unchoked them. The goal of this approach is to limit the number of times the local peer optimistically unchoke selfish peers in the near future. Our approach employs the exponential backoff policy in which the local peer retries his *punished* clients when needed and extends their stay in the punished list if they exhibit the same selfish behavior. Based on the preliminary results we obtained, this policy seems to drastically reduces the exploitation that is caused by the optimistic unchoke.

7 Future Work

We suggest implementing the memory backoff approach that we presented to the optimistic unchoking policy in the design of BiTorrent. Although we have shown that the memory backoff approach is a promising remedy to limit the deficiency of the optimistic unchoke and thus reduce free riding in BitTorrent, however, our work in that area needs further development. We encourage carrying on with the research and modifying BitTorrent with the new policy and testing the system. This could be achieved either on a PlanetLab platform or in a real network where all clients are confined to the new modifications.

References

1. BitTornado Official Website. http://www.bittornado.com/
2. BitTorrent Specifications Website. http://www.bittorrent.org/
3. The True Picture of P2P File Sharing. http://www.cachelogic.com/ (2004)
4. Adar, E., Huberman, B.: Free riding on Gnutella. Tech. rep. (2000)
5. Anagnostakis, K.G., Greenwald, M.B.: Exchange-based Incentive Mechanisms for P2P File Sharing. In: Proceedings of the 24th International Conference on Distributed Computing Systems, pp. 524–533. IEEE Computer Society, Washington, DC (2004)
6. Andrade, N., M.Mowbray, Lima, A., Wagner, G., Ripeanu, M.: Influences on Cooperation in BitTorrent Communities. In: Proceedings of the 2005 ACM SIGCOMM Workshop on Economics of P2P Systems, pp. 111–115. ACM, New York, NY (2005)
7. Axelrod, R.: The Evolution of Cooperation. Basic Books (1984)
8. Bharambe, A.R., Herley, C., Padmanabhan, V.N.: Analyzing and Improving BitTorrent Performance. Tech. Rep. MSR-TR-2005-03, Microsoft Research, Microsoft Corporation One Microsoft Way Redmond, WA 98052, USA (2005)
9. Chu, J., Labonte, K., Levine, B.N.: Availability and Locality Measurement of P2P File Systems. In: Proceedings of Scalability and Traffic Control in IP Networks. San Jose, CA (2002)
10. Cohen, B.: Incentives Build Robustness in BitTorrent. In: Proceedings of The 1st Workshop on Economics of P2P Systems. Berkeley, CA (2003)
11. Feldman, M., Chuang, J.: Overcoming Free-Riding Behavior in P2P Systems. In: ACM Sigecom Exchanges, vol. 5, pp. 41–50. ACM (2005)
12. Gummadi, K.P., Dunn, R.J., Saroiu, S., Gribble, S.D., Levy, H.M., Zahorjan, J.: Measurement, Modeling, and Analysis of a P2P File-Sharing Workload. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, pp. 314–329. ACM, New York, NY (2003)

13. Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., Zhang, X.: Measurements, Analysis, and Modeling of BitTorrent-like Systems. In: Proceedings of the Internet Measurement Conference, pp. 35–48. USENIX Association (2005)
14. Halesand, D., Patarin, S.: How to Cheat BitTorrent and Why Nobody Does. Tech. Rep. PUBLCS-2005012 (2005)
15. Jun, S., Ahamad, M.: Incentives in BitTorrent Induce Free Riding. In: Proceedings of the 2005 ACM SIGCOMM Workshop on Economics of P2P Systems, pp. 116–121. ACM, New York, NY (2005)
16. Legout, A., Urvoy-Keller, G., Michiardi, P.: Rarest First and Choke Algorithms Are Enough. Tech. Rep. inria-00001111, INRIA, Sophia Antipolis (2006)
17. Liogkas, N., Nelson, R., Kohler, E., Zhang, L.: Exploiting BitTorrent for Fun (But Not Profit). In: Proceedings of the 5th International Workshop on P2P Systems (2006)
18. Piatek, M., Isdal, T., Anderson, T., Krishnamurthy, A.: Do Incentives Build Robustness in BitTorrent? In: Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation, pp. 1–14 (2007)
19. Pouwelse, J.A., Garbacki, P., Epema, D., Sips, H.: A Measurement Study of the BitTorrent P2P File-Sharing System. Tech. Rep. PDS-2004-003 (2004)
20. Qiu, D., Shroff, N.B.: Modeling and Performance Analysis of BitTorrent-Like P2P Networks. In: Proceedings of the ACM SIGCOMM, pp. 367–378. ACM, New York, NY (2004)
21. Schlosser, M.T., Condie, T.E., Kamvar, S.D.: Simulating a P2P File-Sharing Network. In: Proceedings of The First Workshop on Semantics in P2P and Grid Computing (2003)
22. Zghaibeh, M., Anagnostakis, K.: On the Impact of Practical P2P Incentive Mechanisms on User Behavior. In: Proceedings of the Joint Workshop on The Economics of Networked Systems and Incentive-Based Computing (2007)
23. Zghaibeh, M., Harmantzis, F.: Revisiting Free Riding and the Tit-for-Tat in BitTorrent: A Measurement Study. Journal of Peer-to-Peer Networking and Applications, Published online: 17 July 2008, DOI 10.1007/s12083-008-0013-7
24. Zghaibeh, M., Harmantzis, F.: An “S-string” Scheme for Business-oriented P2P Networks. Journal of Electronic Commerce Research **7**(3–4), 381–398 (2007)
25. Zghaibeh, M., Harmantzis, F.: Lottery-based Pricing Scheme for P2P Networks. Telecommunication Systems Journal **37**(4), 217–230 (2008)

The Nature of Peer-to-Peer Traffic

João V. P. Gomes, Pedro R. M. Inácio, Mário M. Freire, Manuela Pereira, and Paulo P. Monteiro

Abstract The massive adoption of Peer-to-Peer (P2P) applications brings new challenges for network management. The increase of the bandwidth usage, the shift of the Internet habits of home users and the easiness to share and provide contents is changing the traffic characteristics, and breaking strong assumptions in which network design is based. As a response to concerns from network administrators and Internet Service Providers, traffic management is now a requirement instead of an option, making the traffic classification functionality into an essential administration tool. However, the evasive nature of most P2P applicationsrenders the classification

João V. P. Gomes
Nokia Siemens Networks Portugal, S. A., Rua Irmãos Siemens, 1, Alfragide, 2720-093 Amadora, Portugal; Institute of Telecommunications - Networks and Multimedia Group, Department of Computer Science, University of Beira Interior, Rua Marquês de Ávila e Bolama, 6201-001 Covilhã, Portugal, e-mail: jgomes@penhas.di.ubi.pt

Pedro R. M. Inácio
Nokia Siemens Networks Portugal, S. A., Rua Irmãos Siemens, 1, Alfragide, 2720-093 Amadora, Portugal; Institute of Telecommunications - Networks and Multimedia Group, Department of Computer Science, University of Beira Interior, Rua Marquês de Ávila e Bolama, 6201-001 Covilhã, Portugal, e-mail: pedro.inacio@penhas.di.ubi.pt

Mário M. Freire
Institute of Telecommunications - Networks and Multimedia Group, Department of Computer Science, University of Beira Interior, Rua Marquês de Ávila e Bolama, 6201-001 Covilhã, Portugal, e-mail: mario@di.ubi.pt

Manuela Pereira
Institute of Telecommunications - Networks and Multimedia Group, Department of Computer Science, University of Beira Interior, Rua Marquês de Ávila e Bolama, 6201-001 Covilhã, Portugal, e-mail: mpereira@di.ubi.pt

Paulo P. Monteiro
Nokia Siemens Networks Portugal, S. A., Rua Irmãos Siemens, 1, Alfragide, 2720-093 Amadora, Portugal; Institute of Telecommunications - Pólo de Aveiro, University of Aveiro, 3810-193 Aveiro, Portugal, e-mail: paulo.1.monteiro@nsn.com

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_44, © Springer Science+Business Media, LLC 2010

of their traffic a difficult task to achieve even through Deep Packet Inspections of the data units, mostly due to the application of encryption and randomization mechanisms. In order to meet the demand for new and accurate methods for traffic classification *in the dark*, it is necessary to study and fully understand the deep nature of P2P traffic. In this chapter, the characteristics of this type of traffic are discussed and explained, and recent studies about traffic characterisation are introduced and analysed from the perspective that may enable their application for traffic classification.

1 Introduction

With the recent evolution of Internet to a content distribution oriented architecture, computer communications have gradually migrated from the *client-server* paradigm to the *edge services* paradigm, and more recently to the Peer-to-Peer (P2P) computing paradigm. Applications like Napster and Gnutella opened the way for a renewed look into Internet connections, offering an easy and cheap way to deliver contents. If a few years ago, the traditional *client-server* model was responsible for the asymmetry embedded in Internet traffic, the emergence of the P2P paradigm and its adoption by a large number of applications have been influencing the Internet traffic behaviour towards a more balanced network, at least in terms of the amount of data transmitted in both directions. Although P2P applications started to be popular for providing file-sharing services, they have evolved to become nowadays responsible for the biggest portion of traffic in Local Area Networks (LANs). Voice over Internet Protocol (VoIP) and Instant Messaging (IM) applications have massively adopted the use of P2P architectures, leading to the growth of traffic load within the computer networks. Moreover, recent projects for the implementation of distributed Internet Protocol Television (IPTV) systems over P2P platforms reinforce the aforementioned tendency, and promise a non-negligible contribution for the network traffic increase as well.

Even though the popularity of P2P applications was mainly attained with the illegal exchange of resources under copyright protection, P2P platforms started to be used as an economic mean for delivering legal services and contents [11], due to the low distribution cost they offer. By using the resources of consumers to provide services to other consumers, providers avoid having to own too many expensive central servers and deploying novel high rate connections. BBC iPlayer [1], Joost [3] or Miro [4] are examples of recent web services where P2P platforms are used as a cheap solution to deliver video contents. Although the majority of video sharing services in the web are still based on centralised servers (for example, *YouTube* [9] or *Daily Motion* [2]), distributed systems based on the P2P architecture are gathering more interest. As a consequence, traffic load is increasing dramatically in all segments of computer networks [32], causing the retreat of the bottleneck from LANs to access or metropolitan networks [25]. If P2P traffic is not ruled and shaped, the capacity to guarantee the Quality of Service (QoS) required for multimedia or critical applications, as well as to fairly distribute bandwidth to all of the Internet users, is severely reduced. Henceforth, several Internet Service Providers (ISPs) began to limit, or even block P2P traffic, especially when generated by applications that offer file-sharing services. What once was an exclusive concern of LANs administrators, is today considered by some ISPs as an emerging problem for which their networks are not prepared for, as they were designed under the assumptions of traffic asymmetry and random on / off activity periods [31]. Furthermore, the flexibility of P2P applications brought new concerns for network administrators. The easiness to fetch illegal contents inside organisations, the lack of security, the cases where the communications with external hosts are not allowed, the risk of sharing confidential data or the excessive resources consumption constitute only a few examples of such concerns.

Unfortunately, shaping P2P traffic is not a trivial task [28]. The most used and accurate traffic classification methods are based in Deep Packet Inspection (DPI) techniques, which identify network protocols and applications recurring to the recognition of well known strings inside the packets payload. Such mechanisms present a considerable trade-off between efficiency and complexity and limitations related with performance and privacy laws. Additionally, as they are based in payload information, their effectiveness and scope are limited by the available data within the packet. The recent versions of applications like *BitTorrent* or *eMule* enable the users to encrypt the packets or obfuscate the protocol, as a response to the DPI based tools used by ISPs to block their traffic. Another well known case is *Skype*, whose traffic is always encrypted by a non disclosed algorithm, not only for security purposes, but also because of its evasive secondary effect.

The interest in new methods for classification in the dark, based in behavioural patterns rather than in the packets contents, is thus easily justified. To accomplish such purpose, it is mandatory to study the characteristics of P2P traffic and its behaviour.

Recent studies show that traffic resulting from the use of different P2P applications presents several behavioural patterns related with its nature [19]. These characteristics reflect the heterogeneous character of this kind of traffic. Mathematical measures, as the entropy, may then potentially be used to identify traffic from P2P applications. Therefore, traffic behaviour can be considered for the basis of new tools for traffic classification. If the behavioural patterns used by classification in the dark tools draw on the very nature of P2P traffic, they are not so easily disguised or obfuscated, at least not without prejudicing the P2P application performance per se.

This chapter discusses the impact of P2P traffic in computer networks. Its characteristics are presented and explained, with emphasis in the ones that are consequence of its nature. Entropy is suggested as a way to measure traffic heterogeneity, allowing this property to be used for P2P traffic characterisation. The subject of traffic analysis and classification is herein observed from an *inside of the network* perspective also, so as to address several issues concerning the integration of P2P traffic monitoring systems in current networks.

2 The Importance of Traffic Characterisation

Internet keeps evolving, offering users new tools and services. Telematic applications like the ones built on top of the technologies commonly known by *Web 2.0* or of P2P systems emerge every day, pushing the services to the edges of the networks. This fact increases the importance of the role played by regular users, as it allows them to provide contents, instead of just request them.

Nevertheless, it is also affecting the way users interact with each other through networks. Consequently, traffic within the computer networks is changing, presenting nowadays different properties. It is logical to think that P2P traffic, which is typically heavy in terms of the amount of information transmitted, affect the remaining flows of information travelling through routers and switches, producing a rather negative impact in the network availability and performance. Moreover, recent applications and services often raise new and problematic security issues. Such fact is especially critical when the communication between users is narrower than ever, turning easier the dissemination of virus, worms or other security threats. Consequently, traffic monitoring mechanisms play a vital role in keeping the network operational. Yet, the deep inspection of the traffic is not always an option as it is an extremely demanding task in terms of computation [37]. As so, it is necessary to use behavioural techniques for traffic monitoring and classification, for which traffic characterisation is an all-important instrument since it helps to understand the generic characteristics of traffic from each class and its typical behaviour.

When applied to classification, innovative techniques to characterise the traffic, can lead to the development of new business models, besides assuring the QoS needed for critical and real-time applications and helping to implement security measures to protect the network users. Accurate mechanisms for traffic classification will allow ISPs to charge for specific services, e.g., VoIP communications, IPTV or torrent downloads. Instead of simply charging for the Internet access and offering all services within a single package at a unique price, ISPs will be able to provide a more granular panoply of service level agreements, on an on-demand basis [14].

2.1 The Effect of Peer-to-Peer Traffic in Computer Networks

Although the original and naive conception of the Internet laid on a distributed network paradigm, in which every node should had the ability to perform both roles of client and server, its architecture evolved almost completely to the one of centralised systems. For such reason, Internet traffic lost its symmetry and became characterised by a large difference in the amount of data sent in upstream and downstream: in upstream, traffic consists mainly of small requests; while in downstream, the transmitted data is mostly related with the contents provided by a server. For several years, world wide web and e-mail, which are based in the client-server paradigm, were the *most used* services of the Internet, motivating the asymmetry trend of its traffic. Naturally, computer networks were gradually redesigned while taking into account the assumption that the amount of data sent in upstream would be considerable smaller than the one transmitted in downstream.

The unfolding of the P2P paradigm, and the appearance of several tools based on it, changed the way Internet is used, inspiring a return to its origins. Users are now able to easily communicate with each others directly, offering services and requesting them. However, the changes brought by the new architecture influenced networks in different levels. (Notice that the terms *upstream* and *downstream* will be used, in this chapter, to refer to the data transmission from the client to the server and vice-versa, respectively.)

2.1.1 Consumption of Resources

The proximity between users of P2P systems was primarily used to directly exchange contents (data) between themselves, giving rise to several file-sharing systems, which rapidly became standard for sharing large multimedia contents. The simplicity in accessing and searching for contents increased largely the amount of information shared and transmitted between users.

At the same time, several communication tools for VoIP, IM or video calls were developed based in P2P platforms. These applications, especially the ones used for VoIP and video calls, are eager resource consumers, not only because of the large amounts of data they generate, but also because such traffic needs to be processed quickly, as it results from real-time applications.

Although at the beginning P2P platforms were mainly used for the illegal sharing of multimedia contents, recent Internet services are using P2P based tools as cheap means for providing their contents. Such fact is especially actual for video distribution. IPTV systems like Joost [3], BBC iPlayer [1] or Miro [4] take advantage of the P2P architecture to provide video over Internet Protocol (IP), without the need to have powerful centralised servers or high bandwidth Internet connections. Indeed, P2P platforms allow providers to use the resources of their clients to distribute contents to others alike. In opposition to what happens in the client-server architecture, where each additional client represents an added cost to the provider, in P2P systems each new client increases the capacity and the available resources of the network. Consequently, the dual role played by the nodes in a P2P system consumes an extremely large portion of bandwidth, when compared with the usual clients in client-server architectures. Moreover, the real-time nature of most P2P applications requires more network resources than other widely used Internet services, like web or e-mail. In addition, several studies show that P2P applications are nowadays responsible for the biggest portion of the traffic in LANs [18, 22, 27, 34]. As a consequence, when used in an abusive way, P2P software generates large amounts of traffic which can influence negatively the traffic of the remaining applications, affecting, in this way, the QoS required by other real-time or critical services running over the network. This problem is decisively contributing for the unpopularity of P2P traffic among many ISPs and network administrators.

Sharing distribution costs between clients has obvious advantages for content providers. However, that cost is supported by the ISP of each user. When the contents are massively distributed to several users, the impact in ISPs networks can be highly significant. A meaningful example of this problem is the BBC iPlayer, which is being largely used, threatening the networks availability. As a consequence, ISPs are suggesting that BBC and other content providers should share the costs of increasing the overall Internet capacity [32].

2.1.2 Traffic Asymmetry

Although most of the reasons why ISPs fear P2P applications are related with the amount of traffic they generate, some of them stem from its symmetric nature, which probably represents their biggest challenge. Actually, one of the most direct effects of P2P applications in the traffic of computer networks is the destruction of the traffic asymmetry assumption inherited from the client-server models. In the communications used by typical Internet services like web or e-mail, the traffic generated in upstream is considerable lower than the one transmitted in downstream, creating the assumption that the traffic in computer networks is asymmetric. Networks were thus designed taking that fact into consideration and support nowadays higher debit rates in downstream than in upstream. Nevertheless, P2P applications increased the traffic generated in upstream by the network nodes. As so, administrators and ISPs face nowadays a problem for which their networks are not prepared. The referred problem gains special emphasis in access networks, for which the available bandwidth in upstream is typically not enough to handle the traffic generated by the growing number of P2P applications.

2.1.3 Peer-to-Peer Inside Organisations

Many P2P tools employ *evasive techniques* to prevent the identification of their traffic (see Section 2.2). Although many companies implement several restrictions regarding the usage of P2P applications by their collaborators, they are unable to actually prohibit their utilization and block their traffic. As a result, several problems threat the health of the internal network.

Similarly to what happens in other contexts (see Section 2.1.1), the excessive consumption of resources can affect network availability, preventing the normal operation of legitimate applications inside a given organisation. However, there are a few threats that are specifically problematic for an organisation. The easiness P2P applications have to establish connections to other nodes, even when they are behind a gateway or a firewall, can open a security breach and render the network vulnerable to virus, worms and other security threats; it may also ease the exposure of confidential data to external entities [21]. Moreover, the existence of contents protected by copyright laws (which are commonly shared using P2P applications) in machines within the network of the organisation may raise legal issues.

In some situations, VoIP applications may also raise additional problems for organisations, for it is not yet clear if VoIP applications should be seen and paid as telephony services, comparable to the usual telecommunication companies services [17, 26]. Thus, it is likely to think that common contracts between organisations and telecommunication companies with a clause for exclusivity may be compromised by the usage of VoIP applications.

2.1.4 Network Security

The *virtual proximity* that P2P systems offer to users, which basically allows simple and fast exchange of contents, brought also new security challenges for computer networks. The dissemination of virus, worms and other threats is easier than ever. And, more importantly, as it was described in a previous section, many P2P applications are used inside private networks, opening a hole in the internal security. In order to be seen as feasible solutions for content distribution, applications based in P2P architectures should consider all the security issues they may lead to, and be able to overcome the problems they can raise.

As any tool that allows data exchanging (e.g., e-mail), P2P systems may be the vehicles for the exploitation of several security flaws, even if inadvertently. However, in the case of P2P applications, this fact is specially problematic. Since the traffic they generate is now intentionally *disguised* to avoid monitoring systems, the shared contents are even harder to control, effectively decreasing the security level of a terminal machine or even of the entire network. Furthermore, the data received by a machine remains accessible to other users, easing diffusion of infections. In fact, some studies show that P2P users usually let the client software running even when they are not sit in front of the computer. During that period, the activities of the P2P client are not being directly monitored by the user, increasing the risk of abuse [21].

Skype is usually considered a good case study for security issues related with P2P applications. Its traffic is encrypted using a closed and proprietary algorithm, which turns its analysis and detection harder to perform. Moreover, Skype implements techniques to avoid detection mechanisms and Network Address Translation (NAT) issues, or even to establish connections behind a firewall. Consequently, and since most of the applications like Skype also allow file transferring between users, such data will not be analysed by any security or monitoring mechanism, neither blocked by a firewall. As so, they can open a hole in the network, through which it may be easier to pass malicious data into the system.

VoIP and IM applications are also threaten by the typical security challenges most communication tools face: authentication, non-repudiation, confidentiality, communication security, integrity, availability, privacy or anonymity [26]. Additionally, P2P architectures often use the concept of *supernode* to surpass NAT firewalls (which basically means that the traffic between two nodes is be forward through an intermediate note), being thus unclear if the referred tools are protected against *man-in-the-middle* attacks, in most of the cases.

2.2 *Making Peer-to-Peer a Practicable Solution: Traffic Classification*

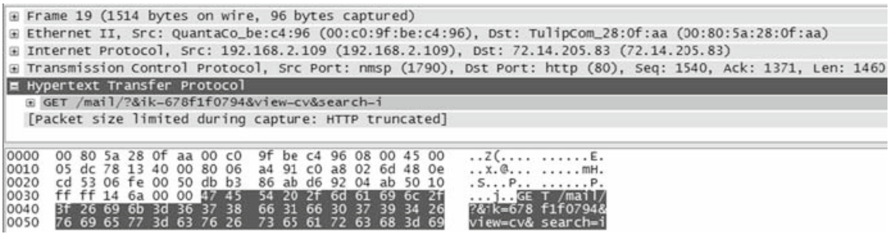
Even though P2P applications may negatively impact computer networks (as explained in the previous section), they are also seen as an effective solution for distributed content delivering [11]. To be able to sustain the impact of P2P traffic, networks should be strictly managed, which involves shaping and prioritising the entire set of flows so as to distribute the available bandwidth fairly, while having into account the needs of each traffic class. Achieving such purpose requires accurate classification mechanisms, that can identify what application or traffic class a certain packet or flow belongs to.

The traffic classification used to be made by recurring to the port numbers of the transport layer protocol of a given connection. Typically, each protocol was definitely associated with a given port number (e.g., ports 80 and 25 were normally used by Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP) traffic, respectively). However, these simple classification mechanisms were easily bypassed using random port numbers, or even port numbers commonly used by *other* well known protocols. Notwithstanding the fact that some applications (e.g., MSN messenger) do not intend to hide their traffic, turning simple its detection and classification, most P2P applications integrate evasive techniques to avoid detection mechanisms and disguise their behaviour. Some of them implement measures to obfuscate their protocol as a response to the prohibitive rules some ISPs settled to block protocols like BitTorrent or eMule. Hence, several classification methods to detect traffic from P2P protocols have been developed, elaborating on the most diverse approaches. The techniques used by these classification systems are usually divided into two different groups, which will be described in the subsequent sections: DPI [14, 22, 29, 33] and classification in dark [10, 13, 16, 23, 24, 30].

2.2.1 Deep Packet Inspection Methods

After the adoption of random port numbers in the P2P connections became a common feature, the most obvious manner to understand what type of data a certain packet carries was to look deep into the packet payload and try to recognise patterns that are common to the traffic of a certain protocol. Based on this idea, DPI methods try to find data signatures within the packet payload and compare it against a database containing well known signatures from several protocols.

Figures 1 and 2 present examples of data signatures used to identify HTTP and SMTP traffic, respectively. A screenshot of the popular network protocol analyser tool Wireshark [8] is shown in Fig. 1, depicting the deep analysis of one packet classified as HTTP traffic. It is possible to observe that different protocols from layer 2 to layer 7 were identified by the tool: Ethernet, IP, Transmission Control Protocol (TCP) and HTTP. In the bottom of the image, one may find the



```
⊞ Frame 19 (1514 bytes on wire, 96 bytes captured)
⊞ Ethernet II, Src: QuantaCo_be:c4:96 (00:c0:9f:be:c4:96), Dst: TulipCom_28:0f:aa (00:80:5a:28:0f:aa)
⊞ Internet Protocol, Src: 192.168.2.109 (192.168.2.109), Dst: 72.14.205.83 (72.14.205.83)
⊞ Transmission Control Protocol, Src Port: nmsp (1790), Dst Port: http (80), Seq: 1540, Ack: 1371, Len: 1460
⊟ Hypertext Transfer Protocol
  ⊞ GET /mail/?&ik-678f1f0794&view-cv&search-i
   [Packet size limited during capture: HTTP truncated]

0000  00 80 5a 28 0f aa 00 c0  9f be c4 96 08 00 45 00   ..Z(.... ......E.
0010  05 dc 78 13 40 00 80 06  a4 91 c0 a8 02 6d 48 0e   ..x.@... .....mH.
0020  cd 53 06 fe 00 50 db b3  86 ab d6 92 04 ab 50 10   .S...P.. ......P.
0030  ff ff 14 6a 00 00 47 45  54 20 2f 6d 61 69 6c 2f   ...j..GE T /mail/
0040  3f 26 69 6b 3d 36 37 38  66 31 66 30 37 39 34 26   ?&ik=678 f1f0794&
0050  76 69 65 77 3d 63 76 26  73 65 61 72 63 68 3d 69   view=cv& search=i
```

Fig. 1 Example of a signature used by Wireshark to identify an HTTP packet

```
<protocol name="smtp" longname="SMTP (Simple Mail Transfer
Protocol)" showsumtemplate="smtp">
  <execute-code>
    <verify>
      <if expr="hasstring($packet[$currentoffset:0], ’^220[\x09-\x0d -~]*
      (e?smtp|simple mail)|^(HELO|EHLO) .*\x0d\x0a’, 0)">
        <if-true>
          <assign-variable name="$protoverify_result" value="%FOUND"/>
        </if-true>
      </if>
    </verify>
```

Fig. 2 Part of the rule included the XML file containing the NetPDL database used to identify SMTP traffic recurring to DPI methods

hexadecimal representation of the first 96 bytes of the packet. The bytes highlighted are the ones used as the packet identification, which ultimately points it as belonging to HTTP traffic. In Fig. 2, it is represented part of the rule used to identify SMTP traffic, which belongs to an eXtensible Markup Language (XML) file containing the NetPDL database used by the NetBee library [7]. This rule is written using the NetPDL [5] language. In some solutions, Intrusion Prevention and Detection Systems (e.g., Snort [6]) are also used as a platform to implement the rules and deeply inspect the traffic.

DPI methods are known by their accuracy, as they are based in the data carried by the packets. Their precision is motivating the development of new and more sophisticated DPI mechanisms which open the way for the new business opportunities [14]. Nevertheless, there are also a few important drawbacks associated with these kind of techniques. Since the classification is achieved recurring to the data encapsulated in the deepest part of a packet contents and to a potentially large database of signatures, they are useless in the cases where the payload is encrypted (e.g., Skype) or when no suitable string pattern has been devised yet. Moreover, string matching is done by *sliding* each of the signatures through the entire payload, as some of them may appear anywhere within it. Therefore, DPI methods are not able to process huge amounts of traffic in high-speed networks [37], at least not without recurring to expensive equipments for parallel processing.

2.2.2 Classification in the Dark

Since looking deep into the contents of the packets is not always a feasible option, new methods for traffic classification have been developed, using behavioural characteristics of the traffic. These kind of approach is known as *classification in the dark* as it does not access the data within the packets. Most classification in the dark mechanisms use generic characteristics like the number of active connections, the inter arrival times between packets, the number of different hosts to which a node is connected, the average value of the packet sizes, the bandwidth usage ratio between upstream and downstream traffic, etc. Typically, the existent methods correlate several combinations of these properties using different tools or measures. Most of them implement several heuristics that are tested sequentially. As a packet or a flow matches (or not) each one of the heuristics, so it is classified as belonging (or not) to a certain protocol. In the literature, it is also possible to find mechanisms that recur to Naïve Bayes and clustering techniques to associate a set of traffic properties to an application or protocol.

Additionally, active methods are also described as an alternative for traffic classification without deep inspection of the packets. Such mechanisms are especially suitable to identify hosts using a specific application or, e.g., users that are probably requesting or sharing illegal contents. The idea behind this approach is to run a fake instance of the application one wants to detect and identify which hosts try to connect or request services from it. By analysing and gathering the information about the identified peers, it is possible to reconstruct the underlying P2P topology and, indirectly, classify the traffic exchanged between those hosts as belonging to the same protocol of the fake instance.

Classification in dark mechanisms are not so accurate as DPI methods because they do not known, directly, what data is inside the packets. If well implemented and supported in a reasonable assumption, it is only possible to make a (strong) suspicion about their contents. Nevertheless, the behaviour-based methodologies can be extremely useful when the packet payload is encrypted, or when the deep inspection of all the packets represents a rather expensive task, in terms of computational requirements. Both approaches should be implemented cooperatively in order to perform an accurate and efficient traffic analysis.

3 Peer-to-Peer Traffic Characterisation

The P2P architecture provides important features for service and content delivery that can help to reduce its inherent costs and improve the performance of the networking systems. Although ISPs fear it and try to avoid it, the usage of P2P applications have been increasing each day. Therefore, it is important to design and prepare computer networks to support and minimise the impact of P2P traffic. To do so, it is necessary to understand its specific properties and its behaviour. Such knowledge helps developing accurate methods for traffic classification, which are used to control the network traffic and implement a more fair and efficient bandwidth sharing policy. Moreover, basing the classification mechanisms in the deep nature of P2P traffic will turn them more difficult to be avoided.

In the following sections, several behavioural characteristics, observed in the traffic generated by P2P applications, are discussed. Most of them are consequence of the P2P nature, others result from properties that are typical of particular P2P tools.

3.1 Generic Properties of Traffic from Peer-to-Peer Protocols

The P2P paradigm is mainly singled by the dual role of the network nodes, acting both as client and server. This singularity and the way it is used by P2P tools are responsible for several specific details reflected in the traffic behaviour. Actually, when observing P2P traffic, one may notice the presence of nodes acting, at the same time, as servers and as clients [13, 24]. The simplest way to observe such behaviour is by identifying, for each active connection, which node has started it (the one requesting the service) and which node has acknowledged it (the one providing the service). A more efficient approach would be to analyse which port numbers are being used by the active connections. One should expect that a node acting as a server has several connections using the same port number, as it is likely providing the same service to several clients. Using the same principle, a node acting as a client should have several connections established using different port numbers, as it is requesting services from various servers.

Several studies [13, 23, 24, 30] also showed that, frequently, P2P applications use parallel connections for the same source-destination IP addresses pair, using both TCP and User Datagram Protocol (UDP) protocols. UDP is a low-overhead transport layer protocol, useful for spreading queries and status. TCP, by its turn, due to its reliability, is used for content delivering.

As it was previously explained, P2P architecture brings users *closer* to each other, being each of them able to provide and request services to and from the others. Consequently, traffic generated by nodes running P2P applications is mostly dominated by the existence of several active connections, most of the times, established with distinct IP addresses. Moreover, they often create some sort of communities whose members run the same P2P application, establishing several connections between themselves [24]. The concept of network diameter appears in a few studies, which conclude that connections established between peers form a network whose diameter is larger when its nodes are running P2P applications [13]. The network diameter is, in this context, defined as the maximum of the shortest distances (number of hops) between all the nodes in the network.

The traffic from a node using P2P tools also presents specific properties regarding the active connections. Usually, the number of distinct IP addresses to which the node is connected to is equal to the number of distinct port numbers used for the connections [22–24]. In some P2P networks, a distinct port number is used for each connection established with other peer.

The real-time prerequisites of many P2P applications is also noticed in the traffic. Although it is not an exclusive property of this class, P2P traffic usually presents a high percentage of TCP packets marked with the *PUSH* tag.

In [24], the authors present non-payload or failed flows as a common (and side) effect of P2P applications. They state that clients usually try to connect to other nodes that have already disconnected from the network. Additionally, a few P2P applications encrypt the packet payload. Although it is not a truly behavioural pattern, such fact also produce an effect in the traffic. Consequence of the encryption function, packet payload is extremely heterogeneous, which can be used as a pattern for some P2P protocols [12, 15].

Many P2P protocols use HTTP connections during their operations. Although such traffic is, for all it counts, HTTP traffic and therefore classified as so, its true nature is the one of P2P traffic, thus presenting properties related with this paradigm. The typical difference between HTTP traffic resulting from web activities and from P2P applications is that web servers use several concurrent connections in order to transfer different web pages elements (text, images, sounds, ect.), while content distribution between peers consists mainly on consecutive connections [30].

3.2 Analysing Peer-to-Peer Traffic Behaviour

The traffic characteristics described in the previous section are present in traffic from most P2P protocols. Yet, some of those properties, like the concurrent use of UDP and TCP, can be changed by the applications with (almost) no performance lost. It is likely that some of them will be changed with the evolution of the protocols and with the adoption of P2P platforms for new services.

In [19, 20], we have analysed traffic from different P2P and non-P2P applications. Since the main purpose of the work was to identify the most deep nature of P2P behaviour, the traffic was captured in the source at the initial phase of the project, near the application user. They analysed traffic from the following applications or services: web browsing, HTTP download, live streaming, streaming download, Secure File Transfer Protocol (sFTP), e-mail, eMule, BitTorrent, Direct Connect ++ (DC++), MSN, Skype and Google Talk. The expression *HTTP download* will be use to refer to the long download of a file, using the HTTP protocol.

Normally, P2P applications are not used to establish one connection and exchange one file with a single peer only. The multiple connections between peers and variety of contents and services they exchange are patterns that influence the traffic. Moreover, as consequence of their viral nature, most P2P services, e.g., file-sharing systems, need to perform several control operations (e.g., search for contents, spreading lists of available contents, requests, responses) which are responsible for specific characteristics of the traffic. Moreover, some of the applications based on P2P architecture have a strong real-time nature and their traffic is, most of the times, considered as urgent. For instance, packets from VoIP traffic must be processed and sent almost in the same moment they are generated, for the sake of the quality of the voice service.

In the study describe in [19, 20], authors noticed that the aforementioned properties influence the packet sizes of the traffic generated by P2P applications. Some studies have already considered the packet sizes, using their distribution to build a traffic model or to define a range of values for a specific protocol. Nonetheless, in this case, the authors state that it is the heterogeneity of the packet sizes values that can reveal the use of P2P traffic, not its approximate distribution. In this context, the size of one unique packet, individually, is completely useless, independently if it is large or small. Figure 3 depicts the packet sizes for traffic generated by four different applications: one *traditional* non-P2P service (HTTP download); one file-sharing application (DC++); and two common applications used for a VoIP call (i.e., MSN and Skype).

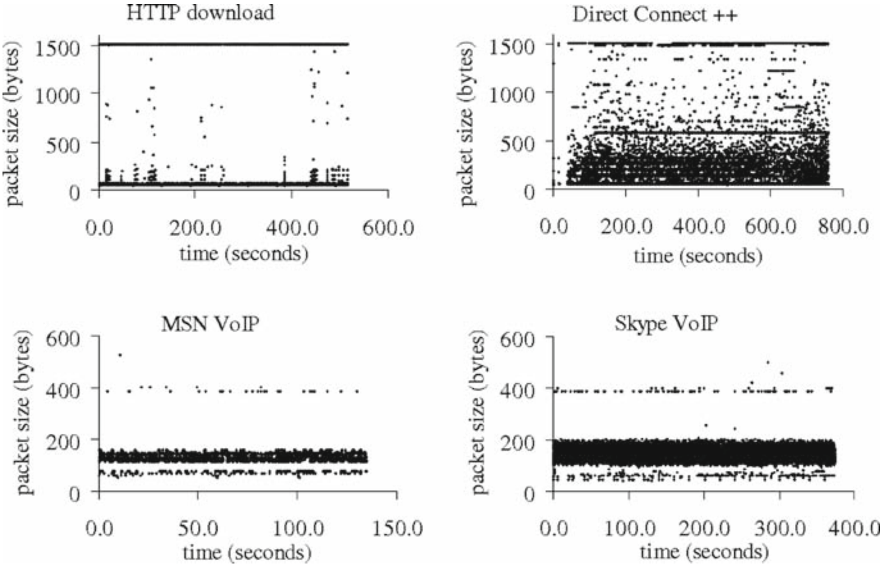


Fig. 3 Representation of the evolution of the packet sizes, in a size versus time chart, for traces containing traffic from HTTP download, DC++, MSN VoIP and Skype VoIP

As one can observe, the traffic generated by the HTTP download is, in terms of packet sizes, extremely homogeneous, being almost completely characterised by two types of packets: the small sized packets concerning acknowledge messages; and the large sized packets, limited by the maximum size allowed by the lower protocols, which carry the data belonging to the file being downloaded. In the case of DC++, the set of packet sizes is more heterogeneous, being constituted by packets with several sizes comprehended between the maximum size and the size of the fixed minimum value for that aspect. This evidence is due to the diversity of control operations and to the *multiple* concurrent connections, which may be using distinct routes with different packet size limits (i.e., different path Maximum Transmission Unit). VoIP traffic is constituted by smaller packets but the distribution of their sizes is extremely heterogeneous. This heterogeneous property is easily understood if one takes into consideration it results from voice traffic, whose packets have to be created and immediately sent. It is not possible to wait for the packet to have the largest size allowed by the lower protocols. Variable bit rate codecs shape the human voice into variable size packets.

The cases presented herein are illustrative examples of the conclusions the authors have reached. The same analysis was made for all the traffic traces considered and the results were consistent with such conclusions: traffic from P2P applications is formed by packets whose sizes belong to a more heterogeneous set when compared with the traffic from non-P2P applications. The packet size heterogeneity is a pattern that results from the most deep nature of P2P traffic. In fact, such behaviour is not easy to be modified without disturbing the normal functioning of the underlying application and, especially, without decreasing its performance and QoS.

3.3 Entropy as a Measure of Heterogeneity

Before proceeding, it is of the opinion of the authors that it is important to explain the concept of Entropy and of its interpretation in the scope of this chapter. Although Entropy may be interpreted from different perspectives, in the scope of this work, it is the definition of the information theory [35], which describes it as a measure of the uncertainty of a given random variate, that will be used. Herein, its value is seen as possible concretisation of the level of heterogeneity of the variable under analysis. It is important to retain that, the entropy value increases as the domain of occurrences of the variable (to which the referred statistic is applied to) gets more dispersed. The formal description of Entropy is given by expression (1) and denoted by $H(n)$, where n represents the number of values in the observation pool, and $p(x_i)$ denotes the probability of occurrence of a given value x_i. It should be noticed that the maximum value the entropy can attain is given by $H(x) = ln(n)$.

$$H(X) = -\sum_{i=1}^{n} p(x_i)\ln(p(x_i)) \tag{1}$$

3.4 Capturing the Nature of Peer-to-Peer Traffic

Saying that P2P traffic is heterogeneous, in what concerns the packet sizes, can be excessively vague, or even meaningless, to be useful for network design or management. It is thus necessary to define a method for quantifying such level of heterogeneity. From now on in this chapter, the terms *heterogeneous* and *homogeneous*, when applied to computer networks traffic, will be used to refer the heterogeneity or homogeneity of the packet sizes.

The authors suggest the use of entropy as a way to measure the level of heterogeneity of the traffic [19, 20]. The entropy value can then be used for P2P traffic characterisation and network management. Nonetheless, measuring the entropy for an increasing or an extremely large set of packets tends to produce a smooth result, and not reflecting the small changes occurred along a given trace. The responsible for this problem is the *Law of Large Numbers*. Calculation of the entropy in a real-time and point-by-point basis is computationally expensive ($O(n^2)$ algorithm) and produces an evolution curve that converges to an expected value, as the number of samples analysed increases, gradually losing the sensitivity to small changes. Therefore, the entropy was calculated for a small set with a fixed number of packets. By implementing a sliding window mechanism that processes the traffic trace sequentially and in a packet-by-packet manner, it is possible to obtain

Table 1 Average value of the entropy calculated for each one of traces, using a window size of 100 values and separating the traffic by the direction of the flows.

Downstream		**Upstream**	
Traffic	**Entropy average**	**Traffic**	**Entropy average**
Skype VoIP 1	3.729	Skype VoIP 2	3.890
Skype VoIP 2	3.698	Skype VoIP 1	3.762
MSN VoIP 1	3.260	Google Talk VoIP	3.725
Google Talk VoIP	2.855	MSN VoIP 1	3.238
eMule download 2	2.498	Skype IM 2	2.667
DC++ download/upload	2.377	Skype IM 1	2.451
BitTorrent	2.273	MSN IM 1	2.261
Skype IM 2	2.153	MSN IM 2	2.221
eMule download 1	2.141	eMule upload 2	2.111
MSN IM 1	1.959	eMule upload 1	2.023
Skype IM 1	1.886	Google Talk IM	1.980
eMule upload 1	1.843	eMule download 2	1.889
Google Talk IM	1.810	MSN VoIP 2	1.854
MSN VoIP 2	1.740	sFTP download	1.484
MSN IM 2	1.612	BitTorrent	1.459
web browsing	1.427	eMule download 1	1.410
eMule upload 2	1.334	DC++ download/upload	1.369
Live Streaming 1	1.278	web browsing	1.287
sFTP download	1.004	Live Streaming 2	1.243
Live Streaming 3	0.772	Live Streaming 1	1.143
Live Streaming 2	0.639	sFTP upload	1.002
sFTP upload	0.552	HTTP download 2	0.752
HTTP download 4	0.352	HTTP download 4	0.578
Streaming download	0.282	Mail upload	0.511
HTTP download 3	0.175	HTTP download 3	0.252
HTTP download 2	0.073	Live Streaming 3	0.218
Mail download	0.050	Mail download	0.035
Mail upload	0.040	HTTP download 1	0.014
HTTP download 1	0.014	Streaming download	0.007

one entropy value for each window step in an efficient manner. In each step, the oldest value in the window is pushed out; while the new one is inserted into the window.

By applying this method to the traces containing the traffic from different classes, it is possible to assess the heterogeneous (or homogeneous nature) of each one of them [19, 20]. The authors noticed that the traffic from P2P traffic (as it is more heterogeneous) presents higher entropy value when compared with the traffic generated by the non-P2P applications. Consider observing Fig. 4 to obtain a graphical example of these words, and notice the evolution curve of the entropy value for three illustrative applications: a non-P2P service (streaming download), a P2P file-sharing tool (DC++) and a VoIP call using P2P system (MSN). The non-P2P application has an entropy value extremely close to zero; in the case of the VoIP application, the entropy values are very high in almost all the represented window steps; while in the case of the file-sharing application, the entropy is high, though not so high as for the VoIP traffic. Identical results were obtained for traffic from other applications belonging to the same classes.

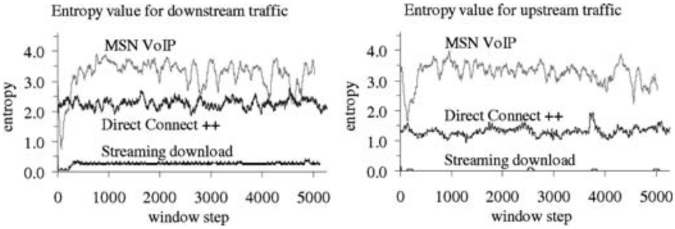


Fig. 4 Comparison of the entropy value, calculated for a window of 100 packets, between traffic from streaming download, DC++ and MSN VoIP.

In order to provide this explanation with a generic perspective of the entropy levels reached by each traffic class, the authors calculated the average of all the intermediate entropy values for each traffic trace and organised such information in Table 1 [19, 20]. It is possible to see that applications implemented over P2P platforms (the cells are highlighted in the table), are almost perfectly organised in the top of the table, as they synthesise the most heterogeneous traffic. It is interesting to notice that the VoIP applications appear in the first rows of the table, showing the highly entropic (regarding the packet sizes) patterns by which this type of applications is characterised. Such remark is especially valid for the traces containing Skype traffic, which present the highest value for the average.

The authors have also noticed that even when P2P traffic is mixed with traffic from non-P2P applications, the level of heterogeneity of the size of its packets if kept (though weaker), holding the entropy value sufficiently high to reveal the presence of P2P traffic. In Fig. 5, it is possible to observe that the presence of P2P traffic increases the overall entropy value. Even when a high rate HTTP download is injected in the trace, entropy remains sufficiently high to suspect of the presence of

P2P traffic. In the moment when all P2P applications stop running, the entropy value decreases to near zero values. Analyses performed using different combinations of traffic classes revealed precisely the same behaviour [19, 20]. The usage of P2P applications raises the overall entropy value of the respective traffic. When non-P2P applications are run during simulation time, it is still possible to notice the presence of P2P traffic, even when their traffic dominates the bandwidth. When only non-P2P are being used, the entropy value decrease to near zero.

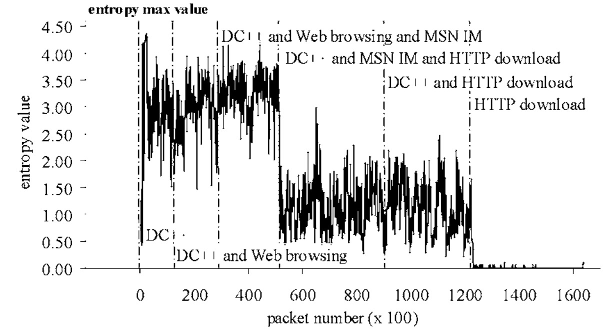


Fig. 5 Evolution of the entropy value for traffic from several applications, calculated using a window size of 100 packets

4 Inside a Network: Using Behavioural Characteristics to Identify Peer-to-Peer Traffic

The importance that the responsible usage of P2P applications (and their traffic control) has for the health of computer networks, motivates the analysis of different behavioural properties of P2P flows, and the development of methods for its classification. Nevertheless, behaviour based mechanisms should not be seen as an alternative to DPI methods. The latter are characterised by the superior accuracy of their results, when compared to the methods based in traffic behaviour. As it was further explained in Section 2.2, each of both approaches has advantages in specific contexts. In order to improve the effectiveness of the whole monitoring device of a network, the cooperation between DPI mechanisms and methods for classification in the dark should be maximised by identifying in which cases each of them should be used.

The implementation of the monitoring system should, for that reason, consider a few important issues regarding the use of DPI and behaviour based methods. Depending on the network context, it is important to discuss if both mechanisms should operate concurrently or sequentially and, in the latter case, if the DPI methods should be used before or after the classification in the dark mechanisms.

In a few cases, like the ones were the packet payload is encrypted, or the traffic is generated by a new or unknown P2P protocol, for which no data signature is known yet, DPI methods are completely useless. In such context, a strong suspicion about the nature of the traffic can be extremely helpful for network management. Therefore, the behavioural analysis may prove itself useful when the DPI classification is not returning a satisfiable result (such case is represented in Fig. 6).

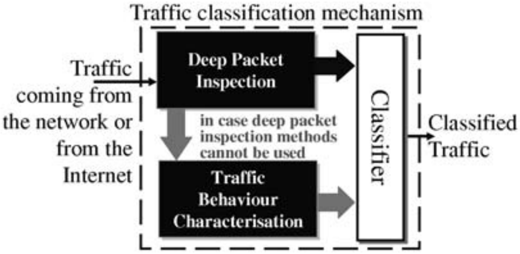

Traffic classification mechanism
Traffic coming from the network or from the Internet
Deep Packet Inspection
in case deep packet inspection methods cannot be used
Traffic Behaviour Characterisation
Classifier
Classified Traffic

Fig. 6 Logical scheme of a traffic classifier where behavioural methods are used only in the cases where the DPI mechanism is unable to identify the traffic

Another problem associated with the DPI techniques is related with the strict trade-off between efficiency and computational power, which ultimately results in a weak scalability factor, and the consequent impossibility to inspect traffic in high rate links, as it requires the comparison of the packet payload against a large signatures database. Furthermore, in some cases, it is not possible to know in what part of the payload such patterns appear a priori, nor if it is required to first reconstruct the fragmented application level flow to find them. In this context, a suspicion about the data carried in the packets may not disclose the exact category of the flow, but it can be an essential input for further inspection mechanisms, e.g., giving a strong hint about what kind of signatures or in what place inside the packet the latter should look for (see Fig. 7).

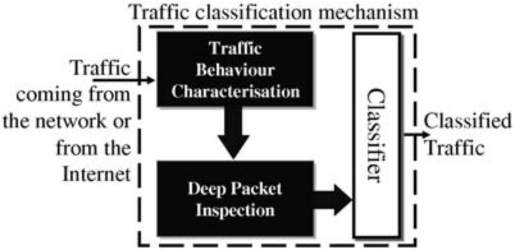

Traffic classification mechanism
Traffic coming from the network or from the Internet
Traffic Behaviour Characterisation
Deep Packet Inspection
Classifier
Classified Traffic

Fig. 7 Logical scheme of a traffic classifier using DPI and behaviour based approaches. In this case, the information retrieved by the behavioural methods is used to improve the results and the performance of the DPI mechanism

In some situations, it may also be useful to perform both analysis in parallel. For instance, some protocols try to disguise their traffic by making it to look like it belongs (or was generated) by another protocol. It is also common to find P2P protocols that are based on (or built on top of) other application protocols, like HTTP. In such case, it is possible that the traffic generated by some P2P applications is classified as being e.g., HTTP traffic, effectively hiding their true nature to the monitoring system. In this context, it would be interesting to perform both types of analyses concurrently and use the two results to identify the true nature of the traffic, as depicted by Fig. 8.

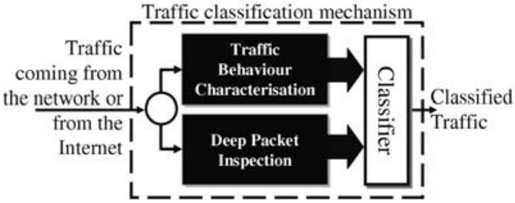

Traffic classification mechanism
Traffic coming from the network or from the Internet
Traffic Behaviour Characterisation
Deep Packet Inspection
Classifier
Classified Traffic

Fig. 8 Logical scheme of a traffic classifier using DPI and behaviour based approaches in a concurrent manner. In this case, the knowledge concerning the traffic nature provided by the two methodologies is used simultaneously to classify the traffic

5 Summary

The P2P architecture presents important advantages for computer networking, offering users the possibility to exchange contents easily and allowing content providers to deliver their services without having to support the high costs of powerful centralised systems. However, the impact P2P applications have in computer networks is not always welcomed by administrators and ISPs. As a consequence, the latter have to consider the option to block or limit such traffic, forcing the developers of P2P applications to respond by implementing techniques to avoid the detection mechanisms.

This cat and mouse game ends up undermining mechanisms which were advantageous for both content providers and ISPs. Content caching, which helps to increment the performance of P2P file sharing systems and reduce the costs supported by the ISPs [31], will be hardly useful if the packet payload is encrypted or if the protocol is disguised. The proliferation of P2P applications in the last few years show that they are not in the verge of disappearing and, as so, it becomes critical to foment the idea of coexistence and find *some room* for them in the network. A few projects for collaboration between ISPs and P2P applications, e.g., Proactive network Provider Participation for P2P (P4P) [36], are already being developed.

For all these reasons, it is important to understand the nature of P2P traffic, and gather the essential knowledge that enables one to design computer networks which are prepared to support the impact of P2P applications. It is equally important to develop efficient monitoring systems capable of controlling such traffic in real-time.

Acknowledgement

This work was supported by *Fundação para a Ciência e a Tecnologia*, Portugal, through the grant contracts SFRH/BDE/15592/2006, SFRH/BDE/15643/2006 and PTDC/EIA/73072/2006 TRAMANET: Traffic and Trust Management in Peer-to-Peer Networks, Nokia Siemens Networks Portugal S.A., Portugal and University of Beira Interior, Portugal.

References

1. BBC iPlayer. URL http://www.bbc.co.uk/iplayer/. Last access at August 20th, 2008
2. Dailymotion – Share Your Videos. URL http://www.dailymotion.com. Last access at August 8th, 2008
3. Joost – Free online TV. URL http://www.joost.com. Last access at August 20th, 2008
4. Miro – free, open source internet tv and video player. URL http://www.getmiro.com. Last access at August 20th, 2008
5. NetPDL Language Specification. URL http://www.nbee.org/doku.php?id=netpdl:index. Last access at August 7th, 2008
6. Snort – the de facto standard for intrusion detection/prevention. URL http://www.snort.org. Last access at November 30th, 2008
7. The NetBee Library. URL http://www.nbee.org/doku.php. Last access at August 7th, 2008
8. Wireshark: Go deep. URL http://www.wireshark.org. Last access at August 7th, 2008
9. YouTube – Broadcast Yourself. URL http://www.youtube.com. Last access at August 8th, 2008
10. Bernaille, L., Akodkenou, I., Soule, A., Salamatian, K.: Traffic Classification On The Fly. ACM SIGCOMM Computer Communication Review **36**(2), 23–26 (2006)
11. Blau, J.: Europe Looks for a Peer-to-Peer TV Alternative (2008). URL http://www.spectrum.ieee.org/apr08/6119. Last access at June 26th, 2008
12. Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., Tofanelli, P.: Revealing Skype Traffic: When Randomness Plays with You. In: Proceedings of the ACM SIGCOMM 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 37–48. ACM Press, New York, NY, USA (2007)
13. Constantinou, F., Mavrommatis, P.: Identifying Known and Unknown Peer-to-Peer Traffic. In: Proceedings of Fifth IEEE International Symposium on Network Computing and Applications, pp. 93–102 (2006)
14. Crawshaw, J.: Deep Packet Inspection: Taming the P2P Traffic Beast. Ligth Reading Insider **6**(7) (2006)
15. Dhamankar, R., King, R.: Protocol Identification via Statistical Analysis (PISA). White Paper, Tipping Point (2007)

16. Erman, J., Mahanti, A., Arlitt, M., Williamson, C.: Identifying and Discriminating Between Web and Peer-to-Peer Traffic in the Network Core. In: Proceedings of the 16th International Conference on World Wide Web, pp. 883–892. ACM Press New York, NY, USA (2007)
17. Finnish Communications Regulatory Authority: Decision of the Finnish Communications Regulatory Authority on Compliance with Law of the Sonera Puhekaista Service (2003). URL `http://www.ficora.fi/attachments/englanti/1156489127354/Files/CurrentFile/SoneraPuhekaista.pdf`. Last access at August 2nd, 2008
18. Gerber, A., Houle, J., Nguyen, H., Roughan, M., Sen, S.: P2P The Gorilla in the Cable. In: National Cable & Telecommunications Association (NCTA) 2003 National Show. Chicago, IL (2003)
19. Gomes, J., Inácio, P., Freire, M., Pereira, M., Monteiro, P.: Capturing the Nature of Peer-to-Peer Traffic Using Fast Entropy Analysis. Submitted for publication
20. Gomes, J., Inácio, P., Freire, M., Pereira, M., Monteiro, P.: Analysis of Peer-to-Peer Traffic Using a Behavioural Method Based on Entropy. In: Proceedings of the IPCCC 2008, IEEE International Performance Computing and Communications Conference (2008)
21. Johnson, M.E., McGuire, D., Willey, N.D.: The Evolution of the Peer-to-Peer File Sharing Industry and the Security Risks for Users. In: Proceedings of the 41st Hawaii International Conference on System Sciences, pp. 383–383. IEEE Press, Waikoloa, HI, USA (2008)
22. Karagiannis, T., Broido, A., Brownlee, N., Claffy, K., Faloutsos, M.: Is P2P Dying or Just Hiding. In: Proceedings of the IEEE GLOBECOM '04 Global Telecommunications Conference, vol. 3, pp. 1532–1538 (2004)
23. Karagiannis, T., Broido, A., Faloutsos, M.: Transport Layer Identification of P2P Traffic. In: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement, pp. 121–134. ACM Press New York, NY, USA, Taormina, Sicily, Italy (2004)
24. Karagiannis, T., Papagiannaki, K., Faloutsos, M.: BLINC: Multilevel Traffic Classification in the Dark. In: Proceedings of the ACM SIGCOMM '05 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, vol. 35, pp. 229–240. ACM Press New York, NY, USA, Philadelphia, Pennsylvania, USA (2005)
25. Karagiannis, T., Rodriguez, P., Papagiannaki, K.: Should Internet Service Providers Fear Peer-assisted Content Distribution? In: Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement, pp. 1–14. ACM, New York, NY, USA (2005)
26. Korpi, S.: Internet Telephony – Security Issues in Skype. Mobile Communities, Seminar on Internetworking (2006). Telecommunications Software and Multimedia Laboratory, Helsinki University of Technology
27. Lehtinen, J.: Design and Implementation of Mobile Peer-to-Peer Application. Master's thesis, Helsinki University of Technology (2006)
28. Madhukar, A., Williamson, C.: A Longitudinal Study of P2P Traffic Classification. In: Proceedings of 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 179–188 (2006)
29. Moore, A., Papagiannaki, K.: Toward the Accurate Identification of Network Applications. In: Proceedings of the 6th International Workshop in Passive and Active Network Measurement. Springer, Boston, MA, USA (2005)
30. Perényi, M., Dang, T., Gefferth, A., Molnár, S.: Identification and Analysis of Peer-to-Peer Traffic. Journal of Communications **1**(7), 36–46 (2006)
31. Rodriguez, P., Tan, S.M., Gkantsidis, C.: On the Feasibility of Commercial, Legal P2P Content Distribution. SIGCOMM Computer Communication Review **36**(1), 75–78 (2006)
32. Sabbagh, D.: BBC iPlayer 'risks overloading the internet' (2008). URL `http://technology.timesonline.co.uk/tol/news/tech_and_web/article3716781.ece`. Last access at August 20th, 2008
33. Sen, S., Spatscheck, O., Wang, D.: Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures. In: Proceedings of the 13th Conference on World Wide Web, pp. 512–521. ACM Press New York, NY, USA (2004)

34. Sen, S., Wang, J.: Analyzing Peer-to-Peer Traffic Across Large Networks. IEEE/ACM Transactions on Networking **12**(2), 219–232 (2004)
35. Shannon, C.E.: A Mathematical Theory of Communication. The Bell System Technical Journal **27**(3), 379–423 (1948)
36. Xie, H., Yang, Y.R., Silberschatz, A.: Towards ISP-Compliant, Peer-Friendly P2P Design. In: Proceedings of The IFIP International Federation for Information Processing Networking, pp. 375–384 (2008)
37. Yu, F.: High Speed Deep Packet Inspection with Hardware Support. Ph.D. thesis, University of California, Berkeley (2006)

Characterization of P2P Systems

Daniel Stutzbach and Reza Rejaie

Abstract The combination of large scale and geographically distributed nature of P2P system has led to their significant impact on the Internet. It is essential to characterize deployed P2P system for at least three reasons: (1) Accurately assessing their impact on the Internet, (2) identifying any performance bottleneck as well as any opportunity for performance improvement, (3) understanding user-driven dynamics in P2P systems.

To characterize a P2P system, one needs to accurately capture snapshots of the resulting P2P overlay. This is challenging because the overlay is often large and dynamic. While the overlay is discovered by a crawler, it is changing which leads to a distorted view of the system. Capturing unbiased view of the traffic in the overlay is equally challenging because it is difficult to show that the captured behavior represent the observed behavior by all peers.

In this chapter, we describe some of the fundamental problems in empirical characterization of widely deployed P2P systems. We present several examples to illustrate the effect of ad-hoc measurement/data collection on the resulting analysis/characterization. We then present two sampling techniques as a powerful approach to capture unbiased view of peer properties in a scalable fashion.

1 Introduction

Understanding existing systems and devising new P2P techniques relies on having access to representative models derived from empirical observations of existing systems. However, the large and dynamic nature of P2P systems makes capturing accurate measurements challenging. Because there is no central repository, data must be

Daniel Stutzbach
Stutzbach Enterprises, Dallas, Texas, e-mail: daniel@stutzbachenterprises.com

Reza Rejaie
University of Oregon, Eugene, Oregon, e-mail: reza@cs.uoregon.edu

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_45, © Springer Science+Business Media, LLC 2010

gathered from the peers who appear and depart as users start and exit the P2P application. Even a simple task such as counting the number of peers can be challenging since each peer can only report its immediate overlay neighbors.

The first half of this chapter surveys techniques for measuring attributes of P2P systems as well as characterizations derived from the application of those techniques. The second half explores two measurement techniques in detail – crawling and sampling – and demonstrates the importance of validating measurement methodology.

Systematically tackling the problem of characterizing P2P systems requires a structured organization of the different components. At the most basic level, a P2P system consists of a set of connected peers. We can view this as a graph with the peers as vertices and the connections as edges. One fundamental way to divide the problem space is into *properties of the peers* versus *properties of the way peers are connected*. Another fundamental division is examining the way the system *is* versus the way the system *evolves*. In some sense, any property may change with time and could be viewed as system evolution. We use the term “static properties” to refer to properties that can be measured at a particular moment in time and modeled with a static model (e.g., peer degree), and the term “dynamic properties” to refer to properties that are fundamentally dynamic in nature (e.g., session length).

Table 1 presents an overview of several interesting properties categorized by whether they are static or dynamic, and whether they are peer properties or connectivity properties.

2 Measurement Techniques

Empirical P2P studies employ one of five basic techniques, each offering a different view with certain advantages and disadvantages:

- **Passive Monitoring**: Eavesdrop on P2P sessions passing through a router.
- **Participate**: Instrument P2P software and allow it to run in its usual manner.

Table 1 Groups of properties

	Peer properties	Connectivity properties
Static properties	Available resources (e.g., files) Geographic location	Degree distribution Clustering coefficient Shortest path lengths Resiliency
Dynamic properties	Session length Uptime Remaining uptime Inter-arrival interval Arrival rate	Stable core Search efficiency Search reliability

Table 2 File sharing measurement studies, grouped by technique. The system under study is shown in parenthesis. B=BitTorrent, D=eDonkey 2000, G=Gnutella, K=Kad, N=Napster, S=Skype, O=Overnet, Z=Kazaa, *=Miscellaneous

Intercept	Participate	Crawl	Sample	Centralize
[37] (B,D,G,Z)	[17] (G)	[8] (G)	[7] (N,G)	[18] (B)
[14] (Z)	[21] (G)	[3] (G)	[42] (N,G)	[38] (B)
[23] (Z)	[22] (G)	[40] (G)	[4] (O)	[15] (B)
[24] (Z)	[35] (G)	[46] (G,K,B)	[11] (D)	[51] (*)
[41] (Z,G)	[44] (G)	[48] (G)	[13] (S)	
[43] (Z,G,*)	[2] (G)		[45] (K)	
[20] (B,D,G,Z,N,*)	[10] (B)			
	[27] (Z)			
	[28] (Z)			

- **Crawl**: Walk the P2P network, capturing information from each peer.
- **Sample**: Select a subset of the peers in the network.
- **Centralize**: Rely on logs maintained by a central server.

Table 2 summarizes the peer-reviewed studies in each category and lists the particular systems they examine. Studies which intercept data have typically focused on Kazaa, which was one of the most popular peer-to-peer systems. Saroiu et al. [41] show that in 2002 Kazaa traffic was between one and two orders of magnitude larger than Gnutella traffic. However, others studies tend to focus on Gnutella, which has several open source implementations available and open protocol specifications. Other popular file-sharing networks such eDonkey 2000, Overnet, and Kad remain largely unstudied. Each of the different measurement techniques has different strengths and weaknesses, explained in detail below.

2.1 Passive Monitoring

Monitoring peer-to-peer traffic at a gateway router provides useful information about dynamic peer properties such as the types and sizes of files being transferred. It also provides a limited amount of information about dynamic connectivity properties such as how long peers remain connected. However, passive monitoring suffers from three fundamental limitations, described below.

First, because it looks at only a cross-section of network traffic, usage patterns may not be representative of the overall user populations. For example, two of the most detailed studies of this type [14, 41] were both conducted at the University of Washington (UW). Because the University has exceptional bandwidth capacity and includes an exceptional number of young people, their measurements may capture different usage characteristics than, for example, a typical home broadband user. This limitation may be somewhat overcome by comparing studies taken from different vantage points. One study [43] overcomes the single-viewpoint limitation by capturing data at several routers within a Tier-1 ISP.

The second limitation of passive monitoring is that it only provides information about peers that are actively sending or receiving data during the measurement window. Monitoring traffic cannot reveal any information about peers which are up but idle, and it is not possible to tell with certainty when the user has opened or closed the application. These caveats aside, passive monitoring is quite useful for providing insight in file sharing usage patterns.

The third limitation is the difficulty in classifying P2P traffic. Karagiannis et al. [20] show that the most common method of identifying P2P traffic, by port number, is increasingly inaccurate.

The passive monitoring technique is predominantly used to study bulk data movement such as HTTP-like file transfers and streaming, where it is relatively easy to identify a flow at its beginning and count the bytes transferred.

2.2 Participate

Instrumenting open-source clients to log information on disk for later analysis facilitates the study of dynamic connectivity properties, such as the length of time connections remain open, bandwidth usage, and the frequency with which search requests are received. However, there is no guarantee that observations made at one vantage point are representative. Some studies employ multiple vantage points, but the vantage points still typically share common characteristics (e.g., exceptionally high bandwidth Internet connections) and still may not be representative.

2.3 Crawl

A crawler is a program which walks a peer-to-peer network, asking every known peer for a list of its neighbors to iteratively explore the entire graph, similar to the way a web-spider operates. Crawling is the only technique for capturing a full snapshot of the topology, needed for graph analysis and trace-driven simulation. However, accurately capturing the whole topology is tricky, particularly for large networks that have a rapidly changing population of millions of peers. All crawlers capture a distorted picture of the topology because the topology changes as the crawler runs.

2.4 Sample

Several studies gather data by sampling a set of peers in order to study static peer properties, such as link bandwidth and shared files. By sampling the set of peers at regular intervals, studies may also examine dynamic peer properties such as the session length distribution. To locate the initial set of peers, researchers have used techniques such as a partial crawl [4, 11, 13, 42], issuing search queries for common search terms [7, 42], and instrumenting a participating peer [7]. One drawback of sampling is that it is difficult to guarantee that the initial set of peers are representative. Additionally, when studying dynamic properties, sampling implicitly gathers more data from peers who are present for a larger portion of the measurement window.

2.5 Centralize

The final measurement technique is to use logs from a centralized source. Due to the decentralized nature of peer-to-peer networks, there typically is no centralized source. However, BitTorrent uses a centralized rendezvous point called a *tracker* that records peer arrivals, peer departures, and limited information about their download progress.

2.6 Summary

Measurement techniques for gathering data about the operation of peer-to-peer systems, summarized in Table 3, each have their advantages and disadvantages.

Table 3 Summary of existing measurement techniques

Technique	Advantages	Disadvantages
Passive monitoring	Provides information about traffic	May not be representative Omits idle peers Omits traffic on non-standard ports
Participate	Provides information about dynamic connectivity	May not be representative
Crawl	Captures the entire topology Unbiased	Doesn't scale May have significant distortion
Sample	Captures peer properties Unbiased techniques available	Haphazard sampling often unrepresentative Dynamic properties inherently biased toward long-lived peers
Centralize	Unbiased	Only available if system has a centralized component

3 What to Measure

The following subsections summarize other empirical studies of peer-to-peer systems, discuss their main findings, and identify important areas which remain unstudied.

3.1 Static Peer Properties

Saroiu, Gummadi, and Gribble provide an extensive and informative study, primarily of static peer properties [42]. While earlier work conceived of peers as equal participants, their landmark study demonstrates that in practice not all peers contribute equally to peer-to-peer systems. Using data collected from Gnutella and Napster in May 2001, their observations show a heavy skew in the distributions of bottleneck bandwidth, latency, availability, and the number of shared files for each host, with each of these qualities varying by many orders of magnitude.

Additionally, they found correlations between several of the properties. Bottleneck bandwidth and the number of uploads have a positive correlation, while bottleneck bandwidth and the number of downloads have a negative correlation. In other words, peers with high bandwidth tend to be uploading many files, while peers with low bandwidth have to spend more time downloading. Interestingly, no significant correlation exists between bottleneck bandwidth and the number of files stored on a peer.

In addition to the sweeping work of Saroiu et al. [42], several studies focus on examining the files shared by peers [2, 7, 11, 49]. A few results have consistently appeared in these studies. First, peers vary dramatically in the number of files that they share, with a relatively small percentage of peers offering the majority of available files. In addition, a large fraction of peers share no files at all (25% in [42], two-thirds in [2, 11], 11–13% in [49]). Second, the popularity of stored files in file-sharing systems is heavily skewed; a few files are enormously popular, while for most files only a few peers have a copy. Fessant et al. [11] found that it may be described by a Zipf distribution. However, Chu, Labonte, and Levine [7] found that the most popular files were relatively equal in popularity, although less popular files still had a Zipf-like distribution.

Studies also agree that the vast majority of files and bytes shared are in audio or video files, leading to the distribution of file sizes exhibiting a multi-modal behavior. Each studies shows that a plurality of files are audio files (48% in [11], 67% in [49], 76% in [7]). However, video files make up a disproportionately large portion of the bytes stored by peers (67% in [11], 53% in [49], 21% in [7]).

Fessant et al. [11] took the additional step of examining correlations in the files shared by peers. Their results show that users have noticeable interests, with 30% of files having a correlation of at least 60% with at least one other file. Of peers with at least 10 files in common, they found that 80% have at least one more file in common. Likewise, of peers with at least 50 files in common, in their data nearly 100% have at least one more file in common.

3.2 Dynamic Peer Properties

Most dynamic peer properties are tied to how long and how frequently peers are active. *Session length* is the length of time a peer is continuously connected to a given peer-to-peer network, from when it arrives until it departs. *Uptime* is the length of time a peer that is still present has been connected. *Remaining uptime* is how much longer until an active peer departs. *Lifetime* is the duration from the first time a peer connects to a peer-to-peer network –ever– to the very last time it disconnects. *Availability* is the percentage of time that a peer and its resources are connected to the peer-to-peer network within some window. *Downtime* is the duration between two successive sessions. Finally, an *inter-arrival interval* is the duration from the arrival of one peer until the arrival of the next peer. The session length, uptime, and remaining uptime are closely related, as shown in Fig. 1. The popularity of file transfers is another dynamic peer property, which we examine separately.

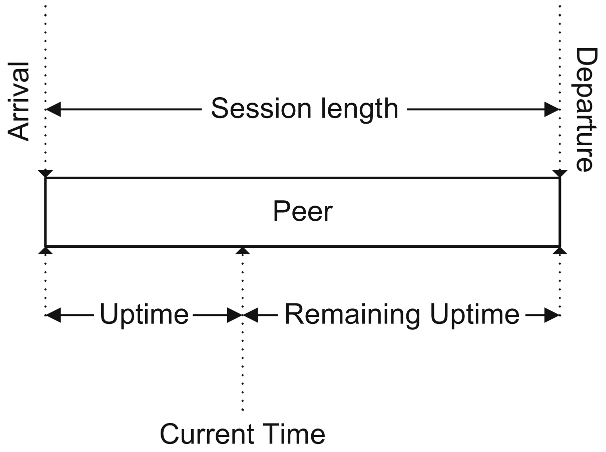


Fig. 1 Illustration of the relationship between session length, uptime, and remaining uptime

Generally, the most important distribution for simulation and analysis is the session-length distribution, as it fully determines the uptime and remaining uptime distributions and strongly influences the availability. The median session length specifies how much churn a protocol must cope with, and the shape of the distribution determines whether some peers are dramatically more stable than others.

Due to its importance, several studies examine the session length distribution. Rhea, Geels, and Kubiatowicz [39] summarize these studies, as shown in Table 4 which is adapted from their paper and updated with our more recent study [46].

While the median differs dramatically, all the studies agree that the session lengths are heavily skewed: many sessions are short, while some session are very long. Several studies draw the conclusion that session lengths can be modeled with a power-law (or Pareto) distribution [5, 13, 26] or exhibit heavy-tailed behavior [13, 14, 43]. Chu, Labonte, and Levine [7] fit session lengths to a log-quadratic

Table 4 Observed session lengths in various peer-to-peer file sharing systems. Adapted from [39]

Citation	Systems observed	Session time
[42]	Gnutella, Napster	50% $\leq$ 60 min
[7]	Gnutella, Napster	31% $\leq$ 10 min
[43]	Kazaa	50% $\leq$ 1 min
[4]	Overnet	50% $\leq$ 60 min
[14]	Kazaa	50% $\leq$ 2.4 min
[46]	Gnutella, Kad	50% $\leq$ 15 min–1 h
[46]	BitTorrent	50% $\leq$ 2 min–30 min

distribution, which can be viewed as a second-order variation of the Pareto distribution. Only one of the earlier studies [5] provide an analysis and fit to support their conclusion. However, Leonard, Rai, and Loguinov [25, pg. 8] suggest that the fit given in [5] seems implausible and point out some possible methodological errors. In our more recent study [46], we examine several common methodological problems that introduce bias into studies of peer churn and examine data from Gnutella, Kad, and BitTorrent. The session lengths we observed were heavily skewed, but did not agree with a power-law or Pareto distribution. However, they could be described with a Weibull distribution.

In BitTorrent, the availability of high-quality tracker logs facilitates the study of peer dynamics. Prior studies of BitTorrent [18, 38] show that session lengths are heavily skewed. However, they do not attempt to create a model based on the data. A surprising discovery shown by Izal et al. [18] is that many peers (81% in their trace) depart before downloading the entire file, while peers who do complete the download linger for more than six hours on average.

While most studies of peer dynamics focus on session length, Bhagwan, Savage, and Voelker [4] provide a study of peer availability in Overnet during January 2003. However, they find that the distribution of availability varies dramatically with the size of the measurement window, due to the significant fraction of hosts who appear briefly and only once.

3.2.1 Files Transfers

Another class of dynamic peer properties is related to the files that peers are actively transferring, which in some sense is the derivative of the files being stored on each peer (discussed above under Static Peer Properties). The properties of files being transferred are most often studied using passive monitoring at gateway routers. Two of the most detailed studies of files being transfered [14, 41] were both conducted at the University of Washington (UW). The first study, [41], focuses on comparing HTTP requests with P2P requests, demonstrating that P2P uses more than twice as much bandwidth as the web on their network. Although they found a smaller number of hosts are involved in the P2P traffic, each object is orders of magnitude larger. Furthermore, they show that a majority of the P2P traffic came from a few large video files. Their second study, [14], more closely examines the popularity and properties of different P2P objects. The popularity of different objects did not match a Zipf distribution, in contrast to the Zipf distribution of popularity observed for Web objects. The authors suggest this may be due to the fact that in P2P systems, users typically download an object at most once, while Web users may return to a website many times. Instead, they found that unpopular objects appear Zipf-like, while popular objects are relatively equal in popularity, matching the results for stored files seen in [7].

Leibowitz et al. [23, 24] provide measurements from an Israeli Internet Service Provider (ISP), and compare their findings with the UW studies. Interestingly, they find that while the UW is an overall provider of P2P content, the ISP they study is an overall consumer of P2P content. Their studies give particular focus to the idea of caching P2P content. In [23], they implement a transparent 300 GB cache yielding a 67% bandwidth savings.

3.3 Static Connectivity Properties

In 2000, a company called "Clip2" developed a Gnutella crawler and published their results on the web. Although not validated by peer-review, their analysis and topology captures have been widely used in simulation studies of improvements for Gnutella-like networks [1, 19, 29–31]. In [8], Clip2 presents analysis of snapshots they captured between June and August of 2000. Using their crawler which took around an hour to survey the entire topology, they gathered snapshots containing between 1,000 and 8,000 peers.

Their work suggests that the Gnutella network has a power-law degree distribution, based on plotting the degree distribution of their snapshots on a log-log scale and demonstrating a linear fit. Adamic et al. [1] repeat this analysis on similar data provided by Clip2. However, neither study considers alternative models of the degree distribution. Several later studies [9, 12, 19, 34, 50] rely on the power-law model, simulating Gnutella using random power-law topologies. Lv et al. [33] show that power-law networks exhibit poorer performance than other types of random graphs.

Ripeanu, Foster, and Iamnitchi [40] implemented a crawler and use it to examine properties of the Gnutella overlay topology. Their crawler uses a client-server architecture running on roughly 50 computers to crawl a 30,000 node network in a few hours. Their crawls were conducted in November 2000–May 2001. The size of the network grew from 2,063 to 48,195 peers over that time. They performed all-pairs shortest-path computations and plotted the distribution of path lengths. 95% of shortest-paths are 7 hops or less, with most shortest-paths being 4 or 5 hops long. They repeat the analysis of [1, 8] by plotting the degree distribution in log-log scale. In their November snapshot, the degree distribution appears linear on the log-log plot, suggesting a power-law distribution. Their March 2001 snapshot is different. Low-degree nodes are approximately equally common, though among high-degree nodes the distribution still appears linear on the log-log plot.

Given that their crawls take a few hours, and peer uptimes may be just a few minutes [7, 14, 43], it is very possible that these topologies are highly inaccurate, leading to a drastically distorted picture of the network. In [48], we created a new crawler, Cruiser, which can crawl the Gnutella network in around 4 min. Later in this chapter, we provide an overview of the design of cruisers and some of the techniques we used to validate the accuracy of its snapshots. Our measurements of Gnutella were not consistent with a power-law distribution; in fact, they showed that virtually all peers had a degree under 35.

3.4 Dynamic Connectivity Properties

In [48], we explore how heavily skewed session lengths influence the topological structure. We found that long-lived peers gradually find one another and form a stable "core" for the peer-to-peer network. By remaining in the system for a long time, these peers have more opportunity to find one another. Once found, these connections remain until one of the peers leave.

When a user starts their P2P application, the application must discover other peers to form connections with. This initial discovery process is called *bootstrapping*. Karbhari et al. [21] provide a comparative study of the bootstrapping mechanisms of several Gnutella implementations.

Sripanidkulchai presented one of the first studies of search terms in a P2P network [44], demonstrating that queries follow a Zipf distribution, except for the most popular queries which are of roughly equal popularity (similar to the distributions of files stored and file transfers). The fact that popular queries are much more common than unpopular queries suggests caching query results may be beneficial [35, 44].

Klemm et al. [22] provide a comprehensive analysis of queries, breaking down the number of queries observed by time of day and geographical region. It includes distributions for the number of sessions that generate queries, the time until the first queries, the query inter-arrival time, and the length of the session. In short, it provides a framework for generating a synthetic query workload as seen from a single peer.

3.5 Summary

Peer-to-peer systems have been a popular topic for empirical studies. Existing studies cover properties of stored files, file transfers, and search terms in great detail. Additionally, Saroiu, Gummadi, and Gribble [42] provide a comprehensive study of static peer properties. However, these measurement studies have been rather ad-hoc, gathering data in the most convenient manner without critically examining their methodology for measurement bias. While an ad-hoc approach is often suitable for first-order approximations (e.g., "file popularity is heavily skewed"), it is generally not appropriate for making precise conclusions (e.g., "session-lengths are power-law”). One of the largest gaps in the existing work is the development and validation of high-fidelity measurement tools for peer-to-peer networks. The remainder of this chapter describes two tools for gathering highly accurate measurements.

4 Cruiser: A Fast P2P Crawler

The global state of a peer-to-peer system is distributed among all the peers. Exploring the graph and capturing the state of each peer captures the global state. A *crawler* is a tool that captures global state in this way. Since the system changes as the crawler explores, the picture is *distorted*, much like a photograph capturing rapid motion. Therefore, crawl speed is important. The faster the crawler runs, the less distortion. Crawlers have most often been used for capturing snapshots of the overlay topology as a graph, needed for studying many static connectivity properties. Many studies [3, 8, 28, 40] take 30 to 120 min to crawl the network and capture a snapshot of the graph. However, many peers are not even present for that long [14, 39, 43]. This suggests existing crawlers are much too slow and may be capturing very distorted snapshots. The accuracy of these snapshots most likely has not been previously addressed because it is challenging to measure the distortion without a perfect reference snapshot for comparison.

This section documents a fast crawler, called *Cruiser*, that can capture complete topologies in just a few minutes, introduces techniques for assessing the accuracy of snapshots, and shows that Cruiser may be used to capture accurate snapshots. Additional details may be found in [48]. We focus on capturing snapshots of the Gnutella topology, as Gnutella is one of the largest P2P systems and has been the target of most prior P2P crawlers, allowing us to make meaningful comparisons. Although we focus on Gnutella, Cruiser uses a plug-in architecture, allowing it to crawl other P2P systems with the addition of an appropriate plug-in.

In order to crawl quickly, the design of Cruiser must overcome several challenges:

- It must make heavy use of parallelism to contact many peers simultaneously. Managing so many connections in parallel can lead to CPU bottlenecks requiring a distributed architecture.
- If the load is too great, Cruiser may lose data.[1] Therefore, Cruiser must carefully control the load by appropriately limiting the number of connections. Since each connection uses a variable amount of resources, this limit must be dynamic.
- Cruiser cannot afford to wait the minutes for a TCP connection attempt to timeout. Instead, the proper trade-off between timing out too quickly (which increases distortion by losing data) and timing out too slowly (which increases distortion by making the crawl slower) must be found empirically.

[1] For example, connections may appear to time out if the CPU load is so great that received packets spend too long in a queue before being processed.

4.1 The Design of Cruiser

Our primary goal in the design of Cruiser is to minimize distortion in captured snapshots by maximizing the speed at which Cruiser explores the overlay topology. We employ several techniques and features to achieve this design goal, as described below.

4.1.1 Two-Tier Networks

Cruiser leverages the two-tier structure of modern P2P networks by only crawling ultrapeers. Since each leaf must be connected to an ultrapeer, this approach enables us to capture all the nodes and links of the overlay by contacting a relatively small fraction of all peers. This strategy leads to a major reduction (around 85%) in the duration of a crawl without any loss of information.

4.1.2 Distributed Architecture

Cruiser employs a master-slave architecture in order to achieve a high degree of concurrency and to effectively utilize available resources on multiple computers. A master process coordinates multiple slave processes that act as independent crawlers and crawl disjoint portions of the network in parallel. The slaves communicate with the master using loose synchronization as follows. Each slave has an independent queue of addresses to contact, which the master fills. Each slave drains its queue by querying peers for their neighbors and reporting back with the data they've gathered. The master extracts new addresses from the data and uses this to fill the queues. The crawl terminates when all the queues are empty.

4.1.3 Asynchronous Communications

Each slave process crawls hundreds of peers in parallel using asynchronous communications. Cruiser implements an adaptive load management mechanism to ensure that slave processes remain busy but do not become overwhelmed. This is important for the steady progress of the crawl especially when different slave nodes have heterogeneous processing capabilities. Toward this end, each slave monitors its CPU load and adjusts its maximum number of parallels connections using an additive-increase multiplicative-decrease (AIMD) algorithm similar to TCP's congestion control mechanism. In practice, each PC typically runs with close to 1,000 parallel connections, contributing an additional speed-up of nearly three orders of magnitude, compared to using synchronous communications (as in [40]).

4.1.4 Appropriate Timeouts

When peers are unresponsive, waiting for TCP to timeout and give up attempting to connect takes a long time. On our systems, a full TCP timeout to an unresponsive address takes more than 3 min. While this is suitable for many interactive and automated applications, one of our primary design goals it to make crawls as quick as possible. We conducted an evaluation of the cost-versus-benefit tradeoff of different timeout values for crawling. As a function of the timeout length, Fig. 2 shows the duration of the crawl and the percentage of peers that were unreachable. We see that while very low timeouts (less than 10 s) result in a dramatic increase in the number of timeouts, there are diminishing returns for using longer timeout values, while the crawl length (and thus distortion) continues to increase. In other words, if a peer has not responded after 10 s, it is unlikely to ever respond. Therefore, we use a timeout of 10 s, providing an additional speedup of more than a factor of two, compared to using full TCP timeouts.

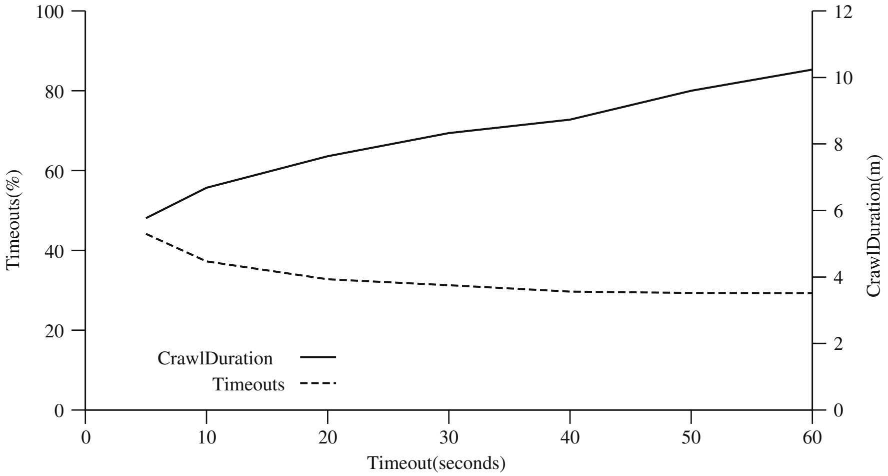


Fig. 2 Effects of the timeout length on crawl duration and snapshot completeness

4.2 Quantifying Snapshot Accuracy

One obvious metric to evaluate the performance of Cruiser is the time it takes to perform a crawl. However, the crawl duration doesn't reveal how accurate the crawl is; it only informs us if the crawl is more accurate than another crawl performed under similar conditions. Snapshot accuracy can not be directly measured since there is no reference snapshot for comparison. Therefore, we must indirectly quantify the effect of crawling speed on snapshot accuracy.

To examine the impact of crawling speed on the accuracy of captured snapshots, we adjust the crawling speed (and thus the crawl duration) of Cruiser by changing the number of parallel connections that each slave process can open. Using this technique, Cruiser can effectively emulate the behavior of previously reported crawlers which have a lower degree of concurrency.

We introduce the following two metrics for evaluating a crawler. The first metric, *edge distortion*, examines the edges in the captured snapshot. For each contacted peer A, with neighbors N_A, we examine each of its neighbors $B \in N_A$ to see if they likewise reported A as their neighbor. If not, we have an inconsistency in the graph caused by the fact that the edge changed sometime between crawling node A and crawling node B. The edge distortion, then, is the fraction of edges that are inconsistent.

The second metric, *node distortion*, examines the peers present in two consecutive snapshots captured back-to-back. We denote the peers as the sets V_1 and V_2. Comparing these two back-to-back snapshots provides insight into how distorted our picture of the network is. If Cruiser were instantly fast and captured perfect snapshots, V_1 and V_2 would be identical. The greater the change that occurs while Cruiser runs, the greater the difference between V_1 and V_2. We define the node distortion as $\frac{|V_1 \Delta V_2|}{|V_1|+|V_2|}$, where $V_1 \Delta V_2$ is the symmetric difference of V_1 and V_2 (i.e., peers in one set or the other, but not both). Note that when $V_1 = V_2$, the node distortion is 0%, and when V_1 and V_2 are completely disjoint the node distortion is 100%.

Figure 3 depicts peer and edge distortion as a function of crawl duration. This figure demonstrates that the accuracy of snapshots decreases with the duration of the crawl, because the increased distortion reflects changes in the topology that occur *while the crawler is running*. Crawlers that take 1–2 h (comparable to those in earlier

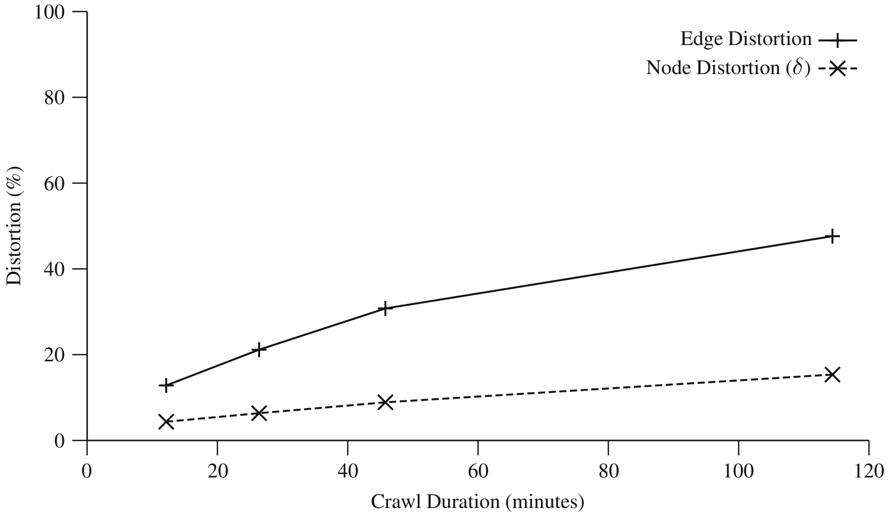


Fig. 3 Effect of crawl speed on the accuracy of captured snapshots

works) have a node distortion of 9–15% and an edge distortion of 31–48%, while at full speed Cruiser exhibits a node distortion of only 4% and an edge distortion of only 13%.

5 Sampling

While fast, Cruiser unavoidably takes $O(|V|)$ time, which means it may still be too slow to capture accurate snapshots of system with a very large population (V) or when the per-peer state is time-consuming to collect. For such cases, we need a mechanism to collect unbiased samples, which is the topic of the this chapter.

5.1 Sampling with Dynamics

We develop a formal and general model of a P2P system as follows. If we take an instantaneous snapshot of the system at time t, we can view the overlay as a graph $G(V,E)$ with the peers as vertices and connections between the peers as edges. Extending this notion, we incorporate the dynamic aspect by viewing the system as an infinite set of time-indexed graphs, $G_t = G(V_t, E_t)$. The most common approach for sampling from this set of graphs is to define a measurement window, $[t_0, t_0 + \Delta]$, and select peers uniformly at random from the set of peers who are present at any time during the window: $V_{t_0,t_0+\Delta} = \bigcup_{t=t_0}^{t_0+\Delta} V_t$. Thus, it does not distinguish between occurrences of the same peer at different times.

This approach is appropriate if peer session lengths are exponentially distributed (i.e., memoryless). However, existing measurement studies [18, 38, 42, 46] show session lengths are heavily skewed, with many peers being present for just a short time (a few minutes) while other peers remain in the system for a very long time (i.e., longer than Δ). As a consequence, as Δ increases, the set $V_{t_0,t_0+\Delta}$ includes an increasingly large fraction of short-lived peers.

A simple example may be illustrative. Suppose we wish to observe the number of files shared by peers. In this example system, half the peers are up all the time and have many files, while the other peers remain for around 1 min and are immediately replaced by new short-lived peers who have few files. The technique used by most studies would observe the system for a long time (Δ) and incorrectly conclude that most of the peers in the system have very few files. Moreover, their results will depend on how long they observe the system. The longer the measurement window, the larger the fraction of observed peers with few files.

One fundamental problem of this approach is that it focuses on sampling *peers* instead of *peer properties*. It selects each sampled vertex at most once. However, the property at the vertex may change with time. Our goal should not be to select a vertex $v_i \in \bigcup_{t=t_0}^{t_0+\Delta} V_t$, but rather to sample the property at v_i at a particular instant t. Thus, we distinguish between occurrences of the same peer at different times: samples $v_{i,t}$ and $v_{i,t'}$ gathered at distinct times $t \neq t'$ are viewed as distinct, even when they come from the same peer. *The key difference is that it must be possible to sample from the same peer more than once, at different points in time.* Using the formulation $v_{i,t} \in V_t$, $t \in [t_0, t_0 + \Delta]$, the sampling technique will not be biased by the dynamics of peer behavior, because the sample set is decoupled from peer session lengths. To our knowledge, no prior P2P measurement studies relying on sampling make this distinction.

Returning to our simple example, our approach will correctly select long-lived peers half the time and short-lived peers half the time. When the samples are examined, they will show that half of the peers in the system at any given moment have many files while half of the peers have few files, which is exactly correct.

If the measurement window (Δ) is sufficiently small, such that the distribution of the property under consideration does not change significantly during the measurement window, then we may relax the constraint of choosing t uniformly at random from $[t_0, t_0 + \Delta]$.

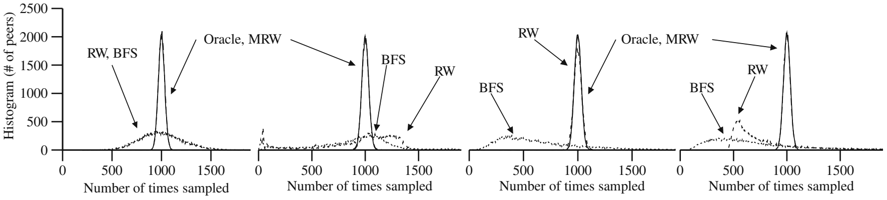
We still have the significant problem of selecting a peer uniformly at random from those present at a particular time. We begin to address this problem in the next section.

Table 5 Kolmogorov-Smirnov test statistic for techniques over static graphs. Values above $1.07 \cdot 10^{-4}$ lie in the rejection region at the 5% level

	Erdös–Rényi	Gnutella	Watts–Strogatz	Barabási–Albert
Breadth-First Search	$4.54 \cdot 10^{-4}$	$2.73 \cdot 10^{-3}$	4.73^{-3}	$2.77 \cdot 10^{-3}$
Random Walk	$3.18 \cdot 10^{-4}$	$1.57 \cdot 10^{-3}$	7.64^{-5}	$2.84 \cdot 10^{-3}$
Metropolis–Hastings	$5.97 \cdot 10^{-5}$	$5.79 \cdot 10^{-5}$	6.08^{-5}	$5.22 \cdot 10^{-5}$

5.2 Sampling from Static Graphs

We now turn our attention to topological causes of bias. Towards this end, we momentarily set aside the temporal issues by assuming a static, unchanging graph. The selection process begins with knowledge of one peer (vertex) and progressively queries peers for a list of neighbors. The goal is to select peers uniformly at random.



(a) Erdös–Rényi (b) Gnutella (c) Watts–Strogatz (small world) (d) Barabási–Albert (power-law)

Fig. 4 Bias of different sampling techniques; after collecting $k \cdot |V|$ samples. The figures show how many peers (y-axis) were selected x times

In any graph-exploration problem, we have a set of visited peers (vertices) and a front of unexplored neighboring peers. There are two ways in which algorithms differ: *(i)* how to chose the next peer to explore, and *(ii)* which subset of the explored peers to select as samples. Prior studies use simple breadth-first or depth-first approaches to explore the graph and select all explored peers. These approaches suffer from several problems:

- The discovered peers are correlated by their neighbor relationship.
- Peers with higher degree are more likely to be selected.
- Because they never visit the same peer twice, they will introduce bias when used in a dynamic setting as described in Section 5.1.

5.2.1 Random Walks

A better candidate solution is the random walk, which has been extensively studied in the graph theory literature (for an excellent survey see [32]). We briefly summarize the key terminology and results relevant to sampling. The transition matrix $P(x,y)$ describes the probability of transitioning to peer y if the walk is currently at peer x:

$$P(x,y) = \begin{cases} \frac{1}{\text{degree}(x)} & y \text{ is a neighbor of x,} \\ 0 & \text{otherwise} \\ & \end{cases}$$

If the vector v describes the probability of currently being at each peer, then the vector $v' = vP$ describes the probability after taking one additional step. Likewise, vP^r describes the probability after taking r steps. As long as the graph is connected and not bipartite, the probability of being at any particular node, x, converges to a *stationary distribution*:

$$\pi(x) = \lim_{r \longrightarrow \infty} (vP^r)(x) = \frac{\text{degree}(x)}{2 \cdot |E|}$$

In other words, if we select a peer as a sample every r steps, for sufficiently large r, we have the following good properties:

- The information stored in the starting vector, v, is lost, through the repeated selection of random neighbors. Therefore, there is no correlation between selected peers. Alternately, we may start many walks in parallel. In either cases, after r steps, the selection is independent of the origin.
- While the stationary distribution, $\pi(x)$, is biased towards peers with high degree, the bias is precisely known, allowing us to correct it.
- Random walks may visit the same peer twice, which lends itself better to a dynamic setting as described in Section 5.1.

In practice, r need not be exceptionally large. For graphs where the edges have a strong random component (e.g., small-world graphs such as peer-to-peer networks), it is sufficient that the number of steps exceed the log of the population size, i.e., $r \geq O(\log|V|)$.

5.2.2 Adjusting for Degree Bias

To correct the bias towards high degree peers, we make use of the Metropolis–Hastings method for Markov Chains. Random walks on a graph are a special case of Markov Chains. In a regular random walk, the transition matrix $P(x,y)$ leads to the stationary distribution $\pi(x)$, as described above. We would like to choose a new transition matrix, $Q(x,y)$, to produce a different stationary distribution, $\mu(x)$. Specifically, we desire $\mu(x)$ to be the uniform distribution so that all peers are equally likely to be at the end of the walk. Metropolis–Hastings [6, 16, 36] provides us with the desired $Q(x,y)$:

$$Q(x,y) = \begin{cases} P(x,y)\min\left(\frac{\mu(y)P(y,x)}{\mu(x)P(x,y)},1\right) & \text{if } x \neq y, \\ 1 - \sum_{z \neq x} Q(x,z) & \text{if } x = y \end{cases}$$

Equivalently, to take a step from peer x, select a neighbor y of x as normal (i.e., with probability $P(x,y)$). Then, with probability $\min\left(\frac{\mu(y)P(y,x)}{\mu(x)P(x,y)},1\right)$, accept the move. Otherwise, return to x (i.e., with probability $1 - \sum_{z \neq x} Q(x,z)$).

To collect uniform samples, we have $\frac{\mu(y)}{\mu(x)} = 1$, so the move-acceptance probability becomes:

$$\min\left(\frac{\mu(y)P(y,x)}{\mu(x)P(x,y)},1\right) = \min\left(\frac{\text{degree}(x)}{\text{degree}(y)},1\right)$$

Therefore, our algorithm for selecting the next step from some peer x is as follows:

- Select a neighbor y of x uniformly at random.
- Query y for a list of its neighbors, to determine its degree.
- Generate a random value, p, uniformly between 0 and 1.
- If $p \leq \frac{\text{degree}(x)}{\text{degree}(y)}$, y is the next step.
- Otherwise, remain at x as the next step.

We call this the Metropolized Random Walk (MRW). Qualitatively, the effect is to suppress the rate of transition to peers of higher degree, resulting in selecting each peer with equal probability.

5.2.3 Evaluation

Although [6] provides a proof of correctness for the Metropolis–Hastings method, to ensure the correctness of our implementation we conduct evaluations through simulation over static graphs. This additionally provides the opportunity to compare

MRW with conventional techniques such as Breadth-First Search (BFS) or naive random walks (RW) with no adjustments for degree bias.

To evaluate a technique, we use it to collect a large number of sample vertices from a graph, then perform a goodness-of-fit test against the uniform distribution. For Breadth-First Search, we simulate typical usage by running it to gather a batch of 1,000 peers. When one batch of samples is collected, the process is reset and begins anew at a different starting point. To ensure robustness with respect to different kinds of connectivity structures, we examine each technique over several types of graphs as follows:

- **Erdös–Rényi:** The simplest variety of random graphs
- **Watts–Strogatz:** "Small world" graphs with high clustering and low path lengths
- **Barabási–Albert:** Graphs with extreme degree distributions, also known as power-law or scale-free graphs
- **Gnutella:** Snapshots of the Gnutella ultrapeer topology, captured in our earlier work [48]

To make the results more comparable, the number of vertices ($|V| = 161,680$) and edges ($|E| = 1,946,596$) in each graph are approximately the same.[2] Table 5 presents the results of the goodness-of-fit tests after collecting $1000 \cdot |V|$ samples, showing that Metropolis–Hastings appears to generate uniform samples over each type of graph, while the other techniques fail to do so by a wide margin.

Figure 4 explores the results visually, by plotting the number of times each peer is selected. If we select $k \cdot |V|$ samples, the typical node should be selected k times, with other nodes being selected close to k times approximately following a normal distribution with variance k.[3] We used $k = 1,000$ samples. We also include an "Oracle" technique, which selects peers uniformly at random using global information. The Metropolis–Hastings results are virtually identical to the Oracle, while the other techniques select many peers much more and much less than k times. In the Gnutella, Watts–Strogatz, and Barabási–Albert graphs, Breadth-First Search exhibits a few vertices that are selected a large number of times ($> 10,000$). The (not-adjusted) Random Walk (RW) method has similarly selected a few vertices an exceptionally large number of times in the Gnutella and Barabási–Albert models. The Oracle and MRW, by contrast, did not select any vertex more than around 1,300 times.

In summary, the Metropolis–Hastings method selects peers uniformly at random from a static graph. The next section examines the additional complexities when selecting from a dynamic graph, introduces appropriate modifications, and evaluates the algorithm's performance.

[2] Erdös–Rényi graphs are generated based on some probability p that any edge may exist. We set $p = \frac{2|E|}{|V| \cdot (|V|-1)}$ so that there will be close to $|E|$ edges, though the exact value may vary slightly. The Watts–Strogatz model require that $|E|$ be evenly divisible by $|V|$, so in that model we use $|E| = 1,940,160$.

[3] Based on the normal approximation of a binomial distribution with $p = \frac{1}{|V|}$ and $n = k|V|$.

5.3 Empirical Results

In addition to the simulator version, we have implemented the MRWB algorithm for sampling from real peer-to-peer networks into a tool called `ion-sampler`. The following subsections briefly describe the implementation and usage of `ion-sampler` and present empirical experiments to validate its accuracy.

5.3.1 Ion-Sampler

The `ion-sampler` tool uses a modular design that accepts plug-ins for new peer-to-peer systems.[4] A plug-in can be written for any peer-to-peer system that allows querying a peer for a list of its neighbors. The `ion-sampler` tool hands IP-address:port pairs to the plug-in, which later returns a list of neighbors or signals that a timeout occurred. The `ion-sampler` tool is responsible for managing the walks. It outputs the samples to standard output, where they may be easily read by another tool that collects the actual measurements. For example, `ion-sampler` could be used with existing measurement tools for measuring bandwidth to estimate the distribution of access link bandwidth in a peer-to-peer system. Listing 1 shows an example of using `ion-sampler` to sample peers from Gnutella.

5.3.2 Empirical Validation

The topology snapshots from Cruiser provide a point of reference for the degree distribution to evaluate the accuracy of `ion-sampler` empirically. By capturing every peer, Cruiser is immune to sampling difficulties. However, because the network changes as Cruiser operates, its snapshots are slightly distorted. In particular, peers arriving near the start of the crawl are likely to have found additional neighbors by the time Cruiser contacts them. Therefore, we intuitively expect a slight upward bias in Cruiser's observed degree distribution. For this reason, we would not expect a perfect match between Cruiser and sampling, but if the sampling is unbiased we still expect them to be very close. We can view the CCDF version of the degree distribution captured by Cruiser as a close upper-bound on the true degree distribution.

Figure 5 presents a comparison of the degree distribution of reachable ultrapeers in Gnutella, as seen by Cruiser and by the sampling tool (capturing approximately 1,000 samples with $r = 25$ hops). It also includes the results of a short crawl,[5] a sampling technique commonly used in earlier studies (e.g., [42]). We interleaved

[4] In fact, it uses the same plug-in architecture as our earlier, heavy-weight tool, Cruiser, which exhaustively crawls peer-to-peer systems to capture topology snapshots.

[5] A "short crawl" is a general term for a progressive exploration of a portion of the graph, such as by using a breadth-first or depth-first search. In this case, we randomly select the next peer to explore.

running these measurement tools to minimize the change in the system between measurements of different tools, in order to make their results comparable.

Examining Fig. 5, we see that the full crawl and sampling distributions are quite similar. The sampling tool finds slightly more peers with lower degree, compared to the full crawl, in accordance with our expectations described above. We examined several such pairs of crawling and sampling data and found the same pattern in each pair. By comparison, the short crawl exhibits a substantial bias towards high degree peers relative to both the full crawl and sampling.

5.3.3 Efficiency

Having demonstrated the validity of the MRWB technique, we now turn our attention to its efficiency. Performing the walk requires $n \cdot r$ queries, where n is the desired number of samples and r is the length of the walk in hops. If r is too low,

```
bash$ ./ion-sampler gnutella --hops 25 -n 10
10.8.65.171:6348
10.199.20.183:5260
10.8.45.103:34717
10.21.0.29:6346
10.32.170.200:6346
10.201.162.49:30274
10.222.183.129:47272
10.245.64.85:6348
10.79.198.44:36520
10.216.54.169:44380
bash$
```

Listing 1: Example usage of the `ion-sampler` tool. We specify that we want to use the Gnutella plug-in, each walk should take 25 hops, and we would like 10 samples. The tool then prints out 10 IP-address:port pairs. We have changed the first octet of each result to “10” for privacy reasons.

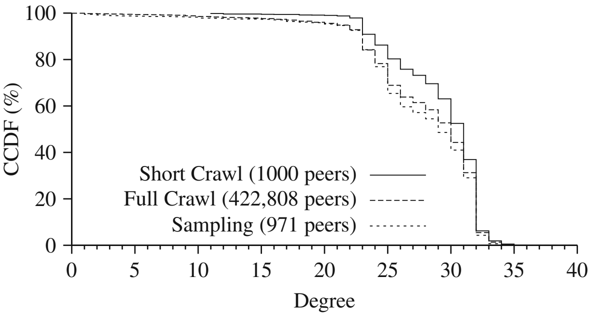


Fig. 5 Comparison of degree distributions observed from sampling versus exhaustively crawling all peers

significant bias may be introduced. If r is too high, it should not introduce bias, but is less efficient. From graph theory, we expect to require $r \geq O(\log|V|)$ for an ordinary random walk. Based on our empirical experiments in [47], we conservatively regard a choice of $r = 25$ as a safe walk length for Gnutella. Choosing $r = 25$, we can collect 1,000 samples by querying 25,000 peers, over an order of magnitude in savings compared with performing a full crawl which must contact more than 400,000.

6 Summary and Future Work

The first half of this chapter surveys techniques for measuring attributes of P2P systems as well as characterizations derived from the application of those techniques. The second half explores two measurement techniques in detail – crawling and sampling – and demonstrates the importance of validating measurement methodology.

In our ongoing work, we are exploring different techniques to improve the accuracy and efficiency of the crawling and sampling technique presented here (and earlier presented in [47, 48]). Additionally, we are examining large-scale traffic monitoring over Distributed Hash Tables (DHTs).

References

1. Adamic, L.A., Lukose, R.M., Huberman, B., Puniyani, A.R.: Search in power-law networks. Physical Review E **64**(46135) (2001)
2. Adar, E., Huberman, B.A.: Free riding on Gnutella. First Monday **5**(10) (2000)
3. Annexstein, F.S., Berman, K.A., Jovanovic, M.A.: Latency effects on reachability in large-scale peer-to-peer networks. In: Symposium on Parallel Algorithms and Architectures, pp. 84–92. Crete, Greece (2001)
4. Bhagwan, R., Savage, S., Voelker, G.: Understanding availability. In: International Workshop on Peer-to-Peer Systems (2003)
5. Bustamante, F.E., Qiao, Y.: Friendships that last: Peer lifespan and its role in P2P protocols. In: International Workshop on Web Content Caching and Distribution (2003)
6. Chib, S., Greenberg, E.: Understanding the Metropolis–Hastings algorithm. The Americian Statistician **49**(4), 327–335 (1995)
7. Chu, J., Labonte, K., Levine, B.N.: Availability and locality measurements of peer-to-peer file systems. In: ITCom: Scalability and Traffic Control in IP Networks II Conferences (2002)
8. Clip2.com, Inc.: Gnutella: To the bandwidth barrier and beyond (2000)
9. Crespo, A., Garcia-Molina, H.: Routing indices for peer-to-peer systems. In: International Conference on Distributed Computing Systems (2002)
10. Erman, D., Ilie, D., Popescu, A., Nilsson, A.A.: Measurement and analysis of BitTorrent signaling traffic. In: Nordic Teletraffic Seminar. Oslo, Norway (2004)

11. Fessant, F.L., Handurukande, S., Kermarrec, A.M., Massoulie, L.: Clustering in peer-to-peer file sharing workloads. In: International Workshop on Peer-to-Peer Systems (2004)
12. Gkantsidis, C., Mihail, M., Saberi, A.: Random walks in peer-to-peer networks. In: IEEE INFOCOM (2004)
13. Guha, S., Daswani, N., Jain, R.: An experimental study of the Skype peer-to-peer VoIP system. In: International Workshop on Peer-to-Peer Systems. Santa Barbara, CA, USA (2006)
14. Gummadi, K.P., Dunn, R.J., Saroiu, S., Gribble, S.D., Levy, H.M., Zahorjan, J.: Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In: ACM Symposium on Operating Systems Principles (2003)
15. Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., Zhang, X.: Measurements, analysis, and modeling of BitTorrent-like systems. In: Internet Measurement Conference. Berkeley, CA (2005)
16. Hastings, W.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika **57**, 97–109 (1970)
17. He, Q., Ammar, M.: Congestion control and message loss in Gnutella networks. In: Multimedia Computing and Networking. Santa Clara, CA (2004)
18. Izal, M., Urvoy-Keller, G., Biersack, E.W., Felber, P.A., Hamra, A.A., Garces-Erice, L.: Dissecting BitTorrent: Five months in a torrent's lifetime. In: Passive and Active Measurement Workshop (2004)
19. Jiang, S., Zhang, X.: Floodtrail: An efficient file search technique in unstructured peer-to-peer systems. In: Globecom. San Francisco, CA (2003)
20. Karagiannis, T., Broido, A., Brownlee, N., Claffy, K., Faloutsos, M.: Is P2P dying or just hiding? In: Globecom. Dallas, TX (2004)
21. Karbhari, P., Ammar, M., Dhamdhere, A., Raj, H., Riley, G., Zegura, E.: Bootstrapping in Gnutella: A measurement study. In: Passive and Active Measurement Workshop (2004)
22. Klemm, A., Lindemann, C., Vernon, M., Waldhorst, O.P.: Characterizing the query behavior in peer-to-peer file sharing systems. In: Internet Measurement Conference. Taormina, Italy (2004)
23. Leibowitz, N., Bergman, A., Ben-Shaul, R., Shavit, A.: Are file swapping networks cacheable? Characterizing P2P traffic. In: International Web Caching Workshop (2002)
24. Leibowitz, N., Ripeanu, M., Wierzbicki, A.: Deconstructing the Kazaa network. In: IEEE Workshop on Internet Applications (2003)
25. Leonard, D., Rai, V., Loguinov, D.: On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks. In: SIGMETRICS (2005)
26. Li, J., Stribling, J., Morris, R., Kaashoek, M.F.: Bandwidth-efficient management of DHT routing tables. In: USENIX/ACM Symposium on Networked Systems Design and Implementation. Boston, MA (2005)
27. Liang, J., Kumar, R., Ross, K.W.: The Kazaa overlay: A measurement study. Computer Networks Journal (Elsevier) (2005)
28. Liang, J., Kumar, R., Xi, Y., Ross, K.W.: Pollution in P2P file sharing systems. In: IEEE INFOCOM. Miami, FL (2005)
29. Liu, Y., Liu, X., Xiao, L., Ni, L.M., Zhang, X.: Location-aware topology matching in P2P systems. In: IEEE INFOCOM (2004)
30. Liu, Y., Zhuang, Z., Xiao, L., Ni, L.M.: AOTO: Adaptive overlay topology optimization in unstructured P2P systems. In: Globecom. San Francisco, CA (2003)
31. Liu, Y., Zhuang, Z., Xiao, L., Ni, L.M.: A distributed approach to solving overlay mismatching problem. In: International Conference on Distributed Computing Systems (2004)
32. Lovász, L.: Random walks on graphs: A survey. Combinatorics: Paul Erdös is Eighty **2**, 1–46 (1993)
33. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: International Conference on Supercomputing (2002)
34. Lv, Q., Ratnasamy, S., Shenker, S.: Can heterogeneity make Gnutella scalable? In: International Workshop on Peer-to-Peer Systems (2002)
35. Markatos, E.P.: Tracing a large-scale peer to peer system: an hour in the life of Gnutella. In: CC Grid (2002)

36. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E.: Equations of state calculations by fast computing machines. Journal of Chemical Physics **21**, 1087–1092 (1953)
37. Plissonneau, L., Costeux, J.L., Brown, P.: Analysis of peer-to-peer traffic on ADSL. In: Passive and Active Measurement Workshop, pp. 69–82. Boston, MA (2005)
38. Pouwelse, J., Garbacki, P., Epema, D., Sips, H.: The BitTorrent P2P file-sharing system: Measurements and analysis. In: International Workshop on Peer-to-Peer Systems. Ithaca, NY (2005)
39. Rhea, S., Geels, D., Kubiatowicz, J.: Handling churn in a DHT. In: USENIX, pp. 127–140 (2004)
40. Ripeanu, M., Foster, I., Iamnitchi, A.: Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. IEEE Internet Computing **6**(1) (2002)
41. Saroiu, S., Gummadi, K.P., Dunn, R.J., Gribble, S.D., Levy, H.M.: An analysis of Internet content delivery systems. In: Symposium on Operating Systems Design and Implementation, pp. 315–327 (2002)
42. Saroiu, S., Gummadi, P.K., Gribble, S.D.: Measuring and analyzing the characteristics of Napster and Gnutella hosts. Multimedia Systems Journal **9**(2), 170–184 (2003)
43. Sen, S., Wang, J.: Analyzing peer-to-peer traffic across large networks. IEEE/ACM Transactions on Networking **12**(2), 219–232 (2004)
44. Sripanidkulchai, K.: The popularity of Gnutella queries and its implications on scalability (2001). URL `http://www-2.cs.cmu.edu/~kunwadee/research/p2p/paper.html`
45. Stutzbach, D., Rejaie, R.: Improving lookup performance over a widely-deployed DHT. In: IEEE INFOCOM. Barcelona, Spain (2006)
46. Stutzbach, D., Rejaie, R.: Understanding churn in peer-to-peer networks. In: Internet Measurement Conference. Rio de Janeiro, Brazil (2006)
47. Stutzbach, D., Rejaie, R., Duffield, N., Sen, S., Willinger, W.: On unbiased sampling for unstructured peer-to-peer networks. IEEE/ACM Transactions on Networking (To appear)
48. Stutzbach, D., Rejaie, R., Sen, S.: Characterizing unstructured overlay topologies in modern p2p file-sharing systems. IEEE/ACM Trans. Netw. **16**(2) (2007)
49. Stutzbach, D., Zhao, S., Rejaie, R.: Characterizing files in the modern Gnutella network. Multimedia Systems Journal **13**(1) (2007)
50. Wang, C., Xiao, L., Liu, Y., Zheng, P.: Distributed caching and adaptive search in multilayer P2P networks. In: International Conference on Distributed Computing Systems (2004)
51. Yang, M., Zhang, Z., Li, X., Dai, Y.: An empirical study of free-riding behavior in the Maze P2P file-sharing system. In: International Workshop on Peer-to-Peer Systems. Ithaca, NY (2005)

Improving Peer-to-Peer Transport Paths for Content Distribution

Gerhard Hasslinger

Abstract Peer-to-peer networks have introduced a distributed communication paradigm, which is useful for many communication services and still accounts for a considerable portion of the Internet traffic. We discuss the efficiency of peer-to-peer content distribution as compared to server based overlays, which support search engines and many other popular web sites. Random source selection schemes of peer-to-peer protocols lead to long transmission paths and unnecessary high traffic load on inter-domain links. This chapter compares several recent proposals for a local exchange of popular data in the Internet with different implications for network resource efficiency, service provisioning and usage.

Key words: Peer-to-peer networks, Overlays, Content delivery, Application layer traffic engineering

1 Introduction

Overlays of various type are used to bridge heterogeneous networks for seamless end-to-end services. Current networking trends towards fixed/mobile convergence make them even more relevant for integrating different technologies with Internet access. A major advantage of overlays is their ability to provide new services or extend existing services on top of and independent of an underlying infrastructure.

Peer-to-peer (P2P) networks establish overlays on terminal equipment of the users, offering global services at a minimum of own network infrastructure for the provider. Fast delivery of large volumes of content within globally scattered communities is a main strength of the peer-to-peer principle, with scalability and adapt-

Gerhard Hasslinger
T-Systems, Deutsche-Telekom-Allee 7, D-64295 Darmstadt, Germany,
e-mail: gerhard.hasslinger@telekom.de

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_46, © Springer Science+Business Media, LLC 2010

ability for heterogeneous access [4, 27]. Peer-to-peer networks are highly efficient for several purposes due to their ability

- to exploit vacant resources (data, storage, computation power, bandwidth) distributed over user equipment,
- to adapt to varying demands with scalability for huge communities, including dynamic flash crowds,
- to embed support for search and replication of data for predefined demands in a self-organizing way,
- to build and manage overlays without or at low cost for network and server infrastructure.

On the other hand, transmission paths for P2P data exchange often span around the globe although very popular data is available at peers in the near of a destination. In contrast to P2P networks, content delivery networks (CDN) form overlays based on distributed server farms, which are known to do a good job in optimizing transmission paths and corresponding delay by delivering data from a server close to each requesting user [28]. Network providers include overlay traffic in the planning and upgrading process of their platforms which often enable mechanisms for load balancing. In this way, inefficient routing on higher layers can be partly reduced by optimization on lower layers, but overlay traffic engineering on the application layer offers additional optimization capabilities. On the other hand, smoothing effects of P2P traffic are visible

- in short term traffic variability on the time scales from milliseconds up to seconds, where the coefficient of variation is essentially lower for P2P traffic when compared to other traffic types on backbone links [18],
- in daily traffic profiles, where the peak-to-mean ratio is usually below 1.5 for P2P and close to 2 for web browsing and other services with direct human interaction [15],
- in the source distribution over the network topology, where flash crowd data is rapidly distributed and made available from all over the network, avoiding local bottlenecks.

Nevertheless, overlays may lead to increased overhead and cross-layer inefficiency by repeating similar and overlapping functions on higher layers or, in the worst case, by employing functions jarring with each other on different layers. Therefore overlays should be set up with implicit or explicit awareness of the underlying network structure and protocols. Proposals for a better supply of the application layer with information about the network topology include measurement based estimation of Internet coordinates [11, 22] and the installation of an information service or oracle collecting network layer information for support of higher layer protocols [1, 5, 21, 29]. It remains a challenging design problem for future Internet architectures to keep the layering structure simple and at the same time to keep the flexibility to respond to new requirements and technical opportunities as the main motivation for the emergence of overlays of various type.

After a discussion of the properties of P2P networking from the view of main involved parties in Sections 2, 3 compares access to and efficient distribution of data via P2P and CDN overlays. Approaches for improving P2P data exchange based on awareness of the network layer are addressed in Section 4 together with caching and bandwidth control options, followed by concluding remarks.

2 P2P Networking on Broadband Internet Platforms

Today, well-known and most popular peer-to-peer applications are file sharing and voice over IP (Skype) attracting millions of users, while a much wider spectrum of Internet application can more or less benefit from P2P networking. P2P solutions for online gaming, video streaming and IPTV also have a potential to extend to mass market and in addition, P2P overlay support is useful for small communities and enterprises. Although P2P protocol designers have developed highly efficient distributed data transfer and control schemes, there are still unsolved open issues for all parties involved in P2P communication, including security concerns and uncontrolled distribution of partly illegal content in self-organizing communities.

On the other hand, business solutions based on P2P with controlled access and digital rights management have also been set up. The BBC iPlayer is such a publicly accessible P2P service for downloading television and radio programs of the last seven days. as a step towards integration of Internet and television [3]. A corresponding research project P2P-NEXT is currently investigating P2P approaches for IPTV applications, which is funded by the European Union with major European broadcasters involved. The objectives of the project give an optimistic view on the applicability and future role of P2P for broadcasting, see <www.p2p-next.org>:

> *"P2P still has a somewhat dubious reputation as an illegal file sharing mechanism akin to Napster, Kazaa, Glockster, etc. Nevertheless, today it is considered by many as an efficient, reliable, and low cost mechanism for distributing any media file or live stream, and it is extensively used. Broadcasters and content providers consider P2P as a future-proof, universal, and ubiquitous two-way (interactive) distribution mechanism. Initially, P2P will complement the existing distribution mechanisms such as satellite, cable and terrestrial networks, but ultimately it may supersede them. The P2P-Next Project extends the notion of a conventional media distribution network. It introduces a concept of on-demand, personalised, and social network."*

From a general viewpoint, there are several parties involved in communication services, who look from different sides on P2P networking:

2.1 Users

The end users always benefit from new service opportunities. P2P approaches become popular when they are cheaper or give more comfortable access to resources than other Internet services. The motivation to make own resources available to others is limited and usually has to be supported by incentive mechanisms [10].

Nevertheless, user generated content on the Internet recently gained popularity via P2P or server platforms. YouTube <www.youtube.com> established a well known service for video streaming with a considerable traffic volume generated by users [7, 14].

2.2 P2P Service and Protocol Designers

P2P service and protocol designers can develop overlay services for global usage on the application layer without own infrastructure. But they have to be aware of the network and transport layer as well as the user behaviour to make a good fit of their approach into the telecommunication landscape. Some popular P2P networks (Guntella, eDonkey, BitTorrent, Skype etc.) seem to be initially launched by single developers or small groups without much support neither from major players in the telecommunications industry nor from universities. A framework for a simple and scalable protocol design is essential for P2P protocol development, allowing to launch and maintain services at minimum effort.

Progress has been made to overcome some problems of initial approaches e.g. to improve the search based on flooding messages in the original Gnutella protocol, which produced an enormous messaging overhead and did not scale when the population raised to millions. The success of the Skype network proved that voice over IP is feasible without special enforcement for quality of service, which was a surprising new experience. Skype was launched when the available bandwidth per users on broadband access platforms had become essentially larger than required for a single voice connection and it turned out that this kind of overprovisioning is sufficient for voice services to achieve acceptable quality. For high quality video applications the bandwidth on nowadays broadband IP networks is still insufficient, which is a main reason why P2P video streaming is much more challenging and up to now less successful. When performance demands for bandwidth or delay are critical then P2P solutions have to be optimized and thus become more elaborate in order to cope with other competitive approaches.

2.3 Content and Service Providers

Content providers may use P2P solutions as an alternative way of delivery, although the problem of file sharing abuse is a major concern. Economical aspects of P2P networking are addressed e.g., in [27]. Content delivery networks (CDN) build overlays on servers in the Internet architecture in order to support many popular web sites with an increasing spectrum of Internet services. They enable short transmission paths for enhanced quality of service, as discussed in more detail in Section 3.

2.4 Network and Broadband Access Providers

The network and Internet service providers (ISPs) also have an ambivalent view on P2P activity. P2P file sharing was and still is a main driver of bandwidth demand throughout the deployment phase of broadband access in recent time. This demand causes higher cost for upgrading the network capacities while also motivating a non-negligible group of users to subscribe to increasing access speeds. Asymmetrical digital subscriber lines (ADSL) are currently a prevalent deployment technology for Internet access from the homes, with presently more than 15 million DSL subscribers in Germany at rates in the range of 1–16 Mb/s. When the first 1–2 million ADSL subscriber lines were deployed in Germany in 2002, a majority of the users seemed to exploit their upstream bandwidth over long time periods probably for P2P networking. Since then the broadband access population has increased 10fold utilizing a much broader spectrum of applications, including file sharing as one of many components, although still generating a considerable portion of the traffic. The bandwidth consumption per user is now distributed over a wide range, see Fig. 1. Nevertheless, the portion of subscribers who fully exploit their access speed is decreasing while the mean traffic volume per user is increasing at a moderate pace, essentially lower than for the available access bandwidth.

Considering a current population of more than 15 million ADSL subscribers in Germany being connected at a mean access bandwidth of at least 3 Mb/s, we have well over 45 Tb/s of access bandwidth deployed in Germany to the homes in total. On the other hand, the Higher Speed Study Group of the Ethernet Alliance [12] recently started work on draft standards for 40 and 100 Gb/s Ethernet for the aggregation of future Internet platforms, such that 10 Gb/s lines will remain the maximum Ether-net speed for the next years. The large discrepancy between the bandwidth which is deployed in the access and in the aggregation reveals that only a minority of the users exploit their access speed allowing for a high statistical multiplexing gain traffic aggregation on broadband access platforms.

2.5 Current Shifts in Internet Usage and Traffic Profiles

Network dimensioning currently can rely on 80:20 rules, where a small fraction of subscribers generates most of the traffic, while the mass of users runs less bandwidth consuming services. Some typical statistics on the traffic contribution of most active users are shown in Fig. 1 based on the byte volume counted per IP address in online sessions on a downstream ADSL aggregation link over a day with about 100 000 different IP addresses visible in the measurement. IP addresses are assigned dynamically to users and are expected to change once or a few times for an active user over a day, such that the volume per user represents one or a sum of several volumes per IP address. The statistics per IP address indicates an even more extreme distribution than an 80:20 assumption, where in fact 90% of the traffic refers to less than 5% of the IP addresses. Although the trend to broader multi purpose usage reduces the P2P traffic portion, ISPs and users still face the problem of inadequate charging for the majority of traffic on the Internet [27]. Subscribers prefer flat rate access without accounting for the traffic volume, which is offered with only minor differentiation in access speeds. Volume based tariffs combined with a low monthly fee and flat rate up to a volume limitation would be sufficient for most users, but then they are often left without awareness or control of costs in excess of the limit until a next monthly bill. Even if the costs for transport of high traffic volumes in the backbone is only a part of the access provisioning costs, flat rate pricing seems to privilege the heavy traffic producers on account of all other users.

In near future, a new wave of video and IPTV services is expected to be delivered by broadcasters and content providers partly via P2P and partly via CDNs [9, 13, 24]. Broadcast and on demand video applications have the potential for another essential increase in the traffic volume and for shifts in the usage statistics such that the fraction of users generating negligible traffic will decrease and the fraction of contributors of large volumes will increase.

3 Distribution and Access to Content on the Internet

Popular client-server based web sites usually are supported by content delivery networks (CDN). A study of transfer paths in Akamais CDN [28] shows how users are redirected from the original web site by the underlying CDN to a close-by server within a hierarchical server farm consisting of thousands of servers. The connection is dynamically switched to another server if performance measurement and load

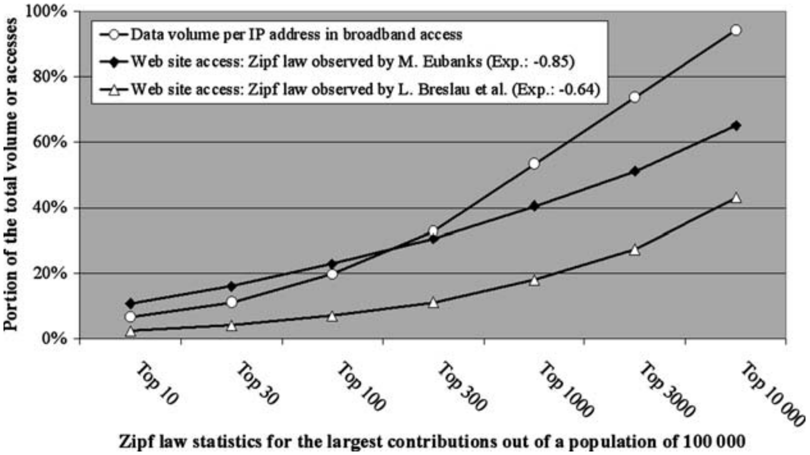


Fig. 1 Zipf law distributions for data volume and access frequencies of web sites

conditions indicate better quality of service for the new path. Su et al. [28] confirm that CDNs are efficient in shortening transmission paths and improving delays and throughput as main quality of service characteristics.

3.1 CDN Versus P2P Content Distribution

As compared to CDN transfers from a server in the near, P2P downloads usually experience much longer paths and delays which also affect throughput and reliability. For network providers, unnecessary long transfer paths impose higher load on peering and interconnection routes including expensive intercontinental links [5, 21]. Figure 2 illustrates the different behaviour of CDN and P2P networks, depicting a usual topology of broadband access networks, where tree-shaped access regions are attached to edge routers of the backbone at points of presence (PoPs).

Considering large provider networks serving millions of subscribers, it can be expected that a majority of the data of a global file sharing network is already found to be replicated on the same ISP platform and often already in the same access region of a P2P downloader. The fact that the major portion of downloads is addressed to a small set of the currently most popular files strengthens this effect.

Sometimes a tendency for P2P content to be exchanged locally arises within social groups. A separation of user communities and content due to different languages is most obvious. When looking at downloads of German content to a German destination via eDonkey, it is not surprising to find another 80:20 rule, such that about 80% of the sources are again located in Germany, whereas the opposite, i.e., less than 20% of source locations in Germany are observed for downloads of English content [26]. The assignment of peers to the same supernode in the eDonkey network may also generate locality within the reach of a supernode, but which does not correspond to national or ISP boundaries.

Traffic locality also has been investigated for data exchange between regions within an ISP platform by Cho et al. [8] using measurement from several Japanese service providers. Based on IP address analysis, they found user-to-user traffic to be dominant, which may correspond to P2P or other applications running between users. The traffic matrices between regions are determined and do not reveal significant local dependencies, i.e., traffic is exchanged between neighboring regions at about the same rates as between distant regions.

The lack of regional locality in traffic matrices is also visible for Internet traffic in Germany, in contrast to traffic on voice platforms, e.g., ISDN, with a considerable portion of local calls. But even when voice networks migrate to voice over IP, voice will contribute only a small fraction of the total IP traffic and thus locality in IP traffic matrices is expected to stay at a low level.

3.2 Zipf Laws and 80:20 Rules

The relevance of Zipf distributions and corresponding 80 : 20 rules for access to a large set of items by a large population is observed in many case studies. According to a Zipf law, the item that has rank R in the order of highest access frequency attracts $A(R) = C \cdot R^{-E}$ accesses, with $0 < E < 1$ and a constant C, which determines the total number of accesses. As a consequence, a large portion of all accesses concentrates on a few most popular items. Zipf-like distributions for web sites have been investigated by Breslau et al. [6] based on a set of measurements from half a dozen different networks, yielding parameters in the range $0.64 < E < 0.85$. Recent studies of access to YouTube [7, 14] again show Zipf-like distributions although with deviation for seldom accessed videos. M. Eubanks [13] reports similar results for the popularity of content on America Free TV <americafree.tv> and for other cases e.g., book selling. Thus, an essential portion of P2P data transfers could already be found and delivered within an ISP's platform and even without crossing the backbone.

4 Optimized Data Exchange in P2P Networks

The problem of replacing long transmission paths between peers around the globe by local data exchange between peers close to each other is addressed in a number of recent approaches. Information servers are proposed [1, 5, 29], which can be queried

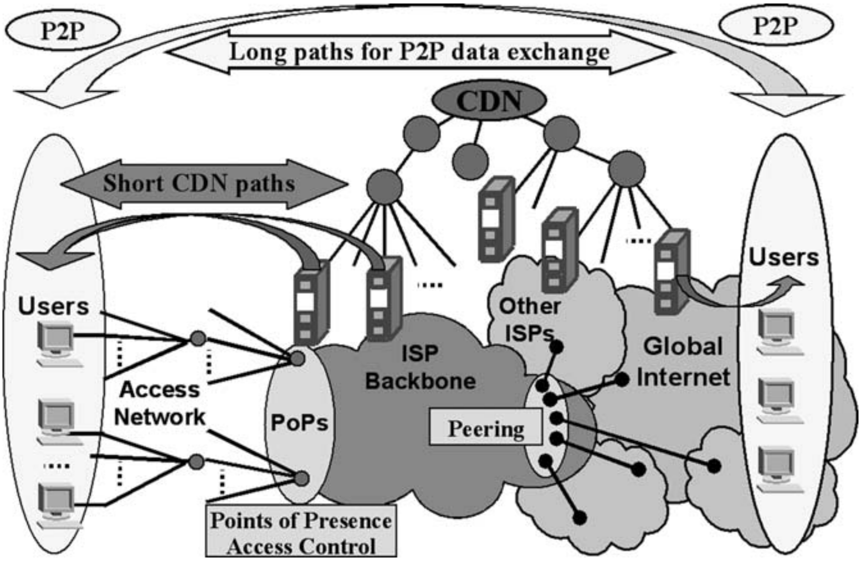


Fig. 2 P2P and CDN overlays on broadband access platforms

by applications in order to localize data sources in the near based on information from network providers and other parties. Alternatives are coordinate systems to be established for the sources of a distributed applications [11, 22] and caches. With knowledge of the position of peers or servers, the data flows of an application can be optimized, even regarding bottlenecks on transmission links [29]. Since some of those proposals have to rely on a common and standardized concept, the Internet Engineering Task Force (IETF) has started activity in this area.

4.1 Current IETF Discussion on Application Layer Traffic Engineering

Presently, the Internet Engineering Task Force (IETF) is preparing a working group on application layer traffic optimization (ALTO) [21], aiming at a standardized support to improve localized data exchange in P2P and other types of content distribution networks. On the IETF-72 meeting in August 2008 in Dublin, the ALTO BoF (Birth of Feathers) attracted about 300 attendees. Although they could not agree to immediately proceed to IETF working group status, a problem statement and a collection of preliminary solution approaches have already been documented in about half a dozen drafts as work in progress [21].

The basic problem to be addressed is how distributed applications can get information about the locations and distances between the involved hosts or peers. A main focus is on a cooperation between network providers and other parties who are able to provide location information for application protocols. Several projects and studies are already investigating approaches for a globally accessible information service that can be fed by parties with knowledge about network topologies within the Internet [1, 5, 29]. Distributed applications can access the information server through queries in order to optimize host or peer relationships for data exchange. As a starting point, information services about locations are already online, e.g., the Prefix WhoIs service <www.pwhois.org> or <www.closestnode.com>. Main preconditions and aspects of a standardized extension of a location information service are

- Which kind of metrics and information is useful to estimate distances and local relationship between different hosts?
- Which parties are expected to provide such information based on which own interest or incentives?
- How should an interface to the service be designed for access by applications?
- How can availability and scalability be achieved and failure cases be handled?
- What is the effect of the location information service for users and traffic pattern in network provider platforms? How can it be measured and optimized?

Two basic methods are considered for traffic path optimization:

- IP address mapping as done by the Prefix WhoIs service, for assigning the autonomous system to an IP address as a crude classification or at a finer granularity and
- Probing methods to measure delay, throughput or other performance parameters. This approach is most sensitive and responsive to current network conditions, but requires considerable effort and overhead. It may be too time consuming for small data transfers. Probing can be implemented on the application layer in the P2P protocol also without involving servers.

A coordinate system built on probing for delays between BitTorrent peers has been studied and implemented in test versions of the Azareus client [22]. The probing is piggybacked on other messages between peers to reduce probe message overhead. As a result, a two dimensional coordinate system is constructed with an additional height component to account for delays in the access. Much effort is needed to maintain a coordinate system. The variability of delay measurement results over time, changes in the routing and churn are among the main factors detracting from the accuracy. The study [22] concludes that useful coordinates can be established on application layer, but the effort to employ a coordinate system seems to be affordable only for large scale P2P networks.

A server system can combine information gathered from several sources to obtain more precise information with less effort. When a P2P download is initiated, the P2P protocol usually determines a list of possible sources offering the required content, which is handed over to the client, who connects to several proposed sources until a sufficient throughput is obtained in a multi source download process. While popular P2P protocols currently choose the sources more or less at random, a server with topology awareness can rank sources in the list due to distances from the requesting peer. In the ranked list, the client can start choosing close-by sources, which may also be preferable with regard to throughput and delay, when they still have enough up-stream capacity available. The solution again involves some overhead and has to be included into each P2P protocol, depending on servers to collect information from network providers or other parties who contribute knowledge on Internet topology and distances.

4.2 Experience from the P4P Project Including Traffic Engineering

A testbed for application layer traffic optimization has already been set up by the P4P project <codex.cs.yale.edu/avi/home-page/p4p-dir/p4p.html > and a P4P working group hosted by the Distributed Computing Industry Association (DCIA) <www.dcia.info>. Recent results of a field trial [29] claim that the mean path length for P2P downloads could be reduced from 5.5 to 0.9 metrohops within the Verizon intra-domain network. In general, the gain for intra-domain paths depends on the size and the structure of a provider platform, which often spans a smaller area with less than 5 hops to cross the backbone.

On the other hand, more improvement is expected for inter-domain traffic paths, which traverse the boundaries of network platforms partly on expensive intercontinental links. Network providers would profit a lot from redirecting traffic paths into their platform wherever possible. A classification of sources according to autonomous systems (AS), which usually correspond to network regions under common administration, is helpful to support this approach. Since not all P2P traffic can be bound within an AS, network providers could add policy information about which neighboring ASes are preferable for connections to peers in their platform, based on their knowledge of inter-AS connectivity and bottlenecks as well as the expenses for utilizing inter-AS links.

A corresponding inter-domain scenario has also been investigated by the P4P project [29], considering usual BitTorrent traffic pattern and volume according to measurement in the Abilene network in 2007. In one scenario, the traffic on the assumed inter-AS links is reduced to 2/3 by preferring sources based on low delay, whereas a second server-based scenario including the P4P traffic optimization method even yields less than 40% of the original P2P traffic load on inter-AS links.

The application layer traffic engineering approaches studied in the P4P project include load balancing to avoid bottlenecks and to minimize costs associated with traffic on network links. Nevertheless, this raises the question whether the network providers or the applications should be responsible for traffic engineering in the future Internet and what is the overall effect when this is done on the network and on the application layer? Counterproductive scenarios may arise, e.g. when the application layer optimization reorganizes the overlay network structure in order to avoid a bottleneck link. As a consequence, the traffic matrix on the network layer will observe a corresponding shift in traffic demands and the network provider may no longer see bottlenecks and the need to upgrade them.

In practice, traffic engineering has to take link failures into account as well as a link upgrading process for steadily increasing Internet traffic [24]. On the network layer, load balancing solutions are available for that purpose [19], but it seems challenging to combine or even integrate them with similar application layer functions under different administration.

4.3 Influence of Biased Source Selection on the Overlay Topology

The proposed biased source selection for exploiting local data exchange has an impact on the overlay network topology. Random source selection leads to a strongly connected global mesh network without hierarchical or regional structures. Randomly built graphs develop towards scale-free networks with small world effects. The global Internet structure is based on random and scale-free properties with a diameter of only about 20 [2, 17].

Biased local source selection strengthens a tendency to build a number of clusters with strong internal connectivity and a loose coupling between those clusters. With preference for connections between nodes in the same AS, clusters or subnet structures can be expected within each AS. Thus improved locality may slow down the distribution of content over network boundaries of ISPs. In the worst case, strict local preference and a high churn in P2P networks may lead to separated subnets for the same content.

Thus, sufficient inter-domain connectivity has to be sustained. An obvious approach is to mix biased local preference with randomness. Then the portion of random selection has to keep the inter-domain traffic throughput at a smallest possible level in a trade-off between avoiding a slow down of content propagation around the globe. The portion depends on the size of the overlay and can be smaller in large overlays, allowing for more efficient localization. Alternative approaches to maintain a structured and optimized overlay network for inter-domain connections have to cope with the randomness and churn in P2P networks.

4.4 Caches

Web caches provide another opportunity to optimize traffic paths [15, 25]. In principle, caches are highly efficient for P2P data, again because of Zipf-like distributions [6, 13, 14] for accesses to content. Therefore storing a small fraction of most popular content is sufficient to attract a high hit rate. Problems of outdated data as in classical web caches are less relevant for most P2P content, which is uniquely identified by hash algorithms. Caches can be placed in aggregation nodes for broadband access close to the subscriber lines yielding minimal transport path lengths.

On the other hand, the problem of supporting the distribution of copyright infringing or other illegal content by caches is apparent. In principle, the content of web caches always reflected content of the Internet, including problematic content although in an agnostic way. Nevertheless, the problem now has attracted much more attention and is pursued by various counteractions since file sharing is suspected to be responsible for decreasing revenues for content providers in the music and film industry. Filtering or classification of content in a cache is as difficult as filtering web content requiring expensive deep packet inspection methods. Since users distrust content inspection which can be seen in conflict with privacy laws of many countries, ISPs should use filtering only on behalf of legal authorities rather than on their own policies.

Since October 2004, the eDonkey/eMule P2P network offered an option to use the web caches of network providers by disguising P2P downloads as usual HTTP web browsing requests. When investigating the usage of this cache option in 2006, only a small portion of 5–10% of the download volume was observed to be supported by caches, although cases of downloading from the cache achieved essentially, often 5-fold higher throughput [26]. Later protocol versions did not proceed with cache support, see <wiki.emule-web.de/index.php/webcache>.

While file sharers obviously could benefit from cache downloads at higher throughput, the alternative of biased source selection with local preference is more favourable for network providers. The entire throughput of a P2P overlay is limited by the upstream capacities of the participating peers as a bottleneck especially in ADSL access. For biased source selection, this bottleneck remains unchanged even if it may be better exploited. But when caches are included, then the access bandwidth of the caches becomes available for downloading in addition to the upstream bottle-neck. Although the bandwidth of a cache is under control of the service provider, file sharing can utilize the cache as well as the P2P overlay to maximize the throughput. The network provider may experience P2P traffic load to persist in the backbone and even to increase in the access due to cache support. From the user perspective, caches in the access would be ideal to improve the throughput and to shorten transport paths and corresponding delays.

In principle, network providers can offer the most efficient support for content distribution using their own infrastructure including multicast/broadcast services. In this way, a single provider can't build architectures of global reach, but it is appropriate for offering regional services or IPTV within a country or lingual community.

4.5 Traffic Control for Bottlenecks

Finally, network providers can try to enforce locality by controlling and reducing the traffic especially on expensive links on peering and transatlantic routes. For that purpose, P2P traffic has to be classified and differentiated from other traffic. There are more than a dozen vendors of application identification systems and experience has been published for detection of main P2P protocols [5, 20]. Reducing P2P traffic on the borders of an ISP network would again give preference for shorter intra-domain download paths.

Nevertheless, P2P traffic classification is subject to non-negligible effort, as demonstrated by more than 1000 behaviour patterns being included in an thorough decision process for classification of BitTorrent [20]. The European Advanced Networking Test Center <www.eantc.com> recently invited manufacturers of P2P filtering solutions to an independent test, but only two vendors agreed to publish their test results [23].

Another problem of filtering can be seen in the fact that there is no clear reason for ISPs to differentiate traffic belonging to the same best effort class, even if it may result in better performance for all users. It would be much easier to differentiate traffic based on QoS classification by the users themselves, such that streaming or voice traffic is prioritized already in the user equipment [16]. But QoS marking and differentiation is rarely applied end-to-end, although most routers in broadband access networks support differentiated services.

5 Conclusions

The capability to launch new Internet services on the application layer without directly involving network infrastructure is a basic precondition for the success of peer-to-peer overlays. Nevertheless, advanced approaches for distribution of large data vo-lumes have to avoid unnecessary traffic load on network platforms in order to stay competitive to alternative ways of content distribution via server based overlays.

Therefore most frequently accessed content has to be delivered from nearby sources. Several recent approaches to improve traffic engineering based on locality awareness for peers or servers in overlays are currently discussed for possible standardization at the IETF with their consequences from the view of content, network, service providers and the users. Although it seems open which approaches will be implemented and will play an important role in the future Internet, progress can be expected towards more efficient localized transport on network platforms from which the users can also benefit in terms of shorter delay and higher throughput.

References

1. V. Agrawal, A. Feldmann and C. Schneideler, Can ISPs and P2P users cooperate for improved performance? ACM SIGCOMM Computer Communication Review 37 (2007) 31–40
2. R. Albert and A.-L. Barabasi, Statistical Mechanics of Complex Networks, Review of Modern Physics 74, 47 (2002) 1–54
3. BBC iPlayer <www.bbc.co.uk/iplayer/> (since 2007)
4. E. Biersack et al., Overlay architectures for file distribution: Fundamental analysis for homogeneous and heterogeneous cases, Computer Networks 51 (2007) 901–917
5. R. Bindal et al., Improving traffic locality in BitTorrent via biased neighbor selection, ICDCS (2006)
6. L. Breslau et al., Web caching and Zipf-like distributions: Evidence and implications, Proc. IEEE Infocom (1999)
7. X. Cheng, C. Dale and J. Liu, Statistics and social network of YouTube videos, IEEE Proc. International Workshop on Quality of Service (IWQoS), Twente, The Netherlands (2008) 249–258
8. K. Cho, K. Fukuda, H. Esaki and A. Kato, The impact and implications of the growth in residential user-to-user traffic, ACM Sigcomm Conf., Pisa (2006) <www.acm.org/sigs/sigcomm/sigcomm2006>
9. Cisco Systems, Global IP traffic forecast and methodology, White paper (2008) <www.cisco.com>
10. B. Cohen, Incentives build robustness in BitTorrent, <bitcon-jurer.org/BitTorrent/bittorrentecon.pdf> (2003)
11. F. Dabek, R. Cox, F. Kaashoek and R. Morris, Vivaldi: A decentralized network coordinate system, ACM Sigcomm Conf., Portland, USA (2004) <conferences.sigcomm.org/sigcomm/2004/>
12. Ethernet Alliance, Higher Speed Ethernet Study Group (2008) <www.ethernetalliance.org>
13. M. Eubanks, The video tsunami: Internet television, IPTV and the coming wave of video on the Internet, Pleanry talk, 71. IETF meeting (2008) <www.ietf.org/proceedings/08mar/slides/plenaryt-3.pdf>

14. P. Gill, YouTube workload characterization, Master Thesis, Univ. of Calgary, Canada (2008) <pages.cpsc.ucalgary.ca/ psessini/papers/pgill_thesis.pdf>
15. G. Hasslinger, ISP Platforms under a heavy peer-to-peer workload, Proc. Peer-to-Peer Systems and Applications, Eds.: R. Steinmetz and K. Wehrle, Springer LNCS 3485 (2005) 369–382
16. G. Hasslinger, F. Guillemin, J. Ferreira and U. Halldorsson, The impact of peer-to-peer networking on network operators and Internet service providers, Eurescom study report P1553 (2005)
<www.eurescom.eu/public/projects/P1500-series/p1553>
17. G. Hasslinger and S. Kempken, Applying random walks in structured and self-organizing networks: Evaluation by transient analysis, PIK – Special Issue on Self-Organizing Networks, Saur Verlag, Vol. 31 (2008) 17–23
18. G. Hasslinger, J. Mende, R. Geib, T. Beckhaus and F. Hartleb: Measurement and characteristics of traffic in broadband access networks, Proc. 20. Internat. Teletraffic Congress, Ottawa, Canada, Springer, LNCS 4516 (2007) 998–1010
19. G. Hasslinger, S. Schnitter and M. Franzke, The Efficiency of Traffic Engineering with Regard to Failure Resilience, Telecommunication Systems Vol. 29/2, Springer (2005) 109–130
20. Y. Hu, D. Chiu and J. Lui, Application identification based on network behavioral profiles, Proc. 16. IEEE Workshop on Quality of Service (IWQoS'08) Twente / Enschede, The Nederlands (2008) 239–248
21. Internet Engineering Task Force (IETF), ALTO BoF: Application layer traffic optimization, <www.ietf.org/proceedings/08jul/minutes/alto.txt> and <alto.tilab.com>
22. J. Ledlie, P. Gardner and M. Seltzer, Network coordinates in the wild, Proc. USENIX Conf. (2007) 299–311
23. The Minnesota Internet traffic studies (MINTS)
<http://www.dtc.umn.edu/mints/references.html>
continuously updated until 2008
24. A. Odlyzko, Internet traffic growth: Sources and implications, Proc. SPIE Vol. 5247 (2003) 1–15
25. C. Rossenhövel, P2P filters ready for Internet prime time?
<www.internetevolution.com/document.asp?doc_id=148803>
26. O. Saleh and M. Hefeeda, Modeling and caching of P2P traffic, Proc. IEEE Internat. Conf. on Network Protocols (2006)
27. J. Schroeder-Bernhardi, Analysis of the communication and traffic in P2P networks including web caches, Master Thesis, KOM-D-260, Univ. of Darmstadt, Germany (2006)
28. H.M. Sigurdsson, U.R. Halldorsson and G. Hasslinger: Potentials and Challenges of Peer-to-Peer Based Content Distribution, Telematics and Informatics, Elsevier, Vol. 24 (2007) 348–365
29. A.-J. Su, D.R. Choffnes, A. Kuzmanovic and F.E. Bustamante, Drafting behind Akamai, Proc. ACM SIGCOMM, Pisa, Italy (2006)

Part XII
Advanced P2P Computing and Networking

A Formal Architectural Model for Peer-to-Peer Systems

Lu Yan

Abstract For complex software systems, a central design concern is system architecture. The formal approach to architectural specification and refinement provides an effective way to ensure the correctness of those complex designs. In this chapter, we apply formal methods in the development process of a typical peer-to-peer system and demonstrate a stepwise paradigm for specifying, refining and implementing such kind of systems. The architectural considerations, formal specification, and possible refinement directions are discussed in details.

1 Introduction

Most current distributed systems follow the client-server paradigm in which a single server stores information and distributes it to clients upon their requests. Peer-to-peer systems, which consider that nodes are equal for sharing information, on the other hand, follow a paradigm in which each node can store information of its own and establish direct connections with another node to share information [1]. In this way, the peer-to-peer systems offer attractive advantages like enhanced load balancing, dynamic information repositories, redundancy and fault tolerance, content-based addressing and improved searching over the traditional client-server systems [2].

Because of the increasingly popularity of peer-to-peer applications (e.g., content distribution [3], media streaming [4], resource sharing [5], social networking [6], grid computing [7], etc.) and the advantages of the peer-to-peer paradigm, we are

Lu Yan
School of Computer Science, University of Hertfordshire, Hatfield, Hertfordshire AL10 9AB, UK,
e-mail: lu.yan@ieee.org

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_47, © Springer Science+Business Media, LLC 2010

motivated to conduct a study of peer-to-peer systems and achieve a way to develop such systems.

In this paper, we show how to design peer-to-peer systems within the action systems framework by combining UML diagrams. We present our approach via a case study of stepwise development of a Gnutella-like peer-to-peer system. We start by briefly describing the action systems framework to the required extent in Section 2. In Section 3 we give an initial specification of the Gnutella system. An abstract action system specification is derived in Section 4. In Section 5 we analyze and refine the system specification introduced in the previous section. We end in Section 6 with concluding remarks and Section 7 with future research directions.

2 Action Systems

The action systems framework is a formal and rigorous approach to specifying distributed systems [8, 9]. The design and reasoning about action systems are carried out with refinement calculus [10].

In this section we will introduce OO-action systems [11], an object-oriented extension of action systems which we select as a foundation to work on. An OO-action system consists of a finite set of classes, each class specifying the behavior of objects that are dynamically created and executed in parallel.

2.1 Actions

We will consider a fixed set *Attr* of attributes (variables) and assume that each attribute is associated with a nonempty set of *values*. Also, we consider a set *Act* of actions defined by the following grammar

$$A \ ::= \ abort | skip | x := v | x :\in V | b \rightarrow A | \{b\} | A;A | A \, [\!] \, A.$$

Here x is a list of attributes, v a list of values, V a nonempty set of values, b a predicate over attributes. Intuitively, *abort* is the action which always deadlocks, *skip* is a stuttering action, $x := v$ is a multiple assignment, $x :\in V$ is a random assignment, $b \rightarrow A$ is a guard of an action, $\{b\}$ is an assertion, $A_1;A_2$ is the sequential composition of the actions A_1 and A_2, and $A_1 \, [\!] \, A_2$ is the nondeterministic choice between the action A_1 and A_2.

The *guard* of an action is defined in a standard way using weakest preconditions [12]

$$g(A) \;=\; \neg wp(A, \mathit{false}).$$

The action A is said to be enabled when the guard is true.

2.2 Classes and Objects

Let *CName* be a fixed set of class names and *OName* a set of valid names for objects. We will also consider a fixed set of object variables *OVar* assumed to be disjoint from *Attr*. The only valid values for object variables are the names for objects in *OName*. The set of object actions *OAct* is defined by extending the grammar of actions as follows

$$\begin{aligned} O ::= {} & A | n := o | new(c) | n := new(c) \\ & | p | n.m | self.m | super.m | O;O | O \mathbin{[\!]} O. \end{aligned}$$

Here $A \in Act$, n is an object variable, o is either an object name or the constants *self* or *super* (all three possibly resulting from the evaluation of an expression), c is a class name, p a procedure name, and m is a method name. Intuitively, $n := o$ stores the object name o into the object variable n, $new(c)$ creates a new object instance of the class c, $n := new(c)$ assigns the name of the newly created instance of the class c to the object variable n, p is a procedure call, $n.m$ is a call of the method m of the object the name of which is stored in the object variable n, $self.m$ is a call of the method m declared in the same object, and $super.m$ is a call of the method m declared in the object that created the calling object. Note that method calls are always prefixed by an object variable or by the constant *self* or *super*.

We define the guard $gd(O)$ of an object action O to be the guard of the action in *Act* obtained by substituting every atomic object action of O with the action *skip*, where an atomic object action is

$$A, n := o, new(c), n := new(c), p, n.m, self.m, super.m.$$

The resulting statement is an action in *Act* and hence its guard is well defined.

A *class* is a pair $< c, \mathscr{C} >$, where $c \in CName$ is the *name* of the class and $\mathscr{C}$ is its *body*, that is, a statement of the form

$$\begin{aligned} \mathscr{C} \;=\; |[\; & \textbf{attr} \;\; y^{\cdot} := y0; x := x0 \\ & \textbf{obj} \;\; n \\ & \textbf{meth} \; m_1 = M_1; \cdots ; m_h = M_h \\ & \textbf{proc} \;\; p_1 = P_1; \cdots ; p_k = P_k \\ & \textbf{do} \; O \; \textbf{od} \\ ]| & \end{aligned}$$

A class body consists of an object action O and of four declaration sections. In the attribute declaration the *shared attributes* in the list y, marked with an asterisk $\cdot$, describe the variables to be shared among all active objects. Therefore they can be used by instances of the class $\mathscr{C}$ and also by object instances of other classes. Initially they get values $y0$. The *local attributes* in the list x describe variables that are local to an object instance of the class, meaning that they can only be used by the instance itself. The variables are initialized to the value $x0$.

The list n of *object variables* describes a special kind of variables local to an object instance of the class. They contain names of objects and are used for calling methods of other objects. We assume that the lists x, y and n are pairwise disjoint.

A *method* $m_i = M_i$ describes a procedure of an object instance of the class. They can be called by actions of the objects themselves or by actions of another object instance of possibly another class. A method consists of a method name m and an object action M.

A *procedure* $p_i = P_i$ describes a procedure that is local to the object instances of the class. It can be called only by actions of the object itself. Like a method, it consists of a procedure name p and an object action forming the body P.

The *class body* is a description of the actions to be executed repeatedly when the object instance of the class is activated. It can refer to attributes which are declared to be shared in another class, and to the object variables and the local attributes declared within the class itself. It can contain procedure calls only to procedures declared in the class and method calls of the form *n.m* or *super.m* to methods declared in other classes. Method calls *self.m* are allowed only if m is a method declared in the same class. As for action systems, the execution of an object action is atomic.

2.3 OO-action System

An *OO-action system OO* consists of a finite set of classes

$$OO = \{< c_1, \mathscr{C}_1 >, \cdots, < c_n, \mathscr{C}_n >\}$$

such that the shared attributes declared in each $\mathscr{C}_i$ are distinct and actions in each $\mathscr{C}_i$ or bodies of methods and procedures declared in each $\mathscr{C}_i$ do not contain *new* statements referring to class names not used by classes in OO. Local attributes, object variables, methods, and procedures are not required to be distinct.

There are some classes in OO, marked with an asterisk $\cdot$. Execution starts by the creation of one object instance of each of these classes. Each object, when created, chooses enabled actions and executes them. Actions operating on disjoint sets of local and shared attributes, and object variables can be executed in parallel. They can also create other objects. Actions of different active objects can be executed in parallel if they are operating on disjoint sets of shared attributes. Objects interact by means of the shared attributes and by executing methods of other objects.

3 Initial Specification of the Gnutella System

Gnutella is a decentralized peer-to-peer file-sharing model which enables file sharing without using servers [13]. Unlike a centralized server network, the Gnutella network does not use a central server to keep track of all user files. To share files using the Gnutella model, a user starts with a networked computer A with a Gnutella *servent*, which works both as a server and a client. Computer A will connect to another Gnutella-networked computer B and then announce that it is *alive* to computer B. B will in turn announce to all its neighbours C, D, E, and F that A is alive. Those computers will recursively continue this pattern and announce to their neighbours that computer A is alive. Once computer A has announced that it is alive to the rest of the members of the peer-to-peer network, it can then search the contents of the shared directories of the peer-to-peer network.

Search requests are transmitted over the Gnutella network in a decentralized manner. One computer sends a search request to its neighbours, which in turn pass that request along to their neighbours, and so on. Figure 1 illustrates this model. The search request from computer A will be transmitted to all members of the peer-to-peer network, starting with computer B, then to C, D, E, F, which will in turn send the request to their neighbours, and so forth. If one of the computers in the peer-to-peer network, for example, computer F, has a match, it transmits the file information (name, location, etc.) back through all the computers in the pathway towards A (via computer B in this case). Computer A will then be able to open a direct connection with computer F and will be able to download that file directly from computer F.

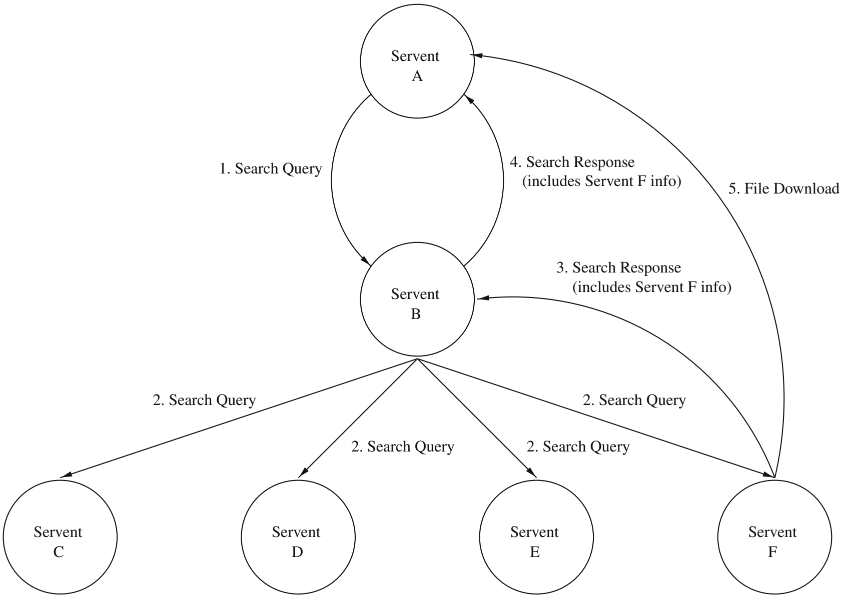


Fig. 1 Gnutella peer-to-peer model

4 Action System Specification of the Gnutella System

When taking a step back, it is seen that the Gnutella system enables at least the following functionalities:

1. Servent can easily join and leave the peer-to-peer network.
2. Servent can publish its content to the shared directories of the peer-to-peer network.
3. Servent can search for and download files from the shared directories of the peer-to-peer network using keywords.

Based on the simple descriptions above, we can identify that servent should provide three basic services, *connect service*, *lookup service* and *download service*, as shown in Fig. 2. From this diagram we divide the system into components and derive a component-based structure of servent in Fig. 3.

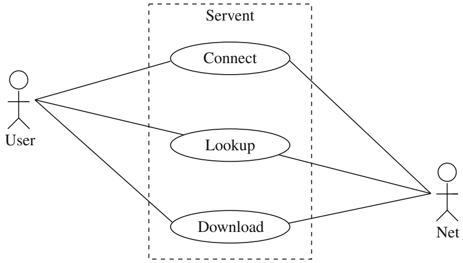


Fig. 2 Use case diagram of servent

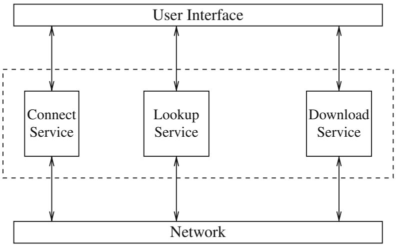


Fig. 3 Structure diagram of servent

The statechart diagram Fig. 4 shows the joint behaviour of servent. Each state is described by a set of attributes. We give unique preconditions for entering each state. Table 1 shows preconditions and invariants for every state of the servent.

An interesting issue to notice is that downloads can be initiated in two ways, i.e., either from a search result or by directly specifying target information. This

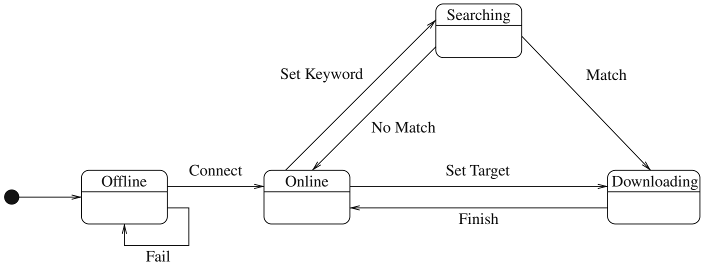


Fig. 4 Statechart diagram of servent

Table 1 Preconditions and invariants for states

State	Precondition	Invariant
Offline	$\neg$connected $\wedge$ keyword = ϕ $\wedge$ target = ϕ	keyword = ϕ $\wedge$ target = ϕ
Online	connected $\wedge$ keyword = ϕ $\wedge$ target = ϕ	connected
Searching	connected $\wedge$ keyword $\neq \phi$ $\wedge$ target = ϕ	connected $\wedge$ target = ϕ
Downloading	connected $\wedge$ keyword = ϕ $\wedge$ target $\neq \phi$	connected $\wedge$ keyword = ϕ

design is reasonable because we do not always need to search the peer-to-peer network to get wanted files. In some cases, name and location information of files is already available. For example, file exchanges between two friends, who have already known each other's IP and shared contents. Take this into consideration, we provide two ways to initiate downloads.

The first version of action system specification of servent can be derived directly from Fig. 4 and Table 1.

$$\{< GnutellaServent, GS >, \cdots\}$$

where the class body in Table 2 consists of attribute declaration, initialisation and a loop of actions which are chosen for execution in a non-deterministic fashion when enabled. Each action is of the form $g \rightarrow S$ where g is the guard and S is a statement to be executed when the guard evaluates to $true$. Here *connected* is a boolean variable; *keyword* is the search criteria; *target* is the location information of shared resources in format *filename@IP*.

The next step is to apply the design in Fig. 3 to our initial specification, which results in three more classes *ConnectService, LookupService* and *DownloadService*. Now the system consists of a set of classes $< c, C >$ where c is the name of the class and C is its body. On the top level, we have components of servent

$$\{< GnutellaServent, GS >^{'}, < ConnectService, Cs >,$$
$$< LookupService, Ls >, < DownloadService, Ds >, \cdots\}$$

Table 2 Initial specification of servent

```
$GS$ = |[ attr $connected^{\cdot}; keyword^{\cdot}; target^{\cdot}; state := Offline$
         do
              $state = Offline \wedge connected \rightarrow$
                   $state := Online$
              [] $state = Online \wedge keyword \neq \phi \rightarrow$
                   $state := Searching$
              [] $state = Online \wedge target \neq \phi \rightarrow$
                   $state := Downloading$
              [] $state = Searching \wedge target = \phi \rightarrow$
                   $keyword := \phi; state := Online$
              [] $state = Searching \wedge target \neq \phi \rightarrow$
                   $keyword := \phi; state := Downloading$
              [] $state = Downloading \rightarrow$
                   $target := \phi; state := Online$
         od
       ]|
```

The class $< GnutellaServent, GS >$, marked with an asterisk $\cdot$, is the root class. At execution time one object of this class is initially created and this in turn creates other objects by executing *new* statements.

Let us look at the actions in Table 2. We can refine them according to service groups. For example, action

$$state = Offline \wedge connected \rightarrow state := Online$$

where *Offline* and *Online* are defined in Table 1, can be refined into action

$$\neg connected \rightarrow connected := c.Connect(\,)$$

In this chapter, however, we skip refinement details here because we do not want to go into details of semantics of action systems nor refinement rules of refinement calculus [10].

The body of the refined specification of servent is described in Table 3. It models a servent that provides three basic services (*ConnectService, LookupService* and *DownloadService*). When it connects itself to the peer-to-peer network, users can search the network via *SetKeyword* method and then download files from the search result. Or alternatively, users can directly give *target* information via *SetTarget* method to download files.

5 Refining Gnutella Servent

Ultimately, we need to derive an implementable specification for each service in the Guntella servent. Hence, every service component should be refined. We notice

Table 3 Refined specification of servent

```
$GS = |[$ attr $connected := false; keyword := \phi; target := \phi$
          obj  $c : ConnectService; l : LookupService;$
               $d : DownloadService$
          meth $SetKeyword(k) = keyword := k;$
               $SetTarget(t) = target := t$
          init $c := new(ConnectService);$
               $l := new(LookupService);$
               $d := new(DownloadService)$
          do
                $\neg connected \rightarrow connected := c.Connect(\,)$
                $[]\ connected \wedge keyword \neq \phi \rightarrow$
                     $target := l.Search(keyword); keyword := \phi$
                $[]\ connected \wedge target \neq \phi \rightarrow$
                     $d.Download(target); target := \phi$
          od
       $]|$
```

ConnectService and *LookupService* share a common functionality that enables appropriate message routing. It is reasonable to introduce a new component *Router* to the system as depicted in Fig. 5. This component will be in charge of routing all the incoming and outgoing messages of the servent. For *DownloadService*, we introduce a new component *FileRepository* in Fig. 5. It will act as a resource exchanger between servent and network.

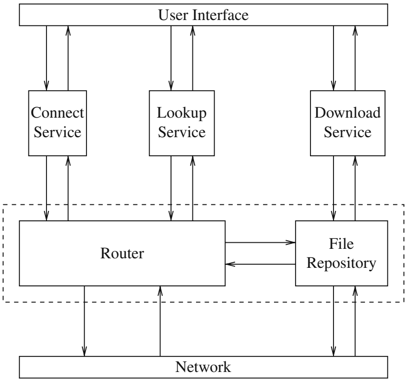


Fig. 5 Schematic diagram of servent

5.1 Refining ConnectService

We start by considering *ConnectService* first. A Gnutella servent connects itself to the peer-to-peer network by establishing a connection with another servent currently on the network, and this kind of connection is passed around recursively. In order to model the communication between servents, we define a set of descriptors and inter-servent descriptor exchange rules as follows:

Ping Used to actively discover hosts on the network. A servent receiving a Ping descriptor is expected to respond with one or more Pong descriptors.

Pong The response to a Ping. Includes the address of a connected Gnutella servent and information regarding the amount of data it is making available to the network.

Furthermore, we need to define the format of Ping descriptor and Pong descriptor. We use the message format in Table 4, where *DescriptorID* is a string uniquely identifying the descriptors on the network. TTL stands for *Time To Live*, which is the number of times the descriptor will be forwarded by servent before it is removed from the network. The TTL is the only mechanism for expiring descriptors on the network. Each servent will decrement the TTL before passing it on to another servent. When the TTL reaches 0, the descriptor will no longer be forwarded. The information carried by the message is encoded in *info*, whose format depends on the variable *type*.

Table 4 Message format

$$
\begin{array}{lll}
Msg = |[& \textbf{attr} & descriptorID; TTL; type; info \\
 & \textbf{meth} & Transmit(\,) = TTL > 0 \rightarrow TTL := TTL - 1 \\
 & \textbf{init} & t \in \{Ping, Pong\} \rightarrow \\
 & & \quad Msg(t) = (descriptorID := \_unique\_ID\_; \\
 & & \quad TTL := \_max\_TTL\_; type := t; info := \_info\_) \\
\quad]| & &
\end{array}
$$

The peer-to-peer nature of the Gnutella network requires servents to route network traffic appropriately. Intuitively, a servent will forward incoming Ping descriptors to all its directly connected servents. Pong descriptors may only be sent along the same path that carried the incoming Ping descriptor as shown in Fig. 6. This ensures that only servents that routed the Ping descriptors will see the Pong descriptor in response. A servent that receives a Pong descriptor with $descriptorID = n$, but has not seen a Ping descriptor with $descriptorID = n$ should remove the Pong descriptor from the network.

The above routing rules can be illustrated in the statechart diagram Fig. 7. Using the same techniques as in the previous section, we can translate the diagram into action system specification and further refine it into Table 5.

The specification of Ping-Pong router models a router that can route Ping-Pong traffic appropriately. When the router is initiating, it connects itself to the peer-to-peer network by sending Ping descriptors to other peers. After initiation, it continues receiving $\_incoming\_message\_$ and replying with apporiate $\_outgoing\_message\_$. Here *descriptorDB* is a set storing *descriptorID* information; *peers* is a set storing its directly and indirectly connected servents information; and $\_this\_IP\_$ is the IP

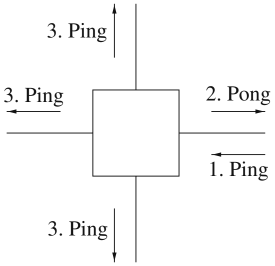


Fig. 6 Ping-pong routing

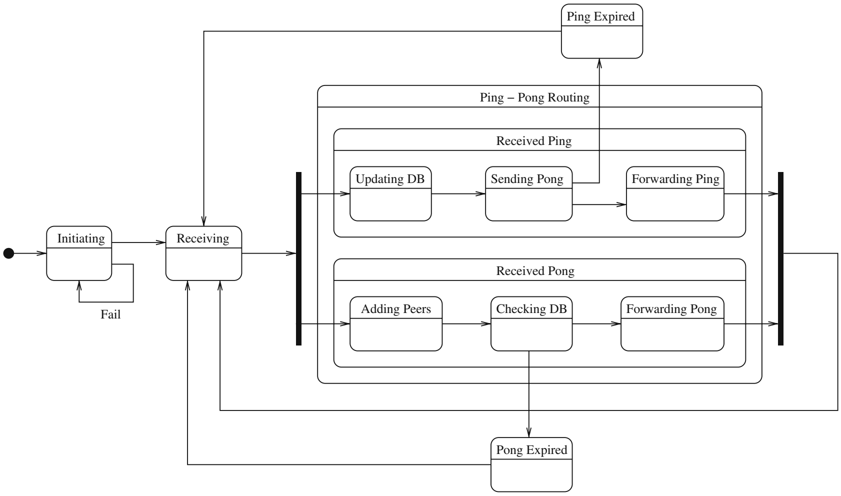


Fig. 7 Statechart diagram of Ping-pong routing rules

address of the responding servent. The sequence of a connect session is summarized in Fig. 8.

In order to reuse the specification in Table 3, we will specify *ConnectService* without making any changes in its interface. The specification is shown in Table 6. When *ConnectService* is initiating, an instance of *Router* is created. Then it keeps checking state variable *connected* in the router and passing the status to servent.

5.2 Refining LookupService

When we think about *LookupService*, we follow almost the same paradigm as *ConnectService* to specify this component. A Gnutella servent starts a search request by broadcasting a *Query* message through the peer-to-peer network. Upon receiving a search request, the servent checks if any local stored files match the query and sends

Table 5 Specification of Ping-pong router

```
$Rc = |[$ attr $connected := false; descriptorDB := \phi; peers := \phi$
          obj  $receivedMsg : Msg; newMsg : Msg$
          meth $ReceiveMsg(\,) = receivedMsg := \_incoming\_message\_;$
               $SendPing(\,) = (newMsg := new(Msg(Ping));$
                   $\_outgoing\_message\_ := newMsg);$
               $SendPong(\,) = (newMsg := new(Msg(Pong));$
                   $newMsg.info.IP := \_this\_IP\_;$
                   $\_outgoing\_message\_ := newMsg);$
               $ForwardMsg(m) = (m.TTL > 0 \rightarrow$
                   $m.Transmit(\,); \_outgoing\_message\_ := m)$
          init $SendPing(\,)$
          do
               $|[\ peers \neq \phi \rightarrow connected := true$
               $[]\ \neg connected \rightarrow SendPing(\,)\ ]|$
               $/\!/$
               $|[\ true \rightarrow$
                   $ReceiveMsg(\,);$
                   if $receivedMsg.type = Ping \rightarrow$
                       $descriptorDB := descriptorDB \cup$
                       $receivedMsg.descriptorID;$
                       $SendPong(\,);$
                       $ForwardMsg(receivedMsg)$
                   $[]\ receivedMsg.type = Pong \rightarrow$
                       $peers := peers \cup receivedMsg.info.IP;$
                       $receivedMsg.descriptorID \in descriptorDB \rightarrow$
                           $ForwardMsg(receivedMsg)$
                   fi
               $]|$
          od
      $]|$
```

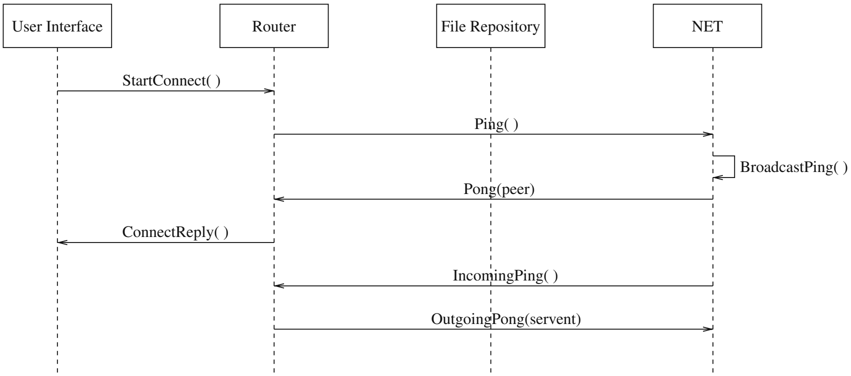


Fig. 8 Sequence diagram of a connect session

Table 6 Specification of *ConnectService*

$$
\begin{array}{rll}
Cs = |[& \textbf{attr} & connected := false \\
 & \textbf{obj} & r : Router \\
 & \textbf{meth} & Connect(\,) = (connected := r.connected) \\
 & \textbf{init} & r := new(Router) \\
 ]| & &
\end{array}
$$

a *QueryHit* message back. We use following descriptors and routing rules to model the communication between servents:

Query The primary mechanism for searching the distributed network. A servent receiving a Query descriptor will respond with a QueryHit if a match is found against its local data set.

QueryHit The response to a Query. This descriptor provides the recipient with enough information to acquire the data matching the corresponding Query.

The message format in Table 4 has to be revised to adopt the new descriptors. The message *type* now includes *Ping, Pong, Query* and *QueryHit*, so a minor change is made in Table 7.

Table 7 Refined message format

$$
\begin{array}{rll}
Msg = |[& \textbf{attr} & descriptorID; TTL; type; info \\
 & \textbf{meth} & Transmit(\,) = TTL > 0 \rightarrow TTL := TTL - 1 \\
 & \textbf{init} & t \in \{Ping, Pong, Query, QueryHit\} \rightarrow \\
 & & \quad Msg(t) = (descriptorID := \_unique\_ID\_; \\
 & & \quad TTL := \_max\_TTL\_; type := t; info := \_info\_) \\
 ]| & &
\end{array}
$$

The routing rules for Query – QueryHit traffic are similar to the rules for Ping-Pong traffic. A servent will forward incoming Query descriptors to all its directly connected servents. QueryHit descriptors may only be sent along the same path that carried the incoming Query descriptor as shown in Fig. 9. This ensures that only those servents that routed the query descriptors will see the QueryHit descriptor in response. A servent that receives a QueryHit descriptor with $descriptorID = n$, but has not seen a Query descriptor with $descriptorID = n$ should remove the QueryHit descriptor from the network.

Like the previous section, we first draw a statechart diagram for the Query – QueryHit routing rules. Then we translate it into action system specification and further refine it.

In Table 8 we have the body of Query – QueryHit router specification, which models a router that is in charge of routing Query – QueryHit traffic appropriately.

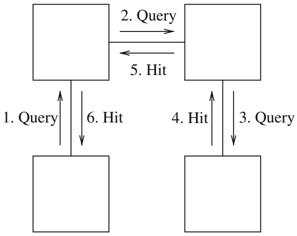


Fig. 9 Query – QueryHit routing

Like a Ping-Pong router, it keeps receiving *_incoming_message_* and replying with apporiate *_outgoing_message_*. Here *descriptorDB* is a set storing *descriptorID* information; *myKeyword* is a string storing search criteria; *cKeyword* is a string storing comparison criteria; *filename* is a string storing destination filename; *target* is the shared resource location information in format *filename@IP*; and *f* is an object of class *FileRepository* which enables local file search service via *Has* and *Find* methods. Details of class *FileRepository* will be elaborated in the next section.

The Query – QueryHit router provides searching function via method *SetKeyword*. Once a keyword is set, a Query descriptor carrying search criteria is generated and broadcasted in the peer-to-peer network via method *SendQuery*. In the mean time, the router keeps receiving Query and QueryHit descriptors. For an incoming Query descriptor, a query request is passed to *FileRepository*. According to the search result, a QueryHit descriptor is sent back in response via method *SendQueryHit* if a match is found, otherwise the Query descriptor is further forwarded via method *ForwardMsg*. Upon receiving a QueryHit descriptor, it checks its keyword field, and then sets *target* information to complete the search session. We summarize the sequence of a search session in Fig. 10.

Now we specify *LookupService* with emphasis on specification reuse. The result is shown in Table 9. When *LookupService* is initiated, an instance of *Router* is created. It provides *Search* method via calling *SetKeyword* method in the router, and then returning the search result to servent.

Until now we have two action systems, *Rc* modeling Ping-Pong routing rules and *Rl* modeling Query – QueryHit routing rules. We notice the two action systems actually model different aspects of a full router. Furthermore, we can compose the two action systems together using *prioritising composition* [14] to derive the action system specification of a full router

$$R = |[\, Rc \,/\!/\, Rl \,]|$$

where on the higher level, we have components of the router

$$\{< Router, R >, < PingPongRouter, Rc >, < QueryRouter, Rl >\}$$

Table 8 Specification of Query – QueryHit router

```
$Rl = |[$ attr $descriptorDB := \phi; myKeyword := \phi;$
               $cKeyword := \phi; filename := \phi; target := \phi$
          obj  $receivedMsg : Msg; newMsg : Msg; f : FileRepository$
          meth $SetKeyword(k) = myKeyword := k;$
               $ReceiveMsg(\,) = receivedMsg := \_incoming\_message\_;$
               $SendQuery(\,) = (newMsg := new(Msg(Query));$
                   $newMsg.info.keyword := myKeyword;$
                   $cKeyword := myKeyword; target := \phi;$
                   $\_outgoing\_message\_ := newMsg);$
               $SendQueryHit(\,) = (newMsg := new(Msg(QueryHit));$
                   $newMsg.info.keyword := receivedMsg.info.keyword;$
                   $newMsg.info.filename := filename;$
                   $newMsg.info.IP := \_this\_IP\_;$
                   $\_outgoing\_message\_ := newMsg);$
               $ForwardMsg(m) = (m.TTL > 0 \rightarrow$
                   $m.Transmit(\,); \_outgoing\_message\_ := m)$
          do
               $|[\ myKeyword \neq \phi \rightarrow SendQuery(\,); myKeyword = \phi\ ]|$
               $/\!/$
               $|[\ true \rightarrow$
                   $ReceiveMsg(\,);$
                   if $receivedMsg.type = Query \rightarrow$
                       $descriptorDB := descriptorDB \cup$
                       $receivedMsg.descriptorID;$
                       if $f.Has(receivedMsg.info.keyword) \rightarrow$
                           $filename := f.Find(receivedMsg.info.keyword);$
                           $SendQueryHit(\,)$
                       $[]\ \neg f.Has(receivedMsg.info.keyword) \rightarrow$
                           $ForwardMsg(receivedMsg)$
                       fi
                   $[]\ receivedMsg.type = QueryHit \rightarrow$
                       if $receivedMsg.info.keyword = cKeyword \rightarrow$
                           $target := receivedMsg.info.filename@$
                           $receivedMsg.info.IP; cKeyword := \phi$
                       $[]\ receivedMsg.info.keyword \neq cKeyword \wedge$
                       $receivedMsg.descriptorID \in descriptorDB \rightarrow$
                               $ForwardMsg(receivedMsg)$
                       fi
                   fi
               $]|$
          od
      $]|$
```

Table 9 Specification of *LookupService*

```
$Ls = |[$ attr $target := \phi$
          obj  $r : Router$
          meth $Search(k) = (r.SetKeyword(k); target := r.target)$
          init $r := new(Router)$
      $]|$
```

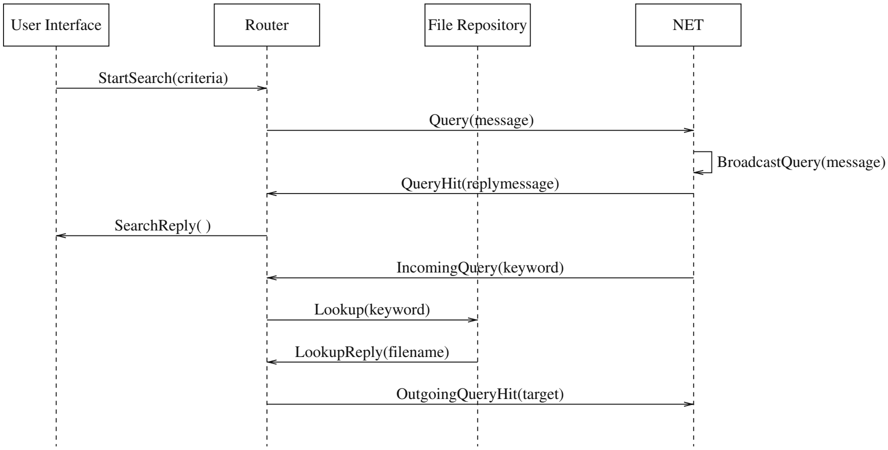


Fig. 10 Sequence diagram of a search session

5.3 *Refining DownloadService*

DownloadService is relatively simple compared to *ConnectService* and *LookupService*. The primary function of this component is to enable a servent to download files from other servents. Once a servent receives a QueryHit descriptor, it may initiate the direct download of one of the files described by the descriptor's result set. Or alternatively, users can initiate the download directly by giving complete *target* information. Files are downloaded out-of-network, i.e., a direct connection between the source and target servent is established in order to perform the data transfer. File data is never transferred over the peer-to-peer network.

Additionally, this component provides the local file query function for other servents. It should be in charge of a local file database which provides data services like *add, delete, update, query* and *refresh* etc. Moreover, it should take full control of local files. Hence we introduce a new component *FileRepository* in Table 10, which will satisfy the above requirements for *DownloadService*. First of all, we provide *SetTarget* method to enable file downloads. To make things simple, we assume that *fileDB* is simply a set of relations $\{key\} \rightarrow \{file\}$. We use relation notations [15] *dom* and *ran* for domain and range operations, and $\lhd$ as a domain restriction operator, defined by $S \lhd r = \{x,y | x \mapsto y \in r \wedge x \in S\}$. For incoming Query descriptors, *Has* and *Find* methods are provided to enable local file searches.

Given *target* information, download action is enabled and servent initiates a download. A download request is sent to the target servent, and then a file is downloaded via HTTP protocol. Afterwards, *fileDB* is refreshed in order to reflect the change of adding new files to the repository. The sequence of a download session is summarized in Fig. 11.

Table 10 Specification of file repository

```
$F = |[$ **attr** $fileDB := \_fileDB\_; filename := \phi; target := \phi$
         **meth** $SetTarget(t) = (target := t);$
                  $Has(key) = (\{key\} \in dom(fileDB));$
                  $Find(key) = (filename := file \wedge \{file\} \in ran(\{key\} \lhd fileDB))$
         **do**
                  $target \neq \phi \rightarrow$
                      $HTTP(target);$
                      $target := \phi;$
                      $Refresh(fileDB)$
         **od**
     $]|$
```

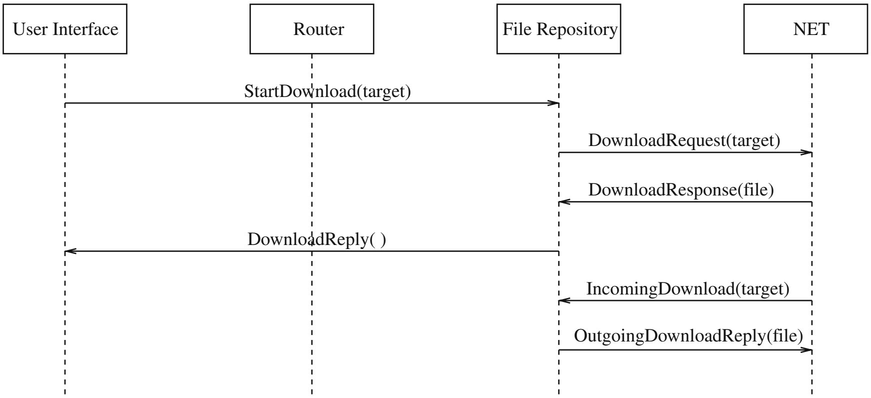


Fig. 11 Sequence diagram of a download session

The last step is to specify *DownloadService*. From the result in Table 11, we can see that when *DownloadService* is initiating, an instance of *FileRepository* is created. It enables *Download* by calling *SetTarget* method in *FileRepository*.

Table 11 Specification of *DownloadService*

```
$Ds = |[$ **obj**  $f : FileRepository$
          **meth** $Download(t) = f.SetKeyword(t)$
          **init** $f := new(FileRepository)$
      $]|$
```

At this stage of the design, we finally have a complete set of classes which are refinement results from the initial specification of servent as follows

$$\begin{gathered}\{< GnutellaServent, GS >^{\cdot}, < ConnectService, Cs >,\\ < LookupService, Ls >, < DownloadService, Ds >,\\ < PingPongRouter, Rc >, < QueryRouter, Rl >,\\ < Router, R >, < FileRepository, F >, < Message, Msg >\}\end{gathered}$$

6 Concluding Remarks

Our experience shows that it is beneficial to combine informal methods like UML and formal methods like action systems together in the development of peer-to-peer systems. In the early stage, we try to catch the characteristic of the system using use case diagrams and statechart diagrams. Then formal specification in action systems framework is derived by further studying and elaborating details of these diagrams. In the later stage, sequence diagrams are used to graphically clarify the structure of the refined action system specification.

Peer-to-peer computing has emerged as one of the most innovation rich areas in computer networking. Millions of users now participate in these systems and the user bases are spreading like wild-fire. What is interesting about P2P is that although it emerged out of user community but it is increasingly finding its base on rich foundation of computing [16].

Various aspects of peer-to-peer systems have been studies in the past. However, few issues were covered on the software architectural considerations of those systems, especially how to model, formalize, and develop software systems based on the peer-to-peer paradigm. In this chapter, we present a detailed case study on the foundation of peer-to-peer software engineering, which complements the existing peer-to-peer research literature perfectly.

7 Future Directions

Unlike the traditional client-server model, a peer in a P2P network plays the role of both client and server. It acts as a client when it consumes resources and services from other peers and and as a server when it provides those to others.

The P2P network is an overlay network with an application-level routing scheme, typically with a Distributed Hash Table (DHT). It is a high-level routing scheme suitable for communication middleware and web services. Therefore, one possible direction in the formal P2P architecture research is to design and verify those large scale complex middleware and web services. The P2P paradigm would greatly improve the system scalability with a better messaging and load-balancing solution. The formal aspect of our approach would ease the simulation of those systems, so that undesirable aggregated behaviors such as deadlock, livelock, node overload etc. can be identified and prevented at the very early stage in the design process.

Another promising direction lies in the formal model itself. The specification of the P2P architectural model presented in this chapter is part of the Model-Driven Architecture (MDA) approach. The underpinned philosophy is a stepwise software development process. First we define system functionality using a Platform-Independent Model (PIM). Then PIM is translated into one or more Platform-Specific Models (PSM) that computers can run. One of the main advantages of MDA is to separate design from architecture. As the concepts and technologies used to realize designs and the concepts and technologies used to realize architectures change/evolve at their own pace, decoupling them allows developers choose the best/most-fitting in both domains.

References

1. E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim: *A survey and comparison of peer-to-peer overlay network schemes.* IEEE Communications Surveys & Tutorials. (2005) 7: 72–93.
2. S. Androutsellis-Theotokis, D. Spinellis: *A survey of peer-to-peer content distribution technologies.* ACM Computing Surveys. 36(4):335C371, December 2004.
3. C. Huang, J. Li, K. W. Ross: *Peer-Assisted VoD: Making Internet Video Distribution Cheap.* Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS'07), Washington, USA, Feb. 2007.
4. S. Guha, N. Daswani, R. Jain: *An Experimental Study of the Skype Peer-to-Peer VoIP System.* Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS'06), Santa Barbara, CA, USA, Feb. 2006.
5. E. Anceaume, M. Gradinariu, A. Ravoaja: *Incentives for P2P Fair Resource Sharing.* Proceedings of the 5th IEEE International Conference on Peer-to-Peer Computing (P2P'05), Konstanz, Germany, 2005.
6. J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J.Yang, A.Iosup, D.Epema, M.Reinders, M. van Steen, H.Sips: *Tribler: A Social-Based Peer-to-Peer System.* Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS'06), Santa Barbara, CA, USA, Feb. 2006.
7. J. Cao, F. B. Liu, C.-Z. Xu: *P2PGrid: integrating P2P networks into the Grid environment.* Concurrency and Computation: Practice and Experience. Volume 19, Issue 7, pp. 1023–1046, 2006.
8. R.J.R. Back and K. Sere: *From Action Systems to Modular Systems.* Software – Concepts and Tools. (1996) 17: 26–39.
9. R.J.R. Back, A.J. Martin and K.Sere: *Specifying the Caltech asynchronous microprocessor.* Science of Computer Programming. (1996) 26: 79–97.
10. R.J. Back and J. Wright: *Refinement Calculus: A Systematic Introduction.* Graduate Texts in Computer Science, Springer-Verlag, 1998.
11. M. Bonsangue, J.N. Kok and K. Sere: *An approach to object-orientation in action systems.* Proceedings of Mathematics of Program Construction (MPC'98), Marstrand, Sweden, June 1998.
12. E.W. Dijkstra: *A Discipline of Programming.* Prentice-Hall International, 1976.

13. S. Oeztunali, S. Rusitschka, A. Southall: *Multilayer Gnutella ł P2P Resource Sharing with an Efficient Flexible Multi-Keyword Search Facility.* Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS'07), Washington, USA, Feb. 2007.
14. E. Sekerinski and K. Sere: *A Theory of Prioritising Composition.* The Computer Journal, Volume 39, Issue 8, pp. 701–712.
15. E. Sekerinski and K. Sere (Eds): *Program Development by Refinement: Case Studies Using the B Method.* Springer-Verlag, 1999.
16. J. I. Khan, A. Wierzbicki (Ed.): *Special Issue: Foundation of Peer-to-Peer Computing.* Elsevier Journal of Computer Communication, Volume 31, Issue 2, Feb. 2008.

P2P Approach for Web Services Publishing and Discovery

Mohammad Towhidul Islam, Mursalin Akon, and Xuemin (Sherman) Shen

Abstract Web service is an emerging paradigm for distributing business applications from different platforms to a wide variety of clients. The critical factor in seamlessly accessing web services is to discover the appropriate service and the related service providers. Unfortunately, current web service technologies use centralized directory to keep the service index, which is not scalable and at the same time vulnerable to single point of failure. Peer to peer system is a popular decentralized architecture which can be used for key look up service with scalability and self organization. Thus there is an opportunity to intersect the P2P framework with web services to provide the scalable solution. In this chapter, we discuss the key methods to deploy web services using the peer-to-peer technology.

1 Introduction

Service oriented architecture (SOA) is a paradigm that creates a uniform interface for distributed systems to use applications from different platforms, independent of their implementation. SOA does not require an administrative supervisory to integrate application interfaces and provide services to the client of this applications. Due to this extra-ordinary advantage, this platform independent architecture has

Mohammad Towhidul Islam
Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada,
e-mail: mtislam@uwaterloo.ca

Mursalin Akon
Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada,
e-mail: mmakon@uwaterloo.ca

Xuemin (Sherman) Shen
Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada,
e-mail: xshen@bbcr.uwaterloo.ca

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_48, © Springer Science+Business Media, LLC 2010

drawn a vast attention in both the research community and the industry for next generation application development such as online scheduler, eHealthcare services.

Web services (WS), an important branch of SOA, becomes increasing popular for its easy deployment, fault-tolerance and unsupervised maintenance [6]. Web services are deployed with an implementation of SOAP/WSDL/UDDI on the top of the existing application framework. The request and the response of the services interact with SOAP (Simple Object Access Protocol) that carries XML (eXtensible Markup Language) data. The service provider describes their services using the WSDL (Web Service Description Language) which is composed using XML. UDDI (Universal Description, Discovery and Integration Protocol) provides directory service that contains the details about services and the service providers. Conventionally, a centralized server keeps the directory information about the available web services. A client is able to discover the required service from the directory. The solution follows the client/server architecture. All serving providers publish their services to a central authority and thus make the service available to all. However, this method suffers from the problem of single point failure. In addition, this system are not scalable. Because, there is a possibility that loads of clients are looking for services from a single server. This will lead to a serious performance bottleneck [11]. As a result, there may be an ample amount of delay in getting the service information while the service provider is sitting idle. Adding more servers to alleviate the problem is not an elegant solution. First, the solution is costly. Second, it requires a constant monitoring to compute the required number of servers. Therefore, it is necessary to find an alternate to provide fault tolerant, robust and scalable solution to deploy the web services.

Peer to Peer (P2P) computing brings a new era in distributed computing. P2P computing is largely used for sharing computer resources and services. Each functional unit in the system is termed as a peer, which acts as both a resource provider and a resource consumer. P2P network does not require any centralized directory or index to keep track of the available resources. In the unstructured P2P network, if a peer needs a specific resource, it will find the appropriate peer with the help of other peer. In the case of structured P2P network, decentralized index of the resource helps in this regard. A P2P network is a dynamic network because the network changes continuously as peers join and leave the network.

While comparing the P2P with web services, there is similarity in the way they function. In both systems, there are providers and the consumers of services. Publishing information of resources or services by providers and discovering required resource or services by consumers are common in both paradigms. However, they differ in the look up procedure. P2P depends on the decentralized discovery while the current web services standard stands on centralized look up service. Although there are some dissimilarity in service discovery, both of the architectures deal with loosely coupled systems. Therefore, the framework of P2P can be exploited to facilitate of web services. P2P solutions the web services provide the remedy of the single point failure and deal with scalability issue. In this chapter, we discuss the methodologies to utilize the P2P architecture in web services.

2 Web Services

There are four primary tasks in web services: publish, discovery, request and response. Publish is a process by which a service provider announces its service as well as the service associated interfaces. Generally, a service provider announces its service by entering service information into a specialized registry. The consumers of the services discover the services by various ways. Discovery is a process of finding an appropriate service that provides required functionality. Upon discovery, the consumer requests the functionality by providing required input. The service responds to the consumer with desired output. The aforementioned process reveals that a web service architecture contains three primary actors: requester, provider and a registry. In this chapter, our prime concern is the service registry which contains the listing and information of the services.

2.1 Architecture of the Web Services

Web services are formed by a layered architecture. The architecture has an excellent analogy with that's if the OSI model. Like the OSI model, each layer understands the corresponding layer of the participants. There are four fundamental elements or layer for the web services: XML (eXtensible Markup Language), SOAP (Simple Object Access Protocol), WSDL (Web Service Definition Language) and UDDI (Universal Description, Discovery and Integration).

The lower layer is the eXtensible Mark-up Language (XML). The XML layer works as information bearer. Through the XML, the actual services or the remote function would be called. XML is well known for its self describing tags. Unlike HTML, in XML, the creator of the file defines the tags according to the needs. Thus the XML with the power of the self describing tags is chosen for describing the web services. As a standard for data representation, XML is often used for data interchange on the Web. One of the challenges in data interchange is the integration of data from different parties. XML uses metadata such as DTD or XSD to describe the structure of data. This metadata make it easier to solve data integration problem in dissimilar platform. Thus XML is used as an excellent carrier for transferring different raw data for web services among participating parties.

Simple Object Access Protocol (SOAP) defines the standard of data exchange using XML document between a consumer and a service provider. The main goal of SOAP is to provide the communication protocol in decentralized, distributed environment of Internet where different operating systems and service applications coexist. A SOAP message contains three different parts. They are *envelope, header and body*. The entire SOAP message is encoded in the envelope. The envelope defines the encoding technique and namespaces. The header part is optional and contains additional information. The body contains exact data to be exchanged.

WSDL is a XML based language which describes web service along with the location of service providers and the service interfaces, i.e., the way of accessing

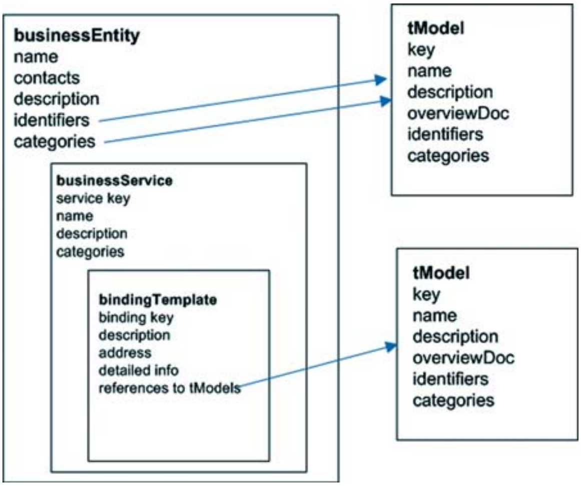


Fig. 1 Schematic structure of UDDI

associated functionalities. The *portType* in WSDL defines the operation or the function that can be invoked by the consumer whereas the *message* describes the parameters of the functions. The *types* define data types of the parameters in message. Thus, clearly WSDL separates the functionality of the operation of a service provider from its description. The separation of functionality from the description gives flexibility of web services just like high level language such as Java.

UDDI provides dynamic binding between a service and a client and retrieve the reference of the service by querying the registry. Discovery of a service and its description is a very important part of web services. UDDI provides listing of the available functions from different service providers as a directory. Once a service provider is ready to publish its service using SOAP and WSDL, the service provider lists the service using UDDI to be discovered by the clients. Generally, a server hosts the directory listing. However, efficient discovery of service from millions of services faces different problems such as scalability, robustness. Therefore, it is necessary to find alternatives for directory listing of web services, either using UDDI or not. In the following section, we will discuss the structure of the UDDI for service registry.

2.2 UDDI Structure

UDDI contains four different types of information about a service. They are business information, service information, binding information and technical information. Figure 1 shows a schematic diagram of UDDI. The *businessEntity* provides information about the organizations which publish the service. The *businessService* describes the classification of service offered by the businessEntity. A particular service may have different versions and implementations. The businessService accumulates all the versions using different entities for each of them. A businessEntity may contain several businessServices. The *bindingTemplate* represents the technical information for the services. This entry is of particular interest of the problem addressing here as this entry of the UDDI structure provides the address/location of a service. In addition, the bindingTemplate also discusses the parameters/inputs for the services and the types of each parameter. A businessService may contain several bindingTemplate elements. The *tModel* or the technical model contains the specification of the service. Unlike other UDDI components, the tModel is not organized in a hierarchy. Generally, tModel represents the WSDL service interface or the interaction model. If needed, the semantics of he service is also represented in tModel. The bindingTemplate often refers to an appropriate tModel to describe an offered service. The maximum benefit of this decoupling is achieved, when several business provides the same service, and hence several bindingTemplates refer to a single tModel.

3 Peer-to-Peer Computing

Peer-to-Peer (P2P) computing is a milestone in distributed computing. In P2P computing, every peer works both as a consumer and a provider of resource. This structure makes the contrast with the client-server computing where a single server provides resource or services to clients. P2P is a popular paradigm for exchanging data among interested peer in decentralized manner. Initially, P2P gained its popularity for file sharing application. Two such early systems are Napster [16] and Gnutella [22], both of which were constructed randomly and data can be located at any peer inside the system. This type architecture is known as *unstructured P2P systems*. It is inefficient to search in unstructured P2P system. *Structured P2P system* organizes the peers according to certain structure and also distributes data among the peers using some criteria. The popular criterion for peer organization and data placement is *Distributed Hash Table (DHT)* [21]. The structured P2P system provides more scalability and efficient searching of data over the unstructured P2P systems. On the other hand, unstructured P2P system is more robust and fault tolerant than the structured P2P system in the case of dynamic insertion and deletion of peers to and from the system.

4 Web Services and P2P

The Fig. 2 shows the structure of a centralized registry. Centralized UDDI service provides an easy way to discover services from a registry database. However in practice, most of the service providers set up their own private registry using available UDDI tools like Microsoft Active Directory, Novel eDirectory [8]. The trend of

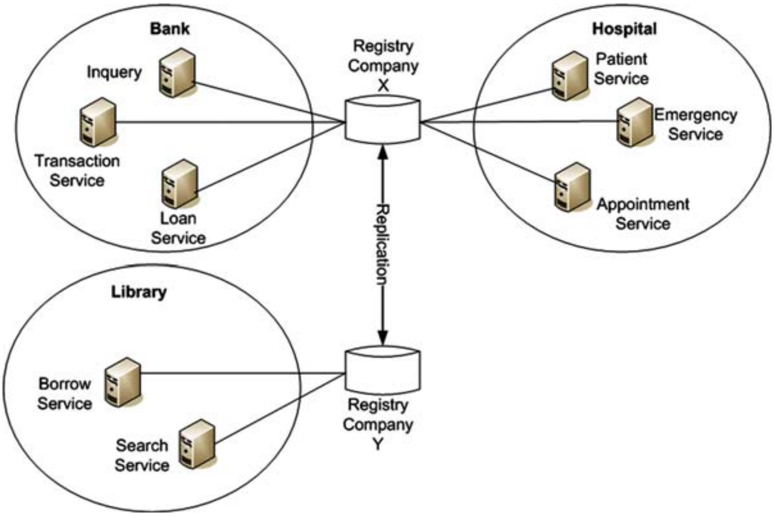


Fig. 2 Service registries using the central registries

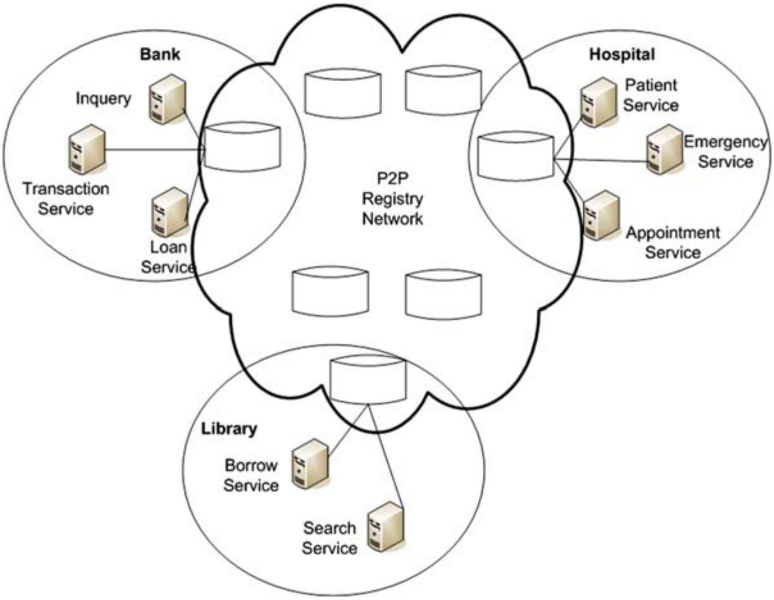


Fig. 3 Service registries using the P2P network

setting up the private registry makes it difficult to find the required service quickly and easily. Theoretically, it is possible to replicate private registries into a central server and provide a centralized access to all. However, in that case, there should be an agreement between the central registry authorities with all the private registry owners. In addition, it is impractical to set up a large central registry server that replicates all the private registries. To get rid of this problem, researchers were looking for the distributed solution of UDDI registry. Since P2P networks have been shown to be efficient decentralized solution for different problems, researchers have found it convenient to connect the private registries with peer-to-peer technology. Figure 3 shows the architecture of peer-to-peer network based web services. In this system, each organization shares the registry with other company through the P2P networks. There is no need for a separate organization to maintain the centralized registry and replicate the information. Early P2P systems require exact keyword matching for resource discovery. On the other hand, web service discovery needs support for semantic search along with keyword search. For example, a person is looking for a meal where the ingredients should not contain egg. Using the semantic technique he may easily search for such a service. Therefore, there are two different types of intersection of P2P and web services are available in the literature. The first type is P2P based web services which support keywords (including partial keyword) for service discovery. The second type is semantic web services based on P2P systems which allow semantics oriented service discovery. In rest of the chapter, we discuss both these architectures.

5 A Framework for Web Services Using P2P

Forster et al. formulate the web services using the concept of WS peers and user peers [7]. They replace the UDDI infrastructure with the two layers of P2P architecture. The first layer of the peers is termed as WS peer. The WS peer caters information about the web services. The second layer contains the user peers which are usually the consumer of the web services.

5.1 Registering a Peer

To join a P2P based web services, a peer has to get the initial information from other existing peer of the network for bootstrapping purpose. A WS peer is connected to other WS peers as a neighbor. A user peer may be connected to other user peers or WS peers. However, each user must have at least one WS Peer in its neighborhood list. This restriction assures that the query from user-peer-layer must reach to WS-peer-layer. When the peers exchange information, each peer keeps the time stamp of the relevant information. At the time of joining, each peer is given a set of addresses of user as well as WS peers. While active in the network, peers either query for the web services or forward queries from other peers. In case of a successful service discovery, a peer caches the address of the service along with the time stamp of caching. It also caches the other related information of the service for future reference. When the cache is full with accumulated information web services, a well known cache replacement policies [20] is used. In addition to reactive information collection by the user peers, the WS peers periodically broadcast the *key name* and the *description* of their services to the neighbors. The neighbors cache the information for the future use.

5.2 Publishing Information

Forster et al. [7] use the structure of the UDDI to publish the web services. A provider uses the *BusinessKey*, *ServiceKey* and *BindingKey* of UDDI to describe its web services. When the provider joins a P2P networks, it introduces files containing UDDI to the P2P network. The service information should be unique to identify a specific service of a service provider from a pool of services. It is easy to implement this uniqueness with a central instance of service. However, in a decentralized environment such as P2P networks, the implementation of uniqueness for a specific service may not be straight-forward. One of the common solutions for ensuring the uniqueness is to use a hash function on the IP address of the provider. As it is described before in Section 2.2, the *tModel* of UDDI is used to describe a service. UDDI specifies that similar services should use the same tModel to provide a common interface. The unique *tModelKey* can also be generated by applying a hash-function on the description of tModel. The uniqueness of tModelKey can be achieved in the following way. When a service provider plans to cater a service, it queries for the same service in the P2P networks. If the provider finds the same service in the P2P networks, the service provider uses the existing tModel. The *bindingTemplate* of the service provider has the reference to the remote file of tModel. If there is no response for the query (i.e., the service does not exist in the P2P networks), it creates a tModel and introduce in the P2P network in the same way as described earlier. This method of publishing information in P2P network does not require substantial changes to the UDDI standard except the creation of unique identifier. Therefore, the service providers do not require any additional effort to maintain complex registry and publish their services to the P2P network with ease.

5.3 Finding the Service

In the aforesaid model for web services, a peer queries for services. When a query reaches a WS peer, the peer searches the local files for the suitable service. If it finds the service, it will respond back to the sender. Otherwise, it looks into its cache for remote references. If remote reference is found, a response is sent back to the requester who in turn contacts with the original source of the service. If the requester fails to find the original source, it notifies the responder WS that the information is stale. The responder removes service information from the local cache. Caching of

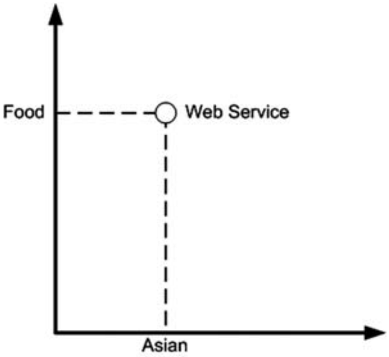


Fig. 4 A two dimensional keyword space. The data element `Web Service` is described with `Food:Asian`

information makes the system to be able to apply complex query and thus make the system efficient.

6 Keyword Based Service Discovery

Schmidt et al. present a P2P based dynamic, flexible and decentralized architecture for web services [15]. The architecture includes a keyword based query engine is for service discovery. The architecture uses Space Filling Curve [12] to map services to peers. It preserves locality in index structure for efficiency. Keywords describing the web services use index for locating the web services in different peers in P2P system with DHT approach. The key innovations of their method are that their system support complex queries with partial keywords. The query cost is a bounded function which is fairly small for any typical P2P network. The following three subsections describe the architecture presented by Schmidt et al.

6.1 System Architecture

The main components of the architecture are the locality preserving mapping of data elements. Figure 4 shows an example where data element is a point that represents *web service*. The keywords *Food* and *Asian* are the coordinates. The representation of keywords in coordinates in known as *index space*. Each data element is associated with a sequence of keywords. Two data elements are termed as local if they either share common keywords or their keywords are lexicographically close. Consider two services having the keywords (University, Waterloo) and (University, Manchester). Then these data elements can be termed as local to each other.

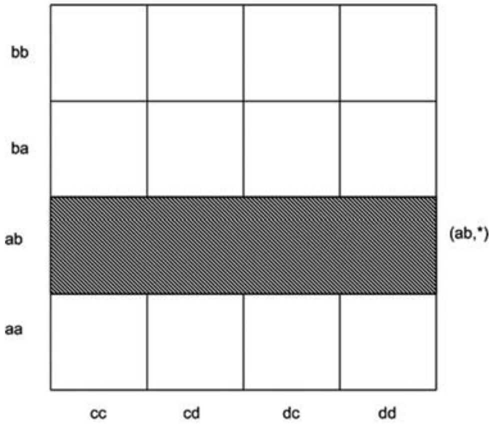


Fig. 5 A two dimensional space defined the query `(ab,*)`

The construction of the index space is crucial because it should support queries for partial keyword and the wild cards. Therefore, Hillbert Space Filling Curve (SFC) [12] is used. SFC transfers a multidimensional space of keywords to 1-dimensional index space while preserving locality and keeping digital causality [12]. Clusters are formed by similar documents sharing closely related keywords in SFC.

The one-dimensional index space generated by SFC is mapped to physical or logical peers of P2P networks. The mapping method is similar to the Chord [17]. Each peer gets an identifier randomly. Each data element gets an identifier from the one-dimensional index space. A data element is mapped to the peers whose identifier is equal or greater than to the identifier of the data element.

In this architecture, a peer joining requires $O((log_2)^2 N)$ massages where N is the number of peers in the system. The cost of peer departure is similar to the cost of peer join. Since this architecture is similar to Chord in terms of peer organization, finding a service requires on average $O(log_2 N)$ hop traversal.

6.2 *Query Engine*

The query engine is responsible to answer queries which may consist of combination of keywords, partial keywords, or wildcards. The query engine maps a complex query onto relevant clusters and then finds out the appropriate peer responsible for the query. The process can be explained with an example. The shaded portion of Fig. 5 represents a 2-dimensional space defined by the query (ab,*). The query (ab,*) refers four data elements which may be contained in a single or multiple clusters as defined by the SFC curves [12]. These clusters are mapped to one or more adjacent peers according to the size of the clusters and the data points. The query for the keyword(s) is directed to the matching clusters. The peers employ a service look up protocol which finds the appropriate service for the query and return the result to the requesting peer.

6.3 Load Balancing

Schmidt et al. observes that the keyword space is typically sparsely populated with data elements. It may happen that some of the peers are overloaded with the data elements whereas other peers are under loaded. This scenario creates a potential problem in load balancing. To avoid this problem, a peer can get several random identifiers at the bootstrapping process. It is already stated that this architecture uses Chord for peer organization. Chord organizes peers in a ring. A joining peer has to find its position inside the ring. A peer sends joining request to all its potential successors in the ring for each generated identifier. Each of the successor peers sends back the response with the amount of load in that peer. The new peer will join the region which is heavily loaded. The cost of finding the best identifier is $O(n \cdot log_2 N)$. However, load balancing at the bootstrapping may not be sufficient. Therefore, the following run time load balancing techniques are also used.

There are two common runtime load balancing approaches. The first load balancing algorithm is simple. This technique periodically checks the load for each peer and transfers the data elements of the loaded peer to the neighboring less loaded peer(s). The cost of load balancing using this algorithm is $O((log_2)^2 N)$. Since this algorithm is expensive, it is run rarely in the system. The second algorithm uses multiple virtual peers inside a physical peer. When a virtual peer is overloaded, it is split into two or more virtual peers. Similarly, load is also split and distributed among the split peers. If a physical peer is overloaded, its virtual peers are transferred to less loaded peers. Thus the load is balanced in the entire system.

7 Semantic Web Service

For service discovery, the current standard of the web services support exact matching of the service name. In another way, the publisher of the web service defines the interface of the function or operation and how to invoke them using WSDL. The consumer of the service finds their required service based on the information on the UDDI registry. This process will bring emerging success if all interested parties agree upon certain standard for naming and describing their service. Unfortunately, there is no such standard available. The lack of the standardization generates the problem of searching the required service efficiently. There could be a superficial difference between the service provider and the service requester in interpreting service functionality. There has to be an unambiguous and machine-interpretable form of representation of the properties, capabilities and the interfaces of the service to get rid of the aforementioned problem. The service providers produce the semantic model with the intended meaning and use of terms. The service providers also provide the formal and informal definition of the entities. The semantic of the services will be published in the UDDI registry. The service requester uses the terms from the semantic model for service discovery. If there is a direct match of service description between the service request and the service provider, only then the result is returned. Otherwise, a composite query is formed to find the desired service.

7.1 Domain Ontology

The description of the semantic web services is usually published using the *Service Annotation Ontologies* and the *Domain Ontologies* [3]. The service annotation ontologies describe the functionality, execution flow, invocation details, etc. of the web services using the terms that is defined in the respective domain ontologies. The service annotation ontologies define the set of attributes for *Input, Output, Precondition and Effects (IOPE)*. The values of the IOPE usually come from the domain ontologies. According to Web Ontology Language (OWL) [2], the service annotation ontologies are divided in three logical parts. They are *Service Profile*, *Service Model* and *Service Grounding*. *Service Profile* keeps information about the service description with the providers contact information, a functional description of the service and a list of additional parameter of the service. *Service Model* specifies the service operation in detail. *Service Grounding* gives details of how to access the service.

An unstructured text cannot fully represent the service annotation ontologies. For example, the term 'window' refers completely two different components when it is used in home and operating system context. Therefore, there is a need for domain ontologies which describe the service from the domain point of view. The domain ontologies describe the terminology and term relationship related to a specific domain. For example, a car rental service provider describes his services using the car domains. The car domain ontologies may contain the terminologies and the relationship of these terms related to car. If everybody follows the same domain ontologies for the car, it is easy to communicate between the service provider and service consumer.

7.2 Semantic Web Service Discovery

Thaden et al. propose an architecture for web service registry based on DAML-S [1]. DAML-S is used as a service description language [18]. A DAML-S description contains four parts. They are *resource, service profile, service model, service grounding* The resource offers the functionality of service. This service is described in three parts. A service profile specifies the required services and provides an external view of the required input. A service model provides the steps to complete the service. Service grounding provides the implementation details. It is quite possible to map a WSDL information into DAML-S format [1]. Thaden et al. describe their services in DAML-S. They define their own ontology to describe different types of services. The ontology generates a hierarchy to define the service relationship. For example, assume the top of the hierarchy contains the *image* category under which there are two categories known as *rasterImage* and *vectorImage*. Now if someone makes a query for the *vector image* it will gives all the services related to the *vectorImage*. To search the service from DAML-S prototype, they develop a tool called *Edutella* query exchange language (QEL) [9]. QEL is able to generate complex queries for searching a service from the service descriptions. A peer simply sends the required QEL query in the P2P network. The P2P network distributes a query to all peers hosting registry information. Upon successful discovery of a service, the system accumulates the search result and sends it back to the requester.

7.3 METEOR-S WSDI: A Semantic Web Services Tool

Verma et al. propose Web Service Discovery Infrastructure (WSDI) based on METEOR-S project [19] for scalable semantic web services publish and discovery. In this project they use a special ontology called *Registry Ontology* which maintains relationship among different ontologies. A particular registry maintains ontology for its domain only and all queries related to this domain is directed to this registry. In their methodology, semantic information about the services is attached in the service description. Since, each of the domain ontology is connected with each other; they can use the shared vocabularies to publish the information. Thus a service description is made explicit and eliminates confusion about functionality. Each service provider can also provide their own version of semantic matching algorithm. To deal with the scalability issue and to handle increasing number of registries, P2P network is chosen for the deployment of WSDI. In the next section we discuss the architecture of the METERO-S WSDI (MWSDI).

7.3.1 Architecture of MWSDI

The MWSDI is divided into four layers. They are *Data Layer*, *Communication Layer*, *Operator Services Layer* and *Semantic Specification Layer*. The data layer consists of web service registries. The registries are kept in such a way that they can be accessed through stand-alone fashion as a result of consistent to UDDI standard. The semantic is published and discovered through the operator service layer.

The semantic specification layer is orthogonal to other layers. This layer creates accessibility of the semantic metadata. The semantic is used in two levels: at the level of registries and the level of individual web services. The Registry Ontology manages the specific domain and maintains relationship among the registries. It also keeps the properties of the registries which include the registry specification name,

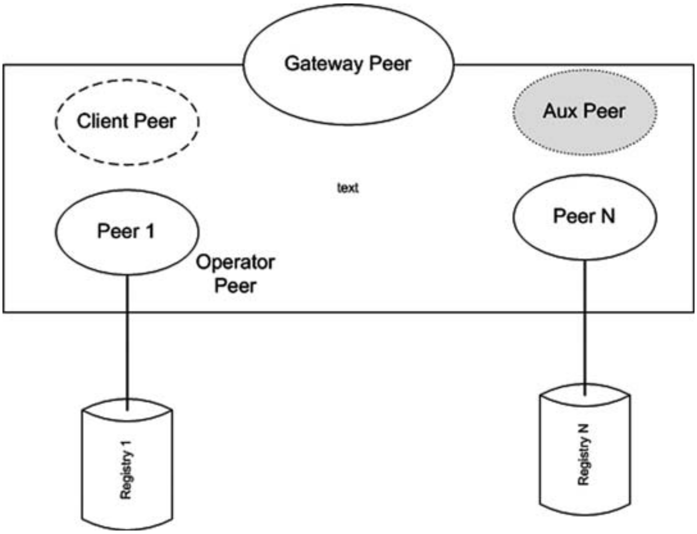


Fig. 6 Components of the communication layer of the METEOR-S WSDI

registry specification version, registry operator details, and quality of the service. A domain specific ontologies supports private semantic publication and discovery of web services. Therefore, individual registry operators create their ontologies from particular domain concepts and terminologies. The MWSDI just maps the input and output of domain ontologies and stores in UDDI data structure [10].

The communication layer establishes communication among different distributed components of MWSDI. This layer utilizes a P2P network to maintain communication. The communication layer consists of four different types of peers: *operator peer*, *gateways peer*, *auxiliary peer*, and *client peer*. Figure 6 shows the different types of peers used in the communication layer. Each operator peer maintains a registry and provides the registry ontology to all other peers. The gateway peer works as the entry point for others who want to join the MWSDI network. It updates the registry ontology for a new peer and delivers this update to all other peers. The auxiliary peer handles the registry ontology and performs a crucial role in the MWSDI architecture. The auxiliary peer is responsible to maintain high availability of the registry ontology because it is critical for performance of the MWSDI. The client peer is used only to utilize the capabilities of the MWSDI.

The operator service layer plays the key role for semantic publishing and discovery of services. This layer acts as n interface between the registry and the users. This layer provides value added services to the user. A user can download the domain specific ontologies using this service. The user chooses the relevant registry using the client peer and discovers registry using templates. Different registry operators provide different algorithms for service publishing and discovery. However, the user of a service does not need to aware of the differences. The templates created in earlier steps communicate the operator service layer. This operator service layer

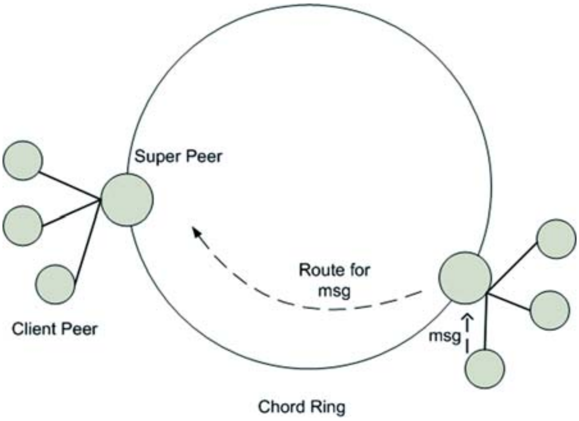


Fig. 7 General architecture of the spider system

translates the template into the desired format for getting the relevant services. Thus the entire complexity of semantic based web service discovery and publication is kept hidden from the users. The non-UDDI registry provider uses the facility of the operator service layer to publish their services. Keyword based UDDI query is also supported from the operator service layer.

8 Spider: A Unified Web Service Discovery Tool

In [13], Sahin et al. propose a system known as SPIDER for web services publishing and discovery. In their system, they identify three different ways of discovering services from different service providers. They are *keyword based, ontology based*, and *behavior based* search. A variation of keyword based search is discussed in Section 6. The spider architecture is built on the top of Chord [17]. It uses *super peer* based overlay on the top of the Chord network. Figure 7 shows the architecture of the spider system. A super peer is responsible for a particular service category and is chosen from a group of peers who provides similar services. The peer with high availability, large computing capacity and ample amount of other resources is selected as the super peer. Other peer on the same service category is termed as *client peers*. Each client peer is connected to the system through a super peer. The super peer processes the requests on behalf of the client peers.

8.1 Keyword Based Search

In keyword based search, a service is presented in WSDL and the keyword is extracted from the WSDL. The SMART [14] tool can be used for this keyword extraction. Interested consumers of a service then locate the service provider by searching list of services using keyword. Once the keywords for the service are defined, the service offering peer sends a message to the system. The message contains the keywords and necessary information to locate the service. The super peer takes the keywords and uses a hash function to create map from keyword to service location. When the system wants to locate a service for a given keyword, it communicates with the related super peer. The super peer in turn uses the same hashing function to locate the corresponding service peer.

8.2 Ontology Based Search

Domain ontology based search is similar to category based searching as described in Section 7.1. The system defines a list of category of services at the beginning based on a standard such as NAICS [4]. Each peer in a network keeps a copy of the list. A joining peer contacts with a nearby peer for the list. A service provider publishes a service according to the related domain ontology. Like the keyword based search, a super peer hashes the address of the service provider along with service category. When querying for a particular service, the corresponding super peer consults its ontology listing and returns all the matching peers providing the certain service.

8.3 Behavior Based Search

User can invoke a service by defining the behavior of a service. This creates a new era in the web services discovery. The service behavior can be described as the process flow language like BPEL [5]. Authors argue that if the behavior of service is known, this information can be used to increase the accuracy of discovery process. Consider an example, where a health care provider describes required interaction between services and a user. One of its services periodically checks the pulse rate of the user and keeps the data for future reference. Someone may want such specific behavior from a service. A service provider peer uses a finite state automation [1] to describe the service. Finite state automation defines the execution path of the service. Super peer hashes extracted information from the finite state automation to generate the key. The super peer stores the behavior list along with necessary information to facilitate service discovery. A user describes the behavior of a desired service using BPEL. This behavior is transferred to an acceptable path by the system. According to the path, the super peer returns the service information.

[1] Finite state automation information can be extracted from the BPEL document of the service

9 Discussion on Open Issues

There are numerous approaches to deploy of web services using peer-to-peer networking. Most of the approaches in the literature focus on the method of discovery of service using the P2P framework. However, quantitative analysis and comparison are still missing. In case of semantic web service publishing and discovery, most of the authors make the assumption that both service provider and the service requester use the same domain ontologies and vocabularies. However, in distributed platform like real life P2P, it is very difficult to exercise this. The diversity of service advertisement and service request should be eliminated to ensure efficient match making algorithms.

There is stability issue related to the P2P web services. Since P2P is vulnerable to frequent join and leave of peers, the network infrastructure is not stable and changes often. Most of the design for P2P web services ignores this fundamental issue. Therefore, a fair comparison between web services in P2P and client server architecture environment is yet to be sought.

10 Conclusion

In this chapter, we presented a comprehensive study on the web services using P2P networks. The web services architecture is similar to a P2P framework in many aspects. However, they have different philosophy to maintain service directory. Maintaining the service directory in a server oriented approach holds back scalable deployment of web services. P2P provides an efficient decentralized solution towards service indexing without modifying the framework of the web services. Deploying web services using P2P provides attractive alternatives to the current day's technology of web services and service oriented architecture. Researchers are actively looking into web services in mobile P2P networks. It is expected this will create a window of opportunity for applications in extra-ordinary but simple P2P networks.

References

1. Ankolekar, A., Burstein, M.H., Hobbs, J.R., Lassila, O., Martin, D.L., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T.R., Sycara, K.P., Zeng, H.: Daml-s: Semantic markup for web services. In: SWWS, pp. 411–430 (2001)
2. Antoniou, G., Antoniou, G., Harmelen, F.V., Harmelen, F.V.: Web ontology language: Owl. In: Handbook on Ontologies in Information Systems, pp. 67–92. Springer-Verlag (2003)
3. Antoniou, G., van Harmelen, F.: A Semantic Web Primer. The MIT Press, Cambridge, Massachusetts (2004)
4. Canada, S.: North american industry classification system (naics) 2007 - canada (2007). URL http://www.statcan.ca/english/Subjects/Standard/naics/2007/naics07-menu.htm

5. Curbera, F., Goland, Y., Klein, J., Leymann, F., Roller, D., Thatte, S., Weerawarana, S.: Business process execution language for web services version 1.0. (2008). URL http://dev2dev.bea.com/techtrack/BPEL4WS.jsp
6. Doulkeridis, C., Zafeiris, V., Nrvg, K., Vazirgiannis, M., Giakoumakis, E.A.: Context-based caching and routing for p2p web service discovery. Distributed and Parallel Databases **21**(1), 59–84 (2007)
7. Forster, F., de Meer, H.: Discovery of web services with a p2p network. In: Discovery of Web Services with a P2P Network. Springer, Krakow, Poland (2004)
8. Moore, C.: Novell rolls out uddi server. http://www.infoworld.com/
9. Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmér, M., Risch, T.: EDUTELLA: a P2P networking infrastructure based on RDF. In: Proceedings of the 11th international conference on World Wide Web, pp. 604–615. ACM New York, NY, USA (2002)
10. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Importing the semantic web in uddi. In: In Proceedings of E-Services and the Semantic Web Workshop, pp. 225–236 (2002)
11. Papazoglou, M.P., Krämer, B.J., Yang, J.: Leveraging web-services and peer-to-peer networks. In: In Proc. of the 15th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2003, pp. 485–501 (2003)
12. Sagan, H.: Space-filling Curves. Springer-Verlag (1994)
13. Sahin, O.D., Gerede, C.E., Agrawal, D., Abbadi, A.E., Ibarra, O.H., Su, J.: Spider: P2p-based web service discovery. In: Proceedings of the Third International Conference Service-Oriented Computing (ICSOC 2005), pp. 157–169. Springer (2005)
14. Salton, G.: The SMART Retrieval System – Experiments in Automatic Document Processing. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1971)
15. Schmidt, C., Parashar, M.: A peer-to-peer approach to web service discovery. World Wide Web **7**(2), 211–229 (2004)
16. Stern, R.H.: Napster: A walking copyright infringement? IEEE Micro **20**(6), 4–5, 95 (2000)
17. Stoica, I., Morris, R., Karger, D., Kaashoek, F.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. SIGCOMM Computer Communication Review **31**(4), 149–160 (2001)
18. Thaden, U., Siberski, W., Nejdl, W.: A Semantic Web based Peer-to-Peer Service Registry Network. Tech. rep. (2003)
19. Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S., Miller, J.: Meteor-s wsdi: A scalable p2p infrastructure of registries for semantic publication and discovery of web services. Information Technology and Management **6**(1), 17–39 (2005)
20. Wierzbicki, A., Leibowitz, N., Ripeanu, M., Wozniak, R.: Cache replacement policies revisited: the case of P2P traffic. In: CCGRID, pp. 182–189. IEEE Computer Society (2004)
21. Zhu, Hu: Efficient, proximity-aware load balancing for DHT-based P2P systems. IEEETPDS: IEEE Transactions on Parallel and Distributed Systems **16** (2005)
22. Zhu, Y.: Making search efficient on gnutella-like P2P systems. In: 19th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2005). Denver, Colorado, USA (2005)

Content-Based Publish/Subscribe Systems

Haiying Shen

Abstract The application and deployment of publish/subscribe systems have developed significantly over the past years. A publish/subscribe system is a powerful paradigm for information dissemination from publishers (data/event producers) to subscribers (data/event consumers) in large-scale distributed networks. Publish/subscribe systems have been used in a variety of applications ranging from personalized information dissemination to large-scale and critical monitoring. This chapter provides a survey on current content-based publish/subscribe systems. It first introduces the publish/subscribe systems, then presents a survey of current systems based on three classification criteria: subscription model, routing and topology. It details different publish/subscribe systems in the centralized category and distributed category including multicast-based systems and Distributed hash table (DHT)-based systems. Finally, it concludes the chapter with concluding remarks and open issues.

1 Introduction

In the past few years, with the tremendous development of Internet and rapid growth of information, more and more Internet applications require information dissemination among a large number of widely scattered entities. In this environment, thousands or even millions entities are distributed globally and their locations and behaviors may vary. The large-scale, dynamic and geographically spread features of the environment requires a scalable, efficient and reliable technique for information dissemination. The rigid and static individual point-to-point and synchronous communications are not able to meet the requirements. Publish/subscribe (pub/sub) systems has been receiving increasing attention for the loosely coupled form of interaction it provides in large scale settings [65]. A pub/sub system [47] enables information

Haiying Shen
University of Arkansas, Fayetteville, AR, USA, e-mail: hshen@uark.edu

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_49, © Springer Science+Business Media, LLC 2010

dissemination from *publishers* (data/event producers) to *subscribers* (data/event consumers) in large-scale distributed networks.

The first pub/sub system was the "news" subsystem in the Isis Toolkit and was described in [19]. This pub/sub technology was invented by Frank Schmuck, who probably should get the credit as the first person to ever invent a fully functional pub-/sub solution [2]. Since then, significant research work has been devoted to developing efficient and scalable pub/sub systems. Pub/sub systems have been applied to a wide range of group communication applications including software distribution, Internet TV, audio or video-conferencing, virtual classroom, multi-party network games, distributed cache update, distributed simulation and shared white-boards. It can also be used in even larger size group communication applications, such as broadcasting and content distribution. Such applications in our daily lives include news and sports ticker services, real-time stock quotes and updates, market tracker, and popular Internet radio sites [16].

A pub/sub system is composed of many nodes distributed over a communication network. In such a system, clients are autonomous entities that exchange information by publishing events and by subscribing to the classes of events they are interested in. Clients are not required to communicate directly among themselves but are rather decoupled: the interaction occurs through the nodes of the pub/sub system that coordinate themselves in order to route information from publishers to subscribers [6]. Figure 1 shows a high-level view of a pub/sub system. In the system, publishers produce information and subscribers consume information. Specifically, publishers publish information in the form of events and subscribers express their interests in an event or a pattern of events in the form of subscription filters. A data event specifies values of a set of attributes associated with the event. The subscriptions can be very expressive and specify complex filtering criteria by using a set of predicates over event attributes. When a pub/sub system receives an event published by a publisher, it matches the event to the subscriptions and delivers the event to the matched subscribers. A subscriber installs and removes a subscription from the pub/sub system by executing the subscribing and unsubscribing operations respectively.

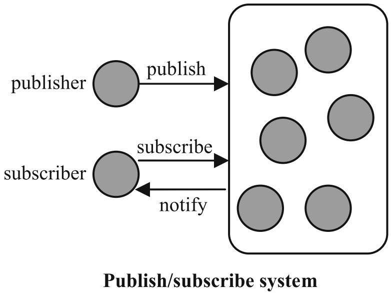


Fig. 1 A high-level view of a pub/sub system

Processes in pub/sub systems are clients of an underlying *notification service* and can act both as producers and consumers of messages, called event notifications or notifications for short. A notification is a message that describes an event. Notifications are injected into the event system via a `publish()` call rather than being published towards a specific receiver. They are conveyed by the underlying notification service to those consumers which have registered a matching subscription with `subscribe()`. Subscriptions describe the kind of notifications consumers are interested in.

A variety of content-based pub/sub systems have been proposed. The pub/sub systems can be classified into groups according to three criteria: subscription model, routing and topology. Based on the expressive power of subscription models, pub-/sub systems can be classified into three categories: topic-based, content-based and type-based. According to routing solutions for the notification service, the pub/sub systems can be categorized into filter-based approaches [8, 26, 27, 36, 73, 90] and the multicast-based approaches [8, 73, 83, 104]. Based on the system topology, current pub/sub systems can be classified into centralized [88] and distributed [25, 28, 29, 102]. The distributed systems can be further classified into broker-based and Distributed Hash Table (DHT)-based systems. DHT systems are also called structured peer-to-peer (P2P) systems, which is one type of P2P systems.

Table 1 Classification of pub/sub systems

Classification criteria	**Categories**	
Subscription model	Topic-based	
	Content-based	
	Type-based	
Routing	Filter-based	
	Multicast-based	
Topology	Centralized	
	Decentralized	
	Broker-based	DHT-based

This chapter is dedicated to providing the reader with a complete understanding of content-based pub/sub systems. Table 1 shows the classification of pub/sub systems based on the three different classification criteria. We will introduce the pub/sub systems based on the three classification methods.

The rest of this chapter is organized as follows. In Section 2, we present the pub-/sub systems based on subscription models. In Section 3, we present the content-based pub/sub systems based on routing models, and introduce multicast techniques. Section 4 details different pub/sub systems according to system topology, and discusses various challenges in modelling the systems. Meanwhile, we present a number of examples for the content-based pub/sub systems discussing their goals, properties, strategies and classification. Finally in Section 5, we conclude this chapter with discussion about a number of open issues for building pub/sub systems.

2 Subscription Models

Different ways for specifying the subscribers' interest result in distinct variants of the pub/sub systems. The subscription models that appeared in the literature are characterized by their expressive power: highly expressive models offer subscribers the possibility to precisely match their interest, i.e., to receive only the events they are interested in [6]. In this section we briefly review the most popular pub/sub subscription models: topic-based model, content-based model and type-based model.

2.1 Topic-Based Systems

In the topic-based pub/sub systems, each event belongs to a certain topic (also referred to as group, channel or subject). Subscribers express their interest in a particular subject and they receive all the events published within that particular subject. Each topic corresponds to a logical channel ideally connecting each possible publisher to all interested subscribers. Network multicasting and diffusion trees can be used to disseminate events to interested subscribers. The topic-based model has been the solution adopted in all early pub/sub systems. Examples of systems that fall under this category are TIB/RV [72], SCRIBE [39], Bayeux [116], CORBA Notification Service [5], ISIS [18] and iBus [72] as well as commercial products Tibco [98] and Vitria [4].

Topic-based pub/sub systems take only coarse-grained subscriptions. The main drawback of the topic-based model is the very limited expressiveness it offers to subscribers. Consequently, a subscriber has to receive all events pertinent to a subject though the subscriber might be interested in only a subset of the events. In addition, topic-based systems provide limited choices of subscriptions. To address problems related to low expressiveness of topics, as indicated in [6], a number of solutions are exploited in pub/sub implementations. For example, the topic-based model is often extended to provide hierarchical organization of the topic space, instead of a simple flat structure such as in [13, 72]. A topic can be then defined as a sub-topic of an existing topic. Events matching the sub-topic will be delivered to all users subscribed to both the topic and sub-topic. Implementations also often include convenience operators, such as wildcard characters, for subscribing to more than one topic with a single subscription. Another method for enhancing expressiveness of the topic-based model is the filtered-topic variant [3, 5], where a further filtering phase is performed once the message is received based on the content of the message. Messages that do not satisfy the filter are not delivered to the application.

2.2 Content-Based Systems

In contrast to topic-based systems, content-based systems allow fine-grained subscriptions by enabling restrictions on the event content. In the content-base pub-/sub systems, notifications typically consist of a number of attribute/value pairs. A subscription may include an arbitrary number of attribute names and filtering criteria on their values. Only those events satisfying all the predicates are delivered to the subscriber. Hence, content-based systems increase subscription selectivity by allowing subscriptions have multiple dimensions [40]. Examples of content-based systems include Gryphon [1, 8, 73, 95], SIENA [26, 27], JEDI [36], LeSubscribe [80], Hermes [77, 78], Elvin [88], Rebeca [48, 49, 70], and CPAS [9]. In content-based pub/sub systems, the matching of subscriptions and publications is based on content and no prior knowledge is needed. Subscriptions in content-based pub/sub systems are more expressive. Subscribers express their interest by specifying conditions over the content of events they are interested in. In other words, a subscription is a request formed by a set of constraints composed through disjunction or conjunction operators. Possible constraints depend on the attribute type and the subscription language. Most subscription languages comprise equality and comparison operators as well as regular expressions. Therefore, these systems are more flexible and useful since subscribers can specify their interests more accurately using a set of predicates. The subscriber need not have to learn a set of topic names and their content before subscribing. The main challenge in building such systems is to develop an efficient matching algorithm that scales to millions of publications and subscriptions.

The complexity of the subscription language affects the complexity of matching operation. Therefore, it is not common to design subscription languages making requests more complex than those in conjunctive form such as those in [20, 23]. The work in [70] presents a complete specification of content-based subscription models. In content-based pub/sub systems, events are distinguished by the properties of the events instead of predefined criterion (i.e., topic name). Thus, the correspondence between publishers and subscribers is on a per-event basis. The difference with a filtered-topic model is that events that do not match a subscriber can be filtered out in any point in the system rather than on the receiver. For these reasons, the higher expressive power of content-based pub/sub comes at the cost of a higher overhead for calculating the set of interested subscribers for each event [26, 45].

2.3 Type-Based Systems

In type-based systems such as Echo [41], XMessage [92] and the work in [43, 44, 46], pub/sub variant events are objects belonging to a specific type, which can encapsulate attributes as well as methods. In a type-based subscription, the declaration of a desired type is the main discriminating attribute. That is, type-based pub/sub systems occupy the middle-ground between coarse-grained topic-based systems and fine-grained content-based systems. In terms of aforementioned models, a type-based pub/sub system is in the middle, by giving a coarse-grained structure on events (like in topic-based) on which fine-grained constraints can be expressed over attributes (like in content-based). For example, in XMessages [92], a publisher and a subscriber can either interact directly with each other, exchanging events or use an XMessage channel that allows multiple publishers and listeners to communicate asynchronously. Publishers and subscribers initially use lookup table to get the reference of XMessage channel for building the connection. Then, they use SQL-like query to filter messages from this channel based on the content of messages. Thus, XMessage is type-based. Messages in XMessage may be defined as any XML content that needs to be transmitted between source and sink while events are also XML strings but have the typed fields.

3 Filter-Based and Multicast-Based Pub/Sub Systems

A main aspect in a pub/sub system is event dispatching in which matched events are routed to subscribers. According to routing solutions, the pub/sub systems can be largely categorized into two classes [24]: the filter-based approaches [8, 26, 27, 36, 73, 90] and the multicast-based approaches [8, 73, 83, 104]. In the filter-based approaches, routing decisions are made through successive content-based filtering at all nodes along the path from source to destination. Every pub/sub server in the path matches the event with remote subscriptions from other servers, and then forwards it towards directions that lead to matching subscriptions. This approach can achieve high efficiency, but at the cost of expensive subscription information management and high processing load at pub/sub servers.

In the multicast-based approach, certain multicast groups are determined before event transmission. For each event, one group is determined at the publisher, and the event is then multicasted to that group. In this method, some nodes in the routing path receive the events they are not interested in. The network efficiency of this approach is often highly sensitive to the data types and the distributions of events and subscriptions in the application.

Recently, much research effort has been devoted to the distributed pub/sub systems. The architecture designs include SIENA [26, 27], Gryphon [8, 73, 95], JEDI [36], Rebeca [48, 49, 70], Elvin [89], Ready [55], and Herald [22]. Most of these systems adopt the filter-based routing approach. For example, in JEDI, a hierarchical interconnection topology is proposed in which a server is only informed of subscriptions from servers in its sub-tree. Events are always forwarded up the hierarchy regardless of the interests in other parts of the network.

Pub/sub systems relying on multicast for event dispatching need content-based matching to discover the events and subscriptions. Event dispatching in a pub/sub system is similar to the traditional multicasting. The only difference is that the addresses of the message receivers are known in multicasting, while in pub/sub systems the receivers need to be determined by content-based matching. The matching problem has been studied for various data types and event schemes [8, 10, 47, 93].

Many pub/sub systems rely on multicast for notification service. That is, a publisher forwards events to many subscribers who subscribe to the publisher.

Many overlay-based multicast systems are proposed in the recent years, such as Narada [87], Bayeux [116], NICE [16] and Scribe [39]. Multicast protocols can be classified into centralized-based and distributed-based. Examples of centralized methods include HBM [84] and ALMI [74]. The distributed multicast implementations can be classified according to a number of criteria. We list the criteria and classifications in the following.

Collaboration techniques. There are two multicast architectures: P2P architectures and proxy (i.e., broker)-based architectures [35]. A P2P architecture pushes the functionality to nodes participating in the multicast group so that each node maintains the state of those groups that it is participating, while a proxy-based architecture lets an organization that provides value and services deploy proxies at strategic locations on the Internet. End node attaches itself to proxies near them, and receive data using plain unicast, or any available multicast media.

Distribution. Two types of multicast according to information distribution are tree-based and flooding [31] (including enhancement of flooding method such as gossip and random walking). The flooding approach such as CAN-based multicast [82] creates a separate overlay network per multicast group and leverages the routing information already maintained by a group's overlay to broadcast messages within the overlay. The tree approach, such as Scribe [39] and Bayeux [116], uses a single overlay and builds a spanning tree for each group, on which the multicast messages for the group are propagated.

Overlay network construction. Currently proposed multicast protocols are either built from scratch or based on an overlay network substrate such as Pastry [85], CAN [81] or Tapestry [113]. Examples of the former category include Narada [35] and NICE [16] and the latter include Scribe [39] based on Pastry, Bayuex [116] based on Tapestry and CAN-based multicast [82] based on CAN. According to the taxonomy of overlay multicast provided in [42], the former category can be further classified into two classes. (1) *Direct tree construction.* Members choose their parents from the members that they know. Protocols such as Yoid [51], BTP [59], Overcast [61] TBCP [68], HMTP [109], NICE [16] and ZIGZAG [99] use this way to construct trees. (2) *First mesh construction, second tree construction.* That is, first efficient meshes are constructed, then trees are constructed out of the meshes by certain routing algorithms. Such examples include Narada [35], Gossamer [32] and Delaunay triangulation [63]. The overlay network substrate category can be further classified into generalized hypercube such as Scribe [39] and Bayuex [116], and Cartesian Hyperspace such as CAN-based multicast [82] according to overlay network construction.

These proposals use two different techniques to design self-organizing multicast in order to improve the scalability of multicast. (1) *Neighbor mapping based on members' assigned addresses.* For example, CAN-based multicast [82] assigns logical addresses from cartesian coordinates on an n-dimensional torus. Delaunary Triangulations [63] assigns points to a plane and determines neighbor mappings corresponding to the Delaunay triangulation of the set of points. (2) *Organizing* *members into hierarchies of clusters.* Nice [16] and Kudos [60] are such instances. Kudos constructs a two level hierarchy with a Narada like protocol at each level of the hierarchy. Banerjee et al. [16] constructs a multi-level hierarchy, which does not involve the use of a traditional routing protocol.

Building a broker-based network is the most common approach for designing a distributed notification service. Each broker communicates with its neighbor using for subscription and publication. A P2P overlay network for multicast is a logical application level network that is built on top of a general network layer like IP unicast. The nodes that are part of the overlay network can route messages between each other through the overlay network. There is an overhead associated with using a logical network for routing since the logical topology does not necessarily mirror the physical topology. However, more sophisticated routing algorithms can be used and deployed since routing is implemented at the application level.

4 Centralized and Distributed Pub/Sub Systems

Content-based pub/sub systems operate either in a centralized manner or a decentralized manner. In a centralized pub/sub system, a centralized server stores all the subscriptions, maps events to the subscriptions, and delivers events to the matched subscribers. The main component of this architecture is the event dispatcher. This component records all subscriptions in the system. When a certain event is published, the event dispatcher matches it to all subscriptions in the system. When the incoming event verifies a subscription, the event dispatcher sends a notification to the corresponding subscriber.

Keeping a global image of subscriptions makes it easy for the sever to find matched subscribers, avoiding unnecessary event delivery. However, the server can easily be overloaded in a large-scale system with thousands or even millions of clients. In addition, such systems suffer from the problem of single point of failure. Thus, centralized pub/sub systems cannot provide high scalability and reliability, which prevents it from being applied to large-scale applications such as global video-conferencing. A distributed pub/sub system [26, 101] is a promising alternative driven by a variety of large-scale communication applications. The main difficulty in building distributed content-based systems is the design of an efficient distributed matching algorithm. Distributed content-based systems can be further classified into broker-based and DHT-based. Broker-based systems such as SIENA [26] depend on a small number of trusted brokers connected by a high bandwidth network [96]. The broker-based systems improve the scalability and reliability of the centralized systems to a certain extend by distributing load among a number of brokers. However, a failure of one broker may lead to a large number of state transfer operations during recovery. Thus, the systems also may not provide very high scalability and reliability in a large-scale environment.

To address the problems, more and more pub/sub systems resort to DHTs [81, 85, 94, 113] due to their high scalability, reliability, fault-tolerance and self-organizing.

DHTs have successfully been used in a number of application domains, such as distributed file systems [7, 37, 71, 86]. Most pub/sub systems, such as Scribe [39], relying on DHTs are topic-based because of DHTs' mapping policy between data and nodes. Recently, much research has been conducted in building content-based pub/sub systems on top of P2P systems [9, 77, 78, 96, 97, 106–108, 111, 114, 115].

4.1 Centralized Pub/Sub Systems

Traditional centralized systems [3, 7, 21, 33, 54, 57, 58, 64, 76, 88] use a centralized server that stores all the subscriptions in the system. The centralized server maps events to the subscriptions, and delivers events to the matched subscribers who are interested in the events. As indicated in [96], centralized systems have the advantage of retaining a global image of the system at all times, enabling intelligent optimizations during the matching process [11, 21, 47, 64, 76]. For example, Fabret et al. [47] proposed data structures and application-specific caching policies and query processing to support high rates of subscriptions and events in the system. Specifically, they used the data structures including a set of indexes, a predicate bit vector and a cluster vector to achieve efficient event matching that is based on clustering and maximizes temporal and spatial locality. However, restrictions have to be placed on subscriptions such that they must contain at least one equality predicate, sacrificing flexibility and expressiveness of subscriptions. Major disadvantages of centralized systems are the lack of scalability and fault-tolerance.

Elvin [50, 88] is a "pure" notification service in which producers send notifications to the service, which in turn sends them to consumers. The notifications describe events using a set of named attributes of simple data types and consumers subscribe to a "class" of events using a boolean subscription expression. When a notification is received at the service from a producer, it is compared to the consumers' registered subscription expressions and forwarded to those whose expressions it satisfies. Once producers are freed of the responsibility to direct notifications, the determination of the significance of a state change becomes less important: they can notify any potentially interesting information, and rely on the notification service to discard notifications of no (current) interest to consumers. While large volumes of unused notifications may be useful from a user's perspective, they consume network bandwidth. To overcome this problem, Elvin includes a quenching mechanism which allows producers to discard unneeded notifications without sending them to the server. In order to support organization-wide notification, the implementation of the notification service must cater for many client applications. A single Elvin server can effectively service thousands of clients (producers or consumers) and evaluate hundreds of thousands of notifications per second on moderate hardware platforms. Further, additional servers can be configured in a federation, sharing the load of notification delivery, providing wide-area scalability and ensuring fault-tolerance in the face of individual server failures.

Hanson et al. [58] introduced an algorithm for finding the matching predicates that is more efficient than the standard algorithm when the number of predicates is large. The authors focus on equality and inequality predicates on totally ordered domains. This algorithm is well-suited for database rule systems, where predicate-testing speed is critical. A key component of the algorithm is the interval binary search tree. It is designed to allow efficient retrieval of all intervals such as range predicates that overlap a point, while allowing dynamic insertion and deletion of intervals. Later on, Hanson et al. [57] proposed a way to develop a scalable trigger system. It is achieved with a trigger cache to use main memory effectively, and a memory-conserving selection predicate index based on the use of unique expression formats called expression signatures. A key observation is that if a very large number of triggers are created, many will have the same structure, except for the appearance of different constant values. When a trigger is created, tuples are added to special relations created for expression signatures to hold the trigger's constants. These tables can be augmented with a database index or main-memory index structure to serve as a predicate index. The design presented also uses a number of types of concurrency to achieve scalability, including token (tuple)-level, condition-level, rule action-level, and data-level concurrency.

Farsite [7] is a serverless distributed file system that logically functions as a centralized file server but whose physical realization is dispersed among a network of untrusted desktop workstations. Farsite is intended to provide both the benefits of a central file server (a shared namespace, location transparent access, and reliable data storage) and the benefits of local desktop file systems (low cost, privacy from nosy sysadmins, and resistance to geographically localized faults). Farsite provides file availability and reliability through randomized replicated storage; it ensures the secrecy of file contents with cryptographic techniques; it maintains the integrity of file and directory data with a Byzantine-fault-tolerant protocol; it is designed to be scalable by using a distributed hint mechanism and delegation certificates for pathname translations; and it achieves good performance by locally caching file data, lazily propagating file updates, and varying the duration and granularity of content leases. Pub/sub matching algorithms work in two phases. First, predicates are matched and then matching subscriptions are derived. Based on Ashayer et al.'s [11] observation that the domain types over which predicates are defined are often of fixed enumerable cardinality in practice, Adya et al. developed a table-based look-up scheme for fast predicate evaluation that finds all matching predicates for each type with one table lookup. They further proposed two DBMS-based matching algorithms and compare the better one with a special purpose pub/sub matching algorithm implementation. Their work showed that for application scenarios that require large subscription workloads and process many events, a DBMS-based solution is not a feasible alternative.

Petrovic et al. proposed S-ToPSS semantic pub/sub system that provides semantic matching [76]. For instance, the system returns notifications about "vehicles" or "automobiles" to a client who is interested in a "car" based on the semantics of the terms. The authors described three approaches, each adding more extensive semantic capability to the matching algorithms. The first approach allows a matching algorithm to match events and subscriptions that use semantically equivalent attributes-synonyms. The second approach uses additional knowledge about the relationships (beyond synonyms) between attributes and values to allow additional matches. More precisely, it uses a concept hierarchy that provides two kinds of relations: specialization and generalization. The third approach uses mapping functions which allow definitions of arbitrary relationships between schema and attribute values.

Liu et al. [64] pointed out that most existing pub/sub systems cannot capture uncertainty inherent to the information in either subscriptions or publications. In many situations, it is difficult to derive exact knowledge of subscriptions and publications. Moreover, especially in selective information dissemination applications, it is often more appropriate for a user to formulate his/her search requests or information offers in less precise terms, rather than defining a sharp limit. To address these problems, the authors proposed a new pub/sub model based on possibility theory and fuzzy set theory to process uncertainties for both subscriptions and publications.

Burcea et al. [21] identified the factors that affect the performance of a distributed pub/sub architecture supporting mobility; formalized mobility algorithms for distributed pub/sub systems and developed and evaluated optimizations that reduce the costs associated with supporting mobility in pub/sub systems. They focused on the "unicast" traffic generated to support mobile users, as opposed to the regular "multicast" traffic used for event dissemination to stationary clients.

4.2 Distributed Broker-Based Pub/Sub Systems

Content-based pub/sub allows fine-grained expressiveness of subscription, and thus is a more attractive solution for content dissemination. However, the design for content-based pub/sub systems is faced with two challenges that affect the performance of a content-based pub/sub network directly. The first challenge is the matching between subscriptions and events. Unlike the traditional multicast system where the addresses of destinations are known, the communication in content-based pub/-sub systems is based on the content of event publications and subscriptions. Thus, it is important to match the subscribers' subscriptions and publishers' events to identify the addresses of destinations. After the destinations are determined, the events need to be routed to the destinations. As indicated in [25], traditional group-based muticast techniques [35] cannot be readily used to route event to all destinations. This is because content-based subscriptions are usually highly diversified, and different events may satisfy the interests of widely varying sets of servers. In the worst case, the number of such sets can be exponential to the network size (2^n where n is the number of servers), and it is impractical to build a multicast group for each such set. The second challenge is how to efficiently route the matched events to the destinations. Therefore, an architecture design should efficiently match an event to subscriptions and meanwhile reduce the nodes participating in routing. In the last few years, a variety of broker-based pub/sub systems have been proposed in order to provide efficient and scalable pub/sub services. Broker-based systems depend on a small number of trusted brokers connected by a high bandwidth network. Brokers form an application level overlay and each broker stores subset of all subscriptions in the system. The overlay is managed by an administrator based on technical or administrative constraints. Examples of the broker-based pub/sub systems include SIENA [26–29], Gryphon [8, 73, 95], JEDI [36], Rebeca [48, 49, 70], Ready [55], Herald [22], MEDYM [25], Kyra [24], EDN [103] and link matching [15]. In the following, we present the details of the systems.

Kyra

To improve event routing efficiency, Cao and Singh [24] proposed Kyra routing scheme that uses content clustering to create multiple pub/sub networks each of which is responsible for a subset of the content space. The goal of Kyra is to reduce the implementation cost of the filter-based approach while still maintaining comparable network efficiency. Cao and Singh studied two major existing approaches for content-based pub/sub systems: filter-based approach, which performs content-based filtering on intermediate routing servers to dynamically guide routing decisions, and multicast-based approach, which delivers events through a few high-quality multicast groups that are pre-constructed to approximately match user interests. These approaches have different trade-offs in the routing quality achieved, the implementation cost and system load generated. The proposed Kyra carefully balanced these trade-offs by combining the advantages of content-based filtering and event space partitioning in the existing approaches to achieve better overall routing efficiency. The main idea is to construct multiple smaller routing networks, so that filter-based routing is implemented in each one with lower cost. Server load is reduced because each Kyra server is guaranteed to only participate in a small number of routing networks. This is achieved through strategically "moving" subscriptions between servers to improve content locality. Therefore, the effectiveness of Kyra is independent of data characteristics of pub/sub applications. Detailed simulation results show that Kyra significantly reduces the storage, processing and network traffic loads on pub/sub servers, while achieving network efficiency close to that of the filter-based approach. Kyra also balances routing load across the pub/sub service network.

SIENA

SIENA [26, 27] builds a symmetric spanning tree and each pub/sub server can be a publisher or subscriber. It selects the notifications that are of interest to clients and then delivers those notifications to the clients via access points. Mainly, SIENA addresses a key design challenge of maximizing expressiveness in the selection mechanism without sacrificing scalability of the delivery mechanism. SIENA focuses on the aspects that fundamentally affect scalability and expressiveness. In particular, SIENA has data model for notifications, the covering relations that formally define the semantics of the data model, the distributed architectures, and the processing strategies to exploit the covering relations for optimizing the routing of notifications. This work shows that the hierarchical architecture is suitable with low densities of clients that subscribe (and unsubscribe) very frequently, whereas the P2P architecture performs better when the total cost of communication is dominated by notifications. In situations where there are high numbers of ignored notifications (i.e., notifications for which there are no subscribers), the P2P architecture is also superior to the hierarchical architecture.

Based on SIENA, Carzaniga et al. [29] proposed a forwarding algorithm in content-based pub/sub networks. Forwarding in such a network amounts to evaluating the predicates stored in a router's forwarding table in order to decide to which neighbor router the message should be sent. The proposed algorithm is based on the general structure proposed for Le Subscribe systems and takes advantage of their fixed or limited number of output interfaces. A forwarding table is conceptually a map from predicates to interfaces of neighbor nodes where a predicate is a disjunction of filters, each one being a conjunctions of elementary conditions over the attributes of a message. The design of a forwarding algorithm involves the design of a forwarding table and of its processing functions. The proposed forwarding algorithm accelerates the decision making in situations where there are large numbers of predicates and high volumes of messages.

Later on, Carzaniga et al. [28] further proposed a routing scheme that can propagate predicates and necessary topological information in order to maintain loop-free and possibly minimal forwarding paths for messages. The routing scheme uses a combination of a traditional broadcast protocol and a content-based routing protocol. This scheme consists of a content-based layer superimposed over a traditional broadcast layer. The broadcast layer handles each message as a broadcast message, while the content-based layer prunes the broadcast distribution paths, limiting the propagation of each message to only those nodes that advertised predicates matching the message. To implement this two-layer scheme, a router runs two distinct routing protocols: a broadcast routing protocol and a content-based routing protocol. The first protocol processes topological information and maintains the forwarding state necessary to send a message from each node to every other node. The second protocol processes predicates advertised by nodes, and maintains the forwarding state that is necessary to decide, for each router interface, whether a message matches the predicates advertised by any downstream node reachable through that interface. This second protocol is based on a dual "push-pull" mechanism that guarantees robust and timely propagation of content-based routing information.

Gryphon

Gryphon [8, 73, 95] organizes a pub/sub network into a single-source tree and proposes a link matching algorithm to forward events towards directions of matching subscriptions. In Gryphon, the flow of streams of events is described via an information flow graph. The information flow graph specifies the selective delivery of events, the transformation of events, and the generation of derived events as a function of states computed from event histories. For this, Gryphon derives from and integrates the best features of distributed communications technology and database technology. The Gryphon approach augments the pub/sub paradigm with the following features: content-based subscription, in which events are selected by predicates on their content rather than by pre-assigned subject categories; event transformations, which convert events by projecting and applying functions to data in events; event stream interpretation, which allows sequences of events to be collapsed to a state and/or expanded back to a new sequence of events; and reflection, which allows system management through meta-events.

MEDYM

MEDYM [25] focuses on the problem of efficiently delivering events from the servers where they are published to the servers with matching subscriptions. In MEDYM, a matcher node matches an event to the subscriptions and obtains a destination list of the matched subscribers. Then, the event delivery message containing the destination list is routed through a dynamically generated dissemination tree with the help of topology knowledge. MEDYM does not rely on static overlay networks for event delivery. Instead, an event is matched against subscriptions early at the publishing server to identify destinations with matching subscriptions, and then sent to destination through a dynamically constructured multicast tree. This architecture achieves low computation cost in matching and high network efficiency in routing. MEDYM is distinguished by its dynamic multicast scheme to support the diversified routing need in pub/sub networks.

HYPER

HYPER [112] is a hybrid approach capable of minimizing both the matching and forwarding overhead within the pub/sub network and the delay experienced by clients receiving the content. It identifies a number of virtual groups by exploring common subscription interests among clients, and messages for each virtual group are only matched once at the group entry point. In addition, for each virtual group, the content delivery tree embedded in the underlying pub/sub network can benefit from short cutting forwarding-only paths.

EDN

EDN [103] partitions the content space subject to the restriction that the schema is fixed. For equality test, the attribute IDs and values are hashed to generate a key to locate the server managing it. For inequality tests, EDN uses an R-tree to decide offline how to assign subscriptions to processors, and requires each processor to maintain a complete map of this assignment. This approach is limited to small-scale systems with a fixed set of subscriptions, and it is also unclear whether it works efficiently for high dimensional content space.

Rebeca

In Rebeca [49, 70], the notification service relies on a network of brokers, which forward notifications according to filter-based routing tables. The topology of the system is constrained to be an acyclic and connected graph for simplicity reasons. The edges are point-to-point connections, forming an overlay network. This model simplifies the implementation and reasoning about communication characteristics. As indicated in [105], the major advantage of these systems is that the routing tables can direct the flow of notifications to only interested nodes. Each broker maintains a routing table which includes content-based filters. When routing, a notification only goes down a link if it is matched by a corresponding filter. The simplest form of routing is simple routing: active filters are simply added to the routing tables with the link they originated from. However, this makes the routing table sizes grow linearly with the number of subscriptions. Two methods can be used to address this problem. The first improvement method is to check and combine filters that are equal. In the second improvement method, if no cover can be found in a given set of filters, merging can be used to create new filters that cover existing ones. Only the resulting merged filter is forwarded to neighbor brokers, where it covers and replaces the base filters.

Later, Fiege et al. [48] pointed out that many works on notification services and many concrete systems such as Siena [26, 27] and JEDI [36] have informal semantics. In addition, in these systems, subscriptions are selected out of all published notifications without distinguishing producers. Any further distinctions are necessarily hard-coded into the communicating components, mixing application structure and component implementation and thereby defeating the very feature of event-based systems of loose coupling. To provide methodological support building pub/sub systems, Fiege et al. presented Rebeca modular design and implementation of an event system which supports scopes and event mappings, two new and powerful structuring methods that facilitate engineering and coordination of components in pub/sub systems. They give a formal specification of scopes and event-mappings within a trace-based formalism adapted from temporal logic.

Link Matching

Banavar et al. [15] proposed a multicast protocol, called link matching, within a network of brokers in a content-based pub/sub system, thereby showing that content-based pub/sub can be deployed in large or geographically distributed settings. With this protocol, each broker partially matches events against subscribers at each hop in the network of brokers to determine which brokers to send the message. Further, each broker forwards messages to its subscribers based on their subscriptions. Basically, the matching is based on sorting and organizing the subscriptions into a parallel search tree data structure, in which each subscription corresponds to a path from the root to a leaf. The matching operation is performed by following all those paths from the root to the leaves that are satisfied by the event. This data structure yields a scalable algorithm because it exploits the commonality between subscriptions as shared prefixes of paths from root to leaf. There is no additional information appended to the message headers in the method that match an event again all subscriptions. Further, at most one copy of a message is sent on each link. The disadvantages of the flooding approach are avoided as the message is only sent to brokers and clients needing the message.

Subscription Summaries

Triantafillou and Economides [101, 102] contributed the notion of subscription summaries, a mechanism appropriately compacting subscription information. They developed the associated data structures and matching algorithms. The proposed mechanism can handle event/subscription schemata that are rich in terms of their attribute types and powerful in terms of the allowed operations on them. The summarization structures of a broker's subscriptions and accompanying algorithms which operate on the summary structures match incoming events to the brokers with relevant subscriptions and maintain the subscriptions in the face of updates. The authors presented an algorithm to efficiently propagate subscription summaries to brokers. They also proposed an algorithm for the efficient distributed processing of incoming events, utilizing the propagated subscription summaries to route the events to brokers with matched subscriptions. They showed that the proposed mechanism is scalable with the bandwidth required to propagate subscriptions increasing only slightly even at huge-scales. The mechanism is significantly more efficient, up to orders of magnitude, depending on the scale, with respect to the bandwidth requirements for propagating subscriptions.

4.3 Distributed DHT-Based Pub/Sub Systems

DHT overlay networks [67, 69, 81, 85, 91, 94, 113] is a class of decentralized systems in the application level that partition ownership of a set of objects among participating nodes, and can efficiently route messages to the unique owner of any given object. Based on DHT overlay networks, a number of application level multicast systems have been proposed that can be used for topic-based pub/sub systems as well as content-based pub/sub systems. Examples of such systems include Scribe [39] based on Pastry, Bayeux [116] based on Tapestry and CAN-based multicast [82] based on CAN. Many content-based pub/sub systems based on DHTs have been proposed [9, 14, 56, 77, 78, 96, 97, 100, 106–108, 111, 114, 115]. DHT-based pub/sub systems inherit the distinguished features of DHT overlay networks including scalability, efficiency, reliability, fault-tolerance, self-organizing from the underlying DHT infrastructure.

4.3.1 Introduction of DHT Overlay Networks

A P2P system consists of peers that act as servers as well as clients in order to make full use of resources. Because of dynamic connections and decentralization characteristic, P2P systems have certain mechanisms to ensure efficient connection and communication. Such mechanisms include those handling nodes join, leave and failure, allocating files to the nodes, etc. In the system, no node is more important than any other and the nodes can communicate with each other. Each node maintains the location information of some other nodes. A node can send message to a chosen node or broadcast the message to several other nodes. Based on overlay topology, P2P systems can be classified into unstructured P2P systems and DHT systems (i.e., structured P2P systems). Unstructured P2P overlay networks such as Gnutella [53] and Freenet [52] do not have strict control over the topologies, and they do not assign responsibility for data to specific nodes. On the contrary, DHT overlay networks have strictly controlled topologies and the data placement and lookup algorithms are precise.

DHT overlay networks is a class of decentralized systems in the application level that partition ownership of a set of objects among participating nodes, and can efficiently route messages to the unique owner of any given object. The DHT overlay networks include Chord [94], CAN [81], Tapestry [113], Pastry [85], Kademlia [69], Symphony [67] and Cycloid [91]. In DHT overlay networks, each object is stored at one or more nodes selected deterministically by a uniform hash function. Specifically, each object or node is assigned an ID (i.e., key) that is the hashed value of the object or node IP address using consistent hash function [62]. An object is stored in a node whose ID closest or immediately succeeds to the object's ID, which is called the object's owner. Though these DHT systems have great differences in implementation, they all support a hash-table interface of `put(key,value)` and `get(key)` either directly or indirectly. `put(key,value)` stores an object in its owner node, and `get(key)` retrieves the object. Queries for the object will be routed incrementally to the node based on the P2P routing algorithm. Each node maintains a routing table recording $O(\log N)$ neighbors in an overlay network with N hosts. These structured systems are highly scalable as it make very large systems feasible; lookups can be resolved in $\log N$ overlay routing hops. DHT overlay networks are widely used for data sharing application. Different from pub/sub systems, content-delivery DHT overlay networks distribute data among nodes, and efficiently forward a data request to the data owner. DHTs' efficient data location enables efficient multicast communication. In addition, DHT overlay networks make pub/sub systems resilient in a dynamic environment where nodes join and leave continuously.

Chord

Chord uses a one-dimensional circular key space. The node responsible for the key is the node whose identifier most closely follows the key numerically; that node is called the key's successor. Each node in Chord maintains two sets of neighbors: a successor list of k nodes that immediately follow it in the key space and a finger list of $O(\log n)$ nodes spaced exponentially around the key space. The ith entry of the finger list points to the node that is 2^i away from the present node in the key space, or to that node's successor if that node is not alive. Therefore, the finger list is always fully maintained without any null pointer. Routing correctness is achieved with these two neighbor lists. A `lookup(key)` is, except at the last step, forwarded to the node closest to, but not past, the key. The path length is $O(\log n)$ since every lookup halves the remaining distance to the destination.

Pastry and Tapestry

Plaxton et al. [79] developed perhaps the first routing algorithm that could be scalably used for P2P systems. Tapestry and Pastry use a variant of the algorithm. The approach of routing based on address prefixes, which can be viewed as a generalization of hypercube routing, is common to all theses schemes. The routing algorithm works by correcting a single digit at a time in the left-to-right order. If node with ID 12345 receives a lookup query with key 12456, which matches the first two digits, then the routing algorithm forwards the query to a node which matches the first three digits (e.g., node 12467). To do this, a node needs to have, as neighbors, nodes that match each prefix of its own identifier but differ in the next digit. For each prefix (or dimension), there are many such neighbors (e.g., node 12467 and node 12478 in the above case) since there is no restriction on the suffix, i.e., the rest bits right to the current bit. This is the crucial difference from the traditional hypercube connection pattern and provides the abundance in choosing neighbors and thus a high fault resilience to node absence or node failure. In addition to these neighbors, each node in Pastry also contains a leaf set, which is the set of $|L|$ numerically closest nodes (half smaller, half larger) to the present node's ID, and a neighborhood set which is the set of $|M|$ geographically closest nodes to the present node.

CAN

CAN chooses its keys from a d-dimensional toroidal space. Each node is identified by a binary string and is associated with a region of this key space, and its neighbors are the nodes that own the contiguous regions. Routing consists of a sequence of redirections, each forwarding a lookup to a neighbor that is closer to the key. CAN has a different performance profile than the other algorithms; nodes have

$O(d)$ neighbors and path-lengths are $O(dN^{1/d})$ hops. Note that when d=log$N$, CAN has $O(\log N)$ neighbors and $O(\log N)$ path length like the other algorithms.

4.3.2 Early DHT-Based Pub/Sub Systems

Most initially proposed DHT-based pub/sub systems such as Scribe [39] and Bayeux [116] are essentially topic-based pub/sub systems. They do not directly support content-based pub/sub services. The systems employ rendezvous node model. A subscription or an event is mapped to a rendezvous node using the DHT key allocation policy. The rendezvous node disseminates events to subscribers using application level multicast. Systems built on Chord and Pastry map each multicast group number to a specific node and then have it act as a rendezvous node for that group. Joining a group means to lookup the rendezvous node and have the nodes on the lookup path record the route back to the new members. Systems built on CAN have the rendezvous node act as an entry point to a distinct overlay network composed only of the group members.

Scribe

Scribe is a scalable application level multicast infrastructure built on top of Pastry. Scribe relies on Pastry to create and manage groups and to build efficient multicast trees for the dissemination of messages to each group. In addition, Scribe provides best-effort reliability guarantees. Scribe is fully decentralized: all decisions are based on local information, and each node has identical capabilities. Each node can act as a multicast source, a root of a multicast tree, a group member, a node within a multicast tree, and any sensible combination of the above. Any Scribe node may create a group; other nodes can then join the group, or multicast messages to all members of the group. Scribe provides best-effort delivery of multicast messages, and specifies no particular delivery order. A node can create, send messages to, and join many groups. Groups may have multiple sources of multicast messages and many members. Scribe can support simultaneously a large numbers of groups with a wide range of group sizes, and a high rate of membership turnover.

A node creates a group with groupId. The groupId can be the hash value of the group's textual name concatenated with its creator's name. The rendezvous point of a group is the owner of the groupId of the group. Scribe creates a multicast tree, rooted at the rendezvous point, to disseminate the multicast messages in the group. The multicast tree is created using a scheme similar to reverse path forwarding [38]. Specifically, a `join` message is routed by Pastry towards the group's rendezvous point. Each node along the route checks its list of groups to see if it is currently a forwarder; if so, it accepts the node as a child, adding it to the children table. Otherwise, it creates an entry for the group, and adds the source node as a child in the associated children table. It then becomes a forwarder for the group by sending a `join` message to the next node along the route from the joining node to the rendezvous point. The original message from the source is then terminated. To enhance reliability, Scribe arranges each nonleaf node in the tree periodically sends a heartbeat message to its children. Furthermore, `forwardHandler(msg)` is invoked by Scribe before the node forwards a multicast message to make sure that parents can successfully forward the message.

SplitStream

SplitStream [30] is an application level multicast system built from Scribe for high-bandwidth data dissemination. Scribe works well only when the interior nodes are highly available. It poses a problem for application level multicast in P2P cooperative environments where peers contribute resources in exchange for using the service. SplitStream addresses this problem by striping the content across a forest of interior-node-disjoint multicast trees that distributes the forwarding load among all participating peers. For example, it is possible to construct efficient SplitStream forests in which each peer contributes only as much forwarding bandwidth as it receives. Furthermore, with appropriate content encodings, SplitStream is highly resilient to failures because a node failure causes the loss of a single stripe on average. To balance forwarding load over participating nodes with heterogeneous bandwidth constraints, SplitStream splits content into k stripes each of which corresponds to a Scribe multicast tree.

Bayeux

Bayeux [116] is another architecture for application layer multicast, where the end-hosts are organized into a hierarchy as defined by the Tapestry overlay location and routing system [113]. Similar to Scribe, Bayeux assigns a unique ID to each topic by using the tuple that uniquely names a multicast session (i.e., topic), and a secure one-way hashing function (such as SHA-1 [62]) to map it into a 160 bit identifier. The owner of the ID becomes the rendezvous point for this topic and the root node of the multicast tree. Clients that want to join a session must know the unique tuple that identifies that session. They can then perform the same operations to generate the file name, and query for it using Tapestry. For each topic, a multicast tree that is rooted at the rendezvous point is created by combining the paths from each subscriber to the rendezvous point. A level of the hierarchy is defined by a set of hosts that share a common suffix in their host IDs. These searches result in the session root node receiving a message from each interested listener, allowing it to perform the required membership operations. The events associated with the topic are disseminated along the corresponding multicast tree starting from the root. Such a technique was proposed by Plaxton et al. [79] for locating and routing to named objects in a network. Therefore, hosts in Bayeux maintain $O(b \log_b N)$ state and end-to-end overlay paths have $O(\log_b N)$ application level hops (b is a small constant).

CAN-based Multicast

CAN defines a virtual d-dimensional Cartesian coordinate space, and each overlay host owns a part of this space. Ratnasamy et al. [82] leveraged the scalable structure of CAN to define an application layer multicast scheme, in which hosts maintain $O(d)$ state and the path lengths are O($dN^{1/d}$) application level hops, where N is the number of hosts in the network. The CAN-based multicast scheme is capable of scaling to large group size without restricting the service model to a single source. Extending the CAN framework to support multicast comes at trivial additional cost, and obviates the need for a multicast routing algorithm because of the structured nature of CAN topologies. Given the deployment of a distributed infrastructure such as a CAN, the CAN-based multicast scheme offers the dual advantages of simplicity and scalability.

Reach

Reach [75] employs the rendezvous model, in which each node serves as a rendezvous point for those subscriptions with suffix matching the node's identifier. At a high level, the rendezvous service is the means by which subscriptions are stored in the network, and by which published messages are directed to "find" the subscriptions they match. This rendezvous node is then an entry point into a "subset tree" of nodes hosting other, more general subscriptions, and thus to which this message should also be routed. This tree is implemented in such a way that it offers join-and-leave flexibility and maximum efficiency as the nodes in the tree are nearby neighbors in the overlay. Reach employs a semantic overlay network and uses a Hamming-distance based routing scheme. Hamming-based encoding scheme defines an identifier hierarchy in which, a parent identifier contains at least all the attributes of a child identifier. This hierarchy is a fundamental concept in Reach and is the basis for content-based multicasting.

HOMED

HOMED [34] maintains a semantic overlay where each node's identifier is derived from its subscriptions. HOMED is suitable for large-scale pub/sub. HOMED prefers a mesh-like structure rather than a tree for a reliable and adaptive event dissemination tree. Moreover, it arranges a node to neighbor with the nodes whose interests are similar to its interest in the overlay network so that only interested nodes participate in disseminating an event. To ease construction and routing, HOMED organizes the overlay network based on the interest digest of each node rather than the complex selection predicate. HOMED can be used not only for flexible topic or type-based systems by nature, but also as a routing substrate for highly selective content-based systems. In HOMED, an event is delivered along the path of a binomial tree. Also, the subscribe/unsubscribe overhead is limited to $O(\log N)$.

4.3.3 DHT and Content Based Pub/Sub Systems

DHT systems are oblivious to the content of a file and use a uniform hash function on files' keys to distribute the files among the different peers. A file's key is the file name or the keyword that can distinguish the file. Therefore, on the one hand, DHTs provide exact-matching service. On the other hand, equality predicates and range predicates are expected when specifying subscriptions in pub/sub systems. Thus, to use DHT substrates for content-based pub/sub systems, a mechanism is needed that helps to distribute subscriptions and events among DHT nodes based on data content.

To tackle this problem, the works in [14, 56, 100] regard a subscription as a number of attributes and ranges. These works use each of the attributes and range constraints as a key to map the subscription to a number of overlay nodes. The single individual mapping for each attribute and value may lead to low scalability, especially when a subscription has many attributes and value ranges. To resolve the problem, some works [9, 77, 78, 96, 106–108, 111, 114, 115] use a scheme to derive a key or a small number of keys from a subscription for the mapping, while other works [97] combine the filter-based routing in broker-based model with the routing in DHT model.

Chord-based Systems

Triantafillou et al. [100] introduced one of the first content-based approximations where Chord DHT is employed as reliable routing infrastructure, so that they do not build a specific pub/sub overlay. The system distributes subscriptions on the Chord nodes based on the keys produced by hashing the attribute and its values. To do so, they employ the rendezvous model, in which a subscription is stored in a number of nodes based on the keys. If the subscription specifies a range over an attribute, the subscription would be stored on a number of nodes by hashing the attribute and each of its possible values within this range. Such systems suffer from the lack of scalability on high-dimensional contexts where a subscription has many attributes and values. The main drawback is that subscription installation and update are expensive due to the large number of nodes and messages potentially involved.

Later, Baldoni et al. [14] proposed a similar approach but, in this case, they used a particular mapping of events and subscriptions to keys from the DHT key space, instead of per-attribute mappings. They introduced a general form mapping that does not depend on the stored subscriptions which is called stateless mapping. It eliminates the need to propagate the knowledge about currently stored subscriptions. Specifically, the authors proposed three different methods for mapping pub/sub subscriptions and events to overlay keys: attribute-split, key space-split and selective-attribute. Furthermore, in order to increase the efficiency of the proposed solution, they proposed to enrich the existing overlay networks with one-to-many primitives, as well as to extend the infrastructure with notification buffering and range discretization capabilities.

Meghdoot

Meghdoot [56] is designed to adapt to highly skewed data sets, which is typical of real applications. Built upon CAN, Meghdoot adapts content-based pub/sub systems to DHT networks in order to provide scalable content delivery mechanisms while maintaining the decoupling between the publishers and the subscribers. Meghdoot stores subscriptions in a zone according to the coordinate determined by event attribute values. To do this, Meghdoot extends the traditional 1D-dimensional CAN to 2D-dimension CAN and relaxes the restrictions on subscriptions. A subscription defines a rectangular region in the D-attribute content space bounded by the minimal and maximal value specified. Unspecified attributes take the whole value range. The hyperrectangle is projected to a point in a 2D-dimension CAN constructed from the minimal and maximal values of the D-dimension rectangle. An event is then mapped to a rectangle in the 2D space, and the mapping is performed in a manner such that the rectangle covers all subscription points relevant to the event. This novel approach reduces the subscription matching problem into a range query operation in CAN. Considering skewed distributions of subscriptions and events in a real application, Meghdoot addresses the load balancing issue by zone splitting and zone replication. However, though it can support range subscriptions, it is still confined to numerical attributes and also can not handle skewed distributions efficiently. In addition, Meghdoot requires that the overlay dimension must be proportional to the number of event attributes, which may lead to very high DHT key space.

Scribe-based System

Tam et al. [96] proposed a content-based pub/sub system built from Scribe. In the approach, topics are automatically detected from the content of subscriptions and publications through the use of a schema, which is a set of guidelines for selecting topics. The schema is application-specific and can be provided by the application designer after some statistical analysis. The schemas are similar to database schemas used in RDBMS. This approach significantly increases the expressiveness of subscriptions compared to purely topic-based systems. However, this scheme does not fully provide the query semantics of a traditional content-based system. Queries are not completely free-form but must adhere to a predefined template. The system places some restrictions on subscriptions and thus sacrifices expressiveness in subscriptions. Moreover, issues of fault-tolerance in subscription storage have yet to be explored in the system, although fault-tolerance in DHT routing and multicast routing can be transparently handled by Pastry and Scribe, respectively.

Hermes

Hermes [77, 78] is an event-based middleware architecture that follows a type- and attribute-based pub/sub model. Hermes uses Pastry DHT routing substrate for installing content based filters close to the publishers. The Cambridge Event Architecture (CEA) [12, 66] is an event-based middleware that supports proper event typing. Hermes follows its approach by associating every event and subscription with an event type that is type-checked at runtime. A scalable routing algorithm using an overlay routing network is developed that avoids global broadcasts by creating rendezvous nodes. Fault-tolerance mechanisms that can cope with different kinds of failures in the middleware are integrated with the routing algorithm, resulting in a scalable and robust system.

CPAS

Considering node cooperation and multi-attribute feature of subscription, Ahullò et al. [9] proposed CPAS, which employs the rendezvous model in order to meet both, events and subscriptions. The system defines a certain set of nodes from the DHT as rendezvous nodes. The rendezvous nodes are responsible of matching events against subscriptions and starting the notification process. Additionally, these rendezvous nodes are selected deterministically, so that the node in DHT responsible for a given key then becomes the rendezvous node. Due to the DHT properties, the chosen node will be globally agreed upon by all nodes. Thus, every node can use the P2P routing substrate to send messages to this rendezvous node. The rendezvous model enables the system to avoid the construction of a specific overlay to disseminate events in a proper way. CAPS employs an order preserving hash function (OPHF) to deterministically map conjunctive predicates from every subscription into a set of keys and every event into a key, in order to deal naturally with multi-dimensional domains, and multiple sources cooperating within the system.

Ferry

Ferry [114] provides a preliminary study of exploiting the embedded trees in DHTs to deliver events. It is designed based on Chord and aims to host any and many content-based pub/sub services. That is, any pub/sub service with a unique scheme can run on top of Ferry, and multiple pub/sub services can coexist on top of Ferry. For each pub/sub service, Ferry does not need to maintain or dynamically generate any dissemination tree. Instead, it exploits the embedded trees in the underlying DHT to deliver events. Ferry can support a pub/sub scheme with a large number of event attributes. Specifically, a subscriber chooses an attribute from all attributes of a subscription whose consistent hash value is equal to or most immediately precede the subscriber's ID. It then maps the subscription to a rendezvous node based on the consistent hash value. Thus, a tree is formed by the underlying DHT links thereby imposing no additional construction of maintenance cost. When a node wants to publish an event, the event is first directed to the rendezvous node where the event is matched to the subscriptions. Once those subscriptions matching the event are identified, the event is then delivered to the corresponding subscribers by using Ferry's event delivery algorithm. In the delivery algorithm, all the event delivery messages to those subscribers who share common ancestor nodes on the tree are aggregated into one single message along the path from the root node to their lowest common ancestor node. To deal with skewed distribution of subscriptions and events, Ferry uses one-hop subscription push and attribute partitioning to balance load. In the one-hop subscription push algorithm, a rendezvous node pushes the subscriptions corresponding to one of the nodes' neighbors in its routing table to the neighbor. In the attribute partitioning algorithm, a value range is partitioned into a number of ranges.

Eferry and HyperSub

Eferry [108], HyperSub [107] and the work in [106] are enhanced systems based on Ferry. The objective of Eferry [108] is to ensure an appropriate amount of rendezvous point nodes in the system and load distribution among them. Eferry achieves this goal with three methods: (1) a novel subscription installation algorithm to choose certain rendezvous point nodes which are evenly distributed in the ID space. (2) ID space partitioning and attributes grouping schemes designed to flexibly adjust the amount of rendezvous point nodes as well as their load. (3) a self-adaptive load balancing algorithm with dynamic ID space split-merge to make sure that no node is unduly loaded. HyperSub [107] and the work in [106] use a locality-preserving hashing mechanism to partition and map the content space to nodes. Subscriptions and events are mapped to the corresponding nodes for efficiently matching. The systems have an efficient event delivery algorithm which exploits the embedded trees in the underlying DHT to deliver events to the corresponding subscribers. In addition, the systems have light-weighted load balancing mechanisms to adjust the load among peers. The load balancing mechanism includes space mapping rotation, content space transformation and dynamic subscriptions migration algorithms.

PRESS

PRESS [115] distinguishes itself from Ferry by proposing a new architecture that aims to preserve subscription locality in subscription management, minimize event matching load, balance load across nodes, and offer efficient and scalable event delivery. The framework of PRESS is based on the three key mechanisms: Subscription Organization Mechanism (SOM), Publication and Matching Mechanism (EPMM) and Event Delivery Mechanism (EDM). SOM uses K-D tree techniques [17] to organize subscriptions in a hierarchical tree manner, and stores the subscriptions only on leaf nodes. SOM preserves locality of subscriptions, i.e., similar/relevant subscriptions are stored on a (or a small number of adjacent) leaf node(s). Each leaf node is responsible for roughly the same number of subscriptions, ensuring load balance across leaf nodes. SOM layers the tree structure on top of a DHT, by which each tree node is hosted by a DHT node and the tree inherits fault-resilience and self-organizing properties of the underlying DHT. Subscription installation is a process of tree navigation from the tree root to the corresponding leaf node(s). The subscription installation may involve multiple overlay hops since the tree spans the DHT overlay, thereby incurring high latency. In addition, every installation goes through the root, creating a potential bottleneck. Hence, PRESS uses K-D tree-lookaside cache at client/subscriber side to alleviate the problems. EPMM allows event publishers to publish an event along the K-D tree to the leaf node that stores the subscriptions relevant to the event. The leaf node then matches the event to the subscriptions and starts delivering the event to the matched subscribers. Similar to subscription installation, event publication could incur high publication latency and create a potential bottleneck on the tree root node. To alleviate the problems, the K-D tree-lookaside cache is employed at the client/publisher side. EDM is virtually maintenance-free. It exploits embedded trees inherent in the underlying DHT to deliver events, thereby eliminating the cost of multicast-tree construction and maintenance. After a leaf node matches an event to the subscriptions stored on it, the leaf node multicasts the event through the corresponding DHT links of its DHT host node. The event is then disseminated along the embedded tree rooted at the DHT node hosting the leaf node, and finally reaches each subscriber. EDM aggregates messages along event dissemination paths, thus reducing the number of event delivery messages and bandwidth consumption. Moreover, exploiting DHT links for event delivery, EDM has three major advantages: (1) The underlying DHT maintenance messages could be piggybacked onto the event delivery messages to reduce the DHT maintenance cost. (2) Proximity neighbor selection in the underlying DHT, as a means of improving routing performance, makes event dissemination along the embedded tree proximity-aware, achieving efficient event delivery performance. (3) The fault-tolerance and self-organizing nature of DHT overlays makes event delivery along the DHT links resilient to node/link failures.

Brushwood-based System

The content-based pub/sub model has been adopted by many services to deliver data between distributed users based on application-specific semantics. Two key issues in such systems, the semantic expressiveness of content matching and the scalability of the matching mechanism, are often found to be in conflict due to the complexity associated with content matching. To address this problem, Zhang et al. [111] presented a content-based pub/sub architecture based on Brushwood P2P matching trees [110]. The authors indicated that the content-based systems have more complex subscription structures that impede the workload partition than topic-based systems due to three reasons. The first reason is the high dimensionality of the content space in which a setting involves a large number of attributes. The second reason is type flexibility which means that attributes may have various types that require different filtering tests. The third reason is skewed data distribution, which could create a load imbalance in the system that throttles the scalability. The system achieves scalability by partitioning the responsibility of event matching to self-organized peers while allowing customizable matching functionalities. Specifically, the authors proposed a P2P architecture that achieves high scalability and generality. The architecture addresses the expressiveness problem with a modular matching tree structure. This tree organizes the subscriptions into hierarchical groups based on their similarity. It supports flexible schemas and multiple attribute types in subscriptions and events, and allows customization of new attributes and filtering types. This matching tree is distributed in a P2P system where each peer processor manages a small fragment of the tree. They maintain the distributed tree by peer-wise communications without global coordination. Events can enter the system from any processor. A decentralized tree navigation algorithm is used to forward the events to those tree fragments that may contain matching subscriptions. In experiments, the proposed system demonstrates high scalability. Specifically, the distributed event matching only visits a small number of processors, processors maintain a small amount of state about peers, and the workload is well-balanced across the processor set.

Combination of Rebeca and Chord

The system proposed in [97] is another content-based pub/sub system built on top of a dynamic Chord P2P overlay network. Both filter updates (e.g., due to subscribing and un-subscribing) and event routing use a broadcasting algorithm. The main advantage of the proposed system is the unique combination of the high expressiveness of content-based filters in Rebeca and the scalability and fault tolerance of Chord P2P system. It helps to remove the single bottleneck and point-of-failure of using only one tree for notifications and filter updates. To avoid introducing routing cycles within a more general redundant graph, the system selects for each notification a spanning subtree of the entire graph. However, to balance the network congestion and reduce single points of failure, the system uses a different tree for every broker. That is, each broker is at the root of its own distinct tree for delivering a published notification. This allows the system to use a generalization of the pub/sub routing strategy. During routing, the system provides a test to assure forwarding is only along those edges which are in the subtree. To provide the routing algorithm with an understanding of how to select the edges for a subtree, the system incorporates a topology component. Furthermore, the system has two components that maintain the structure of the graph and the filters to enhance system robustness when brokers change and fail. Separating the components ensures that the network self-organizes to maintain the optimal topology and can survive simultaneous failure of up to half of its nodes. Because the system delivers via binomial trees, message delivery paths are logarithmically bounded.

Table 2 illustrates a survey of current pub/sub systems based on the classifications.

Table 2 Survey of pub/sub systems.

Centralized systems	**Distributed systems**			
	Broker-based		**DHT-based**	
Content-based	**Topic-based**	**Content-based**	**Topic-based**	**Content-based**
TIB/RV [72]	TIB/RV [72]	SIENA [26–29]	Scribe [39]	Meghdoot [56]
CORBA-NS [5]	JEDI [36]	Gryphon [8, 73, 95]	Bayuex [116]	Hermes [77, 78]
Narada [87]		Rebeca [48, 49, 70]	NICE [16]	CPAS [9]
Elvin [50]		Kyra [24]	SplitStream [30]	Ferry [114]
Farsite [7]		MEDYM [25]	Reach [75]	Eferry [108]
S-ToPSS [76]		Ready [55]	HOMED [34]	HyperSub [107]
JMS [3]		Herald [22]		PRESS [115]
		EDN [103]		Brushwood-based [111]

5 Summary and Challenges

In the last years, a growing attention has been paid to the pub/sub communication paradigm as a means for disseminating events through distributed systems on wide-area networks. This chapter has provided a detailed introduction of pub/sub systems, and has examined all aspects of pub/sub systems including their goals, properties, strategies and classification. To survey and compare different pub/sub systems, we introduced three classification criteria: subscription model, routing and topology. Based on the subscription model, the pub/sub systems can be classified into topic-based, content-based and type-based. Based on routing, the pub/sub systems can be classified into filter-based and multicast-based. Based on topology, the pub/sub systems can be classified into centralized-based and distributed-based, which is further classified into broker-based and DHT-based. A comprehensive review of research works of pub/sub systems focusing on distributed networks has been presented, along with an in-depth discussion of their pros and cons.

We conclude this chapter with discussion about a number of open issues for building pub/sub systems.

- **Tradeoff between the accuracy and efficiency.** Topic-based pub/sub systems cannot provide high accuracy since a node may receive events it is not interested in. On the other hand, highly fined-grained content-based systems lead to high cost for mapping between subscriptions and events as well as node communication. A mechanism that can combine the advantages of both types while overcoming their drawbacks is expected.

- **Proximity.** Mismatch between logical proximity abstraction derived from overlay networks, and physical proximity information in reality is a major obstacle for the deployment and performance optimization issues for pub/sub applications. Most current pub/sub systems fail to take into account the proximity to reduce the node communication cost.
- **Heterogeneity.** With the increasing emergence of various end devices equipped with networking capability, coupled with the diverse network technology development, the heterogeneity of participating nodes of a practical pub/sub system is pervasive. Their distinct properties, including computing ability, differ greatly and deserve serious consideration for the construction of a real efficient widely-deployed application. Most current pub/sub system considering load balance fail to take into account the heterogeneity.
- **Mobility.** With the increasing popularity of wireless communication networks and mobile handheld devices, it becomes an inevitable trend that the pub/sub systems will be applied to the mobile wireless networks. Currently, there are few works devoted to the development of a pub/sub system in a mobile environment. One challenge is how to deal with node mobility.

References

1. Gryphon web site. http://www.research.ibm.com/gryphon/.
2. Publish/subscribe. http://en.wikipedia.org/wiki/Publish/subscribe.
3. Sun microsystems. Java Message Service API, Sun Microsystems. 2003.
4. Vitria. http://www.vitria.com/.
5. Object management group. corba notification service specification, version 1.0.1. omg document formal/2002-08-04, 2002.
6. S. Scipioni, A. Corsaro, and L. Querzoni. Quality of service in publish/subscribe. Technical report, Università di Roma La "Sapienza", 2006.
7. A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceus, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer. FARSITE: Federated, avaiable, and reliable storage for an incompletely trusted environment. In *Proc. of the Fifth USENIX Symposium on Operating Systems Design and Implementation*, December 2002.
8. M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra. Matching events in a content-based subscription system. In *Proc. of the Eighteenth ACM Symposium on Principles of Distributed Computing*, 1999.
9. J. P. Ahullò, P. G. Lòpez, and Antonio F. G. Skarmeta. Caps: Content-based publish/subscribe services for peer-to-peer systems. In *Proceedings of 2nd International Conference on Distributed Event-Based Systems (DEBS)*, July 2008.
10. M. Altinel and M. Franklin. Efficient Filtering of XML Documents for Selective Dissemination of Information. *VLDB Journal*, pages 53–64, 2000.
11. G. Ashayer, H. K. Y. Leung, and H. A. Jacobsen. Predicate matching and subscription matching in publish/subscribe systems. In *Proc. of Workshop on Distributed Event-Based Systems (DEBS)*, pages 539–546, 2002.

12. J. Bacon, A. Hombrecher, C. Ma, K. Moody, and W. Yao. Event storage and federation using odmg. In *Proc. of the 9th Int. Workshop on Persistent Object Systems (POS9)*, pages 265–281, Sept. 2000.
13. S. Baehni, P. Th. Eugster, and R. Guerraoui. Data-aware multicast. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN)*, pages 233–242, 2004.
14. R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg. Content-based publish-subscribe over structured overlay networks. In *Proc. ICDCS*, pages 437–446, July 2005.
15. G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In *Proceedings of the 19th IEEE ICDCS*, pages 262–272, June 1999.
16. S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proc. of ACM SIGCOMM'02*, pages 205–217, 2002.
17. J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
18. K. P. Birman. The process group approach to reliable distributed computing. *Communications of the ACM*, 36(12):36–53, Dec 1993.
19. K. P. Birman and T. A. Joseph. Exploiting virtual synchrony in distributed systems. *Operating Systems Review*, pages 123–138, 1987.
20. S. Bittner and A. Hinze. On the benefits of non-canonical filtering in publish/subscribe systems. In *Proceedings of the International Workshop on Distributed Event-Based Systems (ICDCS/DEBS)*, 2005.
21. I. Burcea, V. Muthusamy, M. Petrovic, H. A. Jacobsen, and E. de Lara. Disconnected operations in publish/subscribe. *Proc. of IEEE Mobile Data Management*, 2004.
22. L. F. Cabrera, M. Jones, and M. Theimer. Herald: Achieving a global event notification service. In *Proc. of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, May 2001.
23. A. Campailla, S. Chaki, E. M. Clarke, S. Jha, and H. Veith. Efficient filtering in publishsubscribe systems using binary decision diagrams. In *Proceedings of The International Conference on Software Engineering*, pages 443–452, 2001.
24. F. Cao and J. P. Singh. Efficient event routing in content-based publish/subscribe service networks. In *Proceedings of INFOCOM*, volume 2, pages 929–940, March 2004.
25. F. Cao and J. P. Singh. MEDYM: match-early and dynamic multicast for content-based publish-subscribe service networks. In *Proceedings of the 4th international workshop on distributed event-based systems*, pages 370–376, 2005.
26. A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving scalability and expressiveness in an Internet-scale event notification service. In *Proc. of ACM Symp. on Principles of Distributed Computing (PODC)*, pages 219–227, 2000.
27. A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Transactions on Computer Systems*, 19(3):332–383, 2001.
28. A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A routing scheme for content-based networking. In *Proceedings of IEEE INFOCOM*, pages 918–928, March 2004.
29. A. Carzaniga and A. L. Wolf. Forwarding in a content-based network. In *Proceedings of ACM SIGCOMM*, pages 163–174, 2003.
30. M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP-19)*, October 2003.
31. M. Castro, M. B. Jones, A-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman. An evaluation of scalable application-level multicast built using peer-to-peer overlays. In *Proc. of IEEE Conference on Computer Communications (INFOCOM'03)*, March 2003.
32. Y. Chawathe. Scattercast: An architecture for internet broadcast distribution as an infrastructure service. ph.d. thesis. Technical report, University of California, Berkeley, 2000.
33. J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: A scalable continuous query system for Internet databases. In *Proceedings of the 2000 ACM SIGMOD*, pages 379–390, 2000.

34. Y Choi, K. Park, and D. Park. HOMED: a peer-to-peer overlay architecture for large-scale content-based publish/subscribe systems. In *Proceedings of the third international workshop on distributed event-based systems (DEBS)*, pages 20–25, May 2004.
35. Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM SIGMETRICS'2000*, January 2000.
36. G. Cugola, E. D. Nitto, and A. Fuggetta. The JEDI Event-based Infrastructure and its Application to the Development of the OPSS WFMS. *IEEE Transactions on Software Engineering*, 2001.
37. F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stocia. Wide-area cooperative storage with CFS. In *Proc. of the 18th ACM Symp. on Operating Systems Principles (SOSP-18)*, October 2001.
38. Y. K. Dalal and R. Metcalfe. Reverse path forwarding of broadcast packets. *Communications of the ACM*, 21(12):1040–1048, Dec. 1978.
39. P. Druschel, M. Castro, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. In *IEEE Journal on Selected Areas in Communications*, 2002.
40. V. S. W. Eide, F. Eliassen, O. Lysne, and O. Granmo. Extending content-based publish/-subscribe systems with multicast support. Technical report, Simula Research Laboratory, 2003.
41. G. Eisenhauer. The ECho event delivery system. Technical Report GITCC-99-08, College of Computing, Georgia Institute of Technology, June 1999. http://www.cc.gatech.edu/tech reports.
42. A. El-Sayed, V. Roca, I. Rhone-Alpes, and L. Mathy. A survey of proposals for an alternative group communication service. *IEEE Network magazine.*, 2003.
43. P. T. Eugster and R. Guerraoui. Content-based publish/subscribe with strucutural reflection. In *Proc. of the 6th USENIX Conf. on Object-Oriented Technologies and Systems (COOTS01)*, Jan 2001.
44. P. T. Eugster, R. Guerraoui, and J. Sventek. Type-based publish/subscribe. Technical report, EPFL, Lausanne, Switzerland, June 2000.
45. P. Th. Eugster, P. Felber, R. Guerraoui, and S. B. Handurukande. Event Systems: How to Have Your Cake and Eat It Too. In *Proceedings of the International Workshop on Distributed Event-Based Systems (DEBS)*, 2002.
46. P. Th. Eugster, R. Guerraoui, and Ch. H. Damm. On Objects and Events. In *Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications*, 2001.
47. F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *Proceedings of ACM SIGMOD*, volume 30, pages 115–126, 2001.
48. L. Fiege, G. Mühl, and F. Gärtner. Modular event-based systems. *The Knowledge Engineering Review*, 17(4):55–85.
49. L. Fiege, G. Mühl, and F. Gärtner. A Modular Approach to Building Event-Based Systems. In *Proceedings of the ACM Symposium on Applied Computing*, 2002.
50. G. Fitzpatrick, T. Mansfield et al. Instrumenting and Augmenting the Workaday World with a Generic Notification Service called Elvin. In *Proc. of European Conference on Computer Supported Cooperative Work (ECSCW)*, 1999.
51. P. Francis. Yoid: Your own internet distribution. Technical report, ACIRI, 2000. http://www.aciri.org/yoid/.
52. The freenet home page. freenet.sourceforge.net, www.freenetproject.org.
53. Gnutella home page. http://www.gnutella.com.
54. J. Gough and G. Smith. Efficient recognition of events in a distributed system. In *Proc. of the 18th Australasian Computer Science Conference*, 1995.
55. R. Gruber, B. Krishnamurthy, and E. Panagos. The architecture of the READY event notification service. In *Proceedings of the 19th Middleware Workshop*, 1999.

56. A. Gupta, O. D. Sahin, D. Agrawal, and A. E. Abbadi. Meghdoot: content-based publish/-subscribe over P2P networks. In *Proceedings of the 5th International middleware conference of ACM/IFIP/USENIX*, pages 370–376, Oct. 2005.
57. E. N. Hanson, C. Carnes, L. Huang, M. Konyala, L. Noronha S. Parthasarathy, J. B. Park, and A. Vernon. Scalable trigger processing. In *Proceedings of the 15th ICDE*, pages 266–275, 1999.
58. E. N. Hanson, M. Chaabouni, C.-H. Kim, and Y.-W. Wang. A predicate matching algorithm for database rule systems. In *Proc. of SIGMOD*, 1990.
59. D. A. Helder and S. Jamin. End-host multicast communication using switch-tree protocols. In *In Proceedings of the Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems (GP2PC)*, 2002.
60. S. Jain, R. Mahajan, D. Wetherall, G. Borriello, and S. D. Gribble. Scalable self-organizing overlays. technical report uw-cse 02-02-02. Technical report, University of Washington, 2002.
61. J. Jannotti, d. Gifford, K. Johnson, and M. Kaashoek. Overcast: Reliable multicasting with an overlay network. In *Proc. of the Fourth USENIX Symposium on Operating Systems Design and Implementation*, October 2000.
62. D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. *Proceedings of the 29th Annual ACM Symposium on Theory of Computing(STOC)*, pages 654–663, May 1997.
63. J. Liebeherr and M. Nahas. Application-layer multicast with delaunay triangulations. In *Global Internet Symposium, IEEE Globecom 2001 Conference*, 2001.
64. H. Liu and H. A. Jacobsen. Modeling uncertainties in publish/subscribe. In *Proc. of Conf. on Data Engineering*, 2004.
65. Y. Liu and B. Plale. Survey of publish subscribe event systems. Technical report, Indiana University, 2003.
66. C. Ma and J. Bacon. Cobea: A corba-based event architecture. In *Proc. of the 4th USENIX Conf. on O-O Tech. and Systems*, pages 117–131, Apr. 1998.
67. G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed Hashing in a Small Wold. In *Proc. of the 4th USENIX Symposium on Internet Technologies and Systems (USITS'03)*, 2003.
68. L. Mathy, R. Canonico, and D. Hutchison. An overlay tree building control protocol. In *3rd International Workshop Networked Group Communications*, 2001.
69. P. Maymounkov and D. Mazires. Kademlia: A Peer-to-peer Information Systems Based on the XOR Metric. In *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002.
70. G. Muhl. Generic Constraints for Content-Based Publish/Subscribe. In *Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS)*, 2001.
71. A. Muthitacharoen, R. Morris, T. M. Gil, and B. Chen. Ivy: A read/write peer-to-peer file system. In *Proc. of the Fifth USENIX Symposium on Operating Systems Design and Implementation*, December 2002.
72. B. Oki, M. Pfluegel, A. Siegel, and D. Skeen. The information bus - an architecture for extensive distributed systems. In *Proceedings of the ACM Symposium on Operating Systems Principles*, December 1993.
73. L. Opyrchal, M. Astley, R. E. Strom J. Auerbach, G. Banavar, and D. C. Sturman. Exploiting ip multicast in content-based publish- subscribe systems. In *Proc. of Middleware*, 2000.
74. D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. In *Proc. of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS'01)*, March 2001.
75. G. Perng, C. Wang, and M. K. Reiter. Providing content-based services in a peer-to-peer environment. In *Proceedings of the third international workshop on distributed event-based systems (DEBS)*, pages 74–79, May 2004.

76. M. Petrovic, I. Burcea, and H. A. Jacobsen. S-ToPSS: Semantic Toronto publish/subscribe system. In *Proc. of Conf. on Very Large Data Bases*, pages 1101–1104, 2003.
77. P. R. Pietzuch and J. Bacon. Peer-to-peer overlay broker networks in an event-based middleware. In *Proc. of Workshop on DEBS*, 2003.
78. P. R. Pietzuch and J. M. Bacon. Hermes: A Distributed Event-Based Middleware Architecture. In *Proceedings of 1st International Workshop on Distributed Event-Based Systems (DEBS)*, pages 611–618, July 2002.
79. C. Plaxton, R. Rajaraman, and A. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proc. of ACM SPAA*, June 1997.
80. R. Preotiuc-Pietro, J. Pereira, F. Llirbat, F. Fabret, K. Ross, and D. Shasha. Publish/subscribe on the web at extreme speed. In *Proc. of ACM SIGMOD Conf. on Management of Data*, 2000.
81. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of ACM SIGCOMM'01*, pages 329–350, 2001.
82. S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *In Proceedings of NGC*, 2001.
83. A. Riabov, Z. Liu, J. Wolf, P. Yu, and L. Zhang. Clustering Algorithms for content-based publication-subscription systems. In *Proc. of ICDCS*, 2002.
84. V. Roca and A. El-Sayed. A host-based multicast(hbm) solution for group communications. In *1st IEEE International Conference on Networking(ICN01)*, July 2001.
85. A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms(Middleware)*, 2001.
86. A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale persistent peer-to-peer storage utility. In *Proc. of the 18th ACM Symp. on Operating Systems Principles (SOSP-18)*, October 2001.
87. A. Rowstron, P. Druschel, and M. Castro. Scribe: The design of a large-scale event notification infrastructure. In *Proc. of the 3th Int. Workshop on Networked Group Communications*, 2001.
88. B. Segall and D. Arnold. Elvin has left the building: a publish/subscribe notification service with quenching. In *Proceedings of AUUG*, pages 243–255, sep. 1997.
89. B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps. Content Based Routing with Elvin4. In *Proceedings of AUUG2K*, June 2000.
90. R. Shah, R. Jain, and F. Anjum. Efficient Dissemination of Personalized Information Using Content-Based Multicast. In *Proceedings of IEEE Infocom*, 2002.
91. H. Shen, G. Chen, and C. Xu. Cycloid: A scalable constant-degree p2p overlay network. *Journal of Performance Evaluation's Special Issue on Peer-to-Peer Networks*, (3):195–216, 2006.
92. A. Slominski, Y. Simmhan, A. L. Rossi, M. Farrellee, and D. Gannon. Xevents/xmessages: Application events and messaging framework for grid. Technical report, Indiana University, 2001.
93. C. Snoeren, K. Conley, and D. K. Gifford. Mesh based content routing using XML. In *Proc. of SOSP*, 2001.
94. I. Stoica, R. Morris, D. Liben-Nowell, M. F. Kaashoek, D. Karger, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Trans. on Networking*, August 2002.
95. R. Strom, G. Banavar, T. Ch, M. Kaplan, K. Miller, B. Mukherjee, D. Sturman, and M. Ward. Gryphon: An information flow based approach to message brokering. In *Proc. of the International Symposium on Software Reliability Engineering*, 1998.
96. D. Tam, R. Azimi, and H.-A. Jacobsen. Building content-based publish/subscribe systems with distributed hash tables. In *Proceedings of the international workshop on databases, information systems and peer-to-peer computing*, September 2003.
97. W. W. Terpstra, S. Behnel, L. Fiege, A. Zeidler, and A. P. Buchmann. A peer-to-peer approach to content-based publish/subscribe. In *Proc. of Workshop on DEBS*, 2005.

98. Tibco software inc. tibco rendezvous faq, 2003. http://www.tibco.com/solutions/products/active enterprise/rv/faq.jsp.
99. D. Tran, K. Hua, and T. Do. Zigzag: An efficient peer-to-peer scheme for media streaming. In *Proc. of IEEE Conference on Computer Communications (INFOCOM'03)*, 2003.
100. P. Triantafillou and I. Aekaterinidis. Content-based publish-subscribe over structured P2P networks. In *Proceedings of the third international workshop on distributed event-based systems (DEBS)*, pages 104–109, May 2004.
101. P. Triantafillou and A. Economides. Subscription summaries for scalability and efficiency in publish/subscribe. In *Proc. of Workshop on Distributed Event-Based Systems*, pages 619–624, 2002.
102. P. Triantafillou and A. Economides. Subscription summarization: a new paradigm for efficient publish/subscribe systems. In *Proceedings of the 24th IEEE ICDCS*, pages 562–571, 2004.
103. Y. Wang, L. Qiu, D. Achlioptas, G. Das, P. Larson, and H. J. Wang. Subscription partitioning and routing in content-based publish/subscribe networks. In *Proceedings 16th International Symposium on DIStributed Computing (DISC)*, October 2002.
104. T. Wong, R. Katz, and S. McCanne. An evaluation of preference clustering in largescale multicast applications. In *Proc. of IEEE INFOCOM*, March 2000.
105. X. Yang and Y. Zhu. A peer-to-peer approach to content-based publish/subscribe. In *Proceedings of the 2nd international workshop on Distributed event-based systems table of contents*, pages 1–8, 2003.
106. X. Yang and Y. Zhu. A DHT-based Infrastructure for Content-based Publish/Subscribe Services. In *Proceedings of P2P*, 2007.
107. X. Yang, Y. Zhu, and Y. Hu. A large-scale and decentralized infrastructure for content-based publish/subscribe services. In *Proceedings of the 36th International Conference on Parallel Processing (ICPP)*, 2007.
108. X. Yang, Y. Zhu, and Y. Hu. Scalable content-based publish/subscribe services over structured peer-to-peer networks. In *Proceedings of the 15th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2007.
109. B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. In *Proc. of IEEE Conference on Computer Communications (INFOCOM'02)*, 2002.
110. C. Zhang, A. Krishnamurthy, and O. Y. Wang. Brushwood: Distributed trees in peer-to-peer systems. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 47–57, 2005.
111. C. Zhang, A. Krishnamurthy, O. Y. Wang, and J. P. Singh. Combining flexibility and scalability in a peer-to-peer publish/subscribe system. In *Proc. of Middleware*, 2005.
112. R. Zhang and Y. C. Hu. HYPER: a hybrid approach to efficient content-based publish/-subscribe. In *Proceedings of international conference on distributed computing systems (ICDCS)*, June 2005.
113. B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry:an infrastructure for fault-tolerant widearea location and routing. Technical Report UCB/CSD-01-1141, Computer Science Division, UC Berkeley, April 2001.
114. Y. Zhu and Y. Hu. Ferry: an P2P-based architecture for content-based publish/subscribe services. *IEEE Trans Parallel Distrib Syst*, 18(5):672–685, 2007.
115. Y. Zhu and H. Shen. An efficient and scalable framework for content-based publish/subscribe systems. *Peer-to-Peer Networking and Applications*, 1(1):3–17, March 2008.
116. S. Zhuang, B. Zhao, A. Joseph, R. Kotz, and J. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proc. of the Eleventh Intl. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 2001.

Supporting Collaboration and Creativity Through Mobile P2P Computing

Adam Wierzbicki, Anwitaman Datta, Łukasz Żaczek, and Krzysztof Rzadca

1 Introduction

Among many potential applications of mobile P2P systems, collaboration applications are among the most prominent. Examples of applications such as Groove (although not intended for mobile networks), collaboration tools for disaster recovery (the WORKPAD project), and Skype's collaboration extensions, all demonstrate the potential of P2P collaborative applications. Yet, the development of such applications for mobile P2P systems is still difficult because of the lack of middleware.

The largest potential for mobile P2P applications today lies in the use of mobile phone platforms, especially of phones for the 3G (UMTS) network, or general-purpose platforms like Ultra-Mobile PCs (UMPCs). The recent increase in the capabilities of these devices has been dramatic. Today, it is possible to develop a variety of applications for mobile phones or other devices that would share resources of their users', including information that is available to the particular users only. However, in mobile P2P environments, not only is it impossible to used advanced

The work presented in this article is partially supported by A*Star SERC GRANT NO: 072 134 0055.

Adam Wierzbicki
Polish Japanese Institute of Information Technology, Warsaw, Poland,
e-mail: adamw@pjwstk.edu.pl

Anwitaman Datta
Nanyang Technological University, Nanyang Avenue, Singapore,
e-mail: anwitaman@ntu.edu.sg

Łukasz Żaczek
Polish Japanese Institute of Information Technology, Warsaw, Poland,
e-mail: lzaczek@pjwstk.edu.pl

Krzysztof Rzadca
NanyangTechnologicalUniversity, NanyangAvenue, Singapore,e-mail: krz@pjwstk.edu.pl

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_50, © Springer Science+Business Media, LLC 2010

middleware based on the Service Oriented Architecture, but it becomes difficult to reuse the most developed platforms for P2P systems, such as JXTA or the Microsoft P2P framework.

P2P applications for mobile devices can be developed today with the support of the 3G network's infrastructure. Such applications are not using a pure Peer-to-Peer computing model, as they require specialized servers of the 3G infrastructure to perform user and resource location. Yet, the sharing of resources by such applications can be done in a purely P2P manner. For 3G mobile phones, the use of SIP and the IP Multimedia Subsystem (IMS) can enable the development of such hybrid P2P applications. Petri Pöyhönen, Vice President of Nokia Networks, has chosen the following words to describe the purpose of IMS: "we have no practical mechanism to engage another application-rich terminal in a *peer-to-peer* session. (...) *there is immense value in sharing with peers*. (...) We will be sharing real-time video (see what I can see), an MP3-coded music stream, a whiteboard to present objects, and we will be exchanging real-time data." This quote indicates that like SIP, IMS is seen as a middleware to develop P2P applications. Yet, a fully distributed P2P application would be independent of the operator's IMS infrastructure, and would consequently be able to operate also in an environment where the UMTS network is not available. The challenge therefore is not just in the support of P2P services in a mobile environment, but rather to support them in a *heterogeneous mobile and fixed environment*, enabling collaborative work for nomadic teams with or without infrastructure support, preferably without significant disruption of services or decrease of service capabilities.

The developing P2PSIP standard is a candidate for supporting such P2P applications for mobile devices. Pure P2P applications that are used for collaboration can be modeled using three layers: the *application layer* which implements application-specific logic, the *collaboration middleware layer* that provides several supporting functions for the application layer and may use P2PSIP, and the *P2P overlay layer* that implements the most basic function required by the collaboration middleware layer, including reliability and security functions.

Assuming such a layered decomposition, the question remains what are the crucial services that should be included in a collaboration middleware that would support a wide range of collaborative applications? A second question could be: is the decomposition between the collaboration middleware and P2P overlay layers? This decomposition is, after all, purely logical, since all of the logic would be implemented in the application layer of the OSI protocol stack. In this chapter, we shall demonstrate that a vertical integration of the two layers could result in several advantages. These advantages are mostly due to the fact that the collaboration middleware layer can supply the P2P overlay layer with important information: the *implicit social network created by mining the metadata and statistics produced by the collaboration applications* (or by the function calls of the collaborative middleware). Using social network information in routing and network management can not only enhance the performance of routing in mobile P2P environments (particularly in Delay Tolerant Networks), but also allows to satisfy additional constraints on routing (such as trust requirements). Other P2P overlay functions, such as publish-subscribe operations, can also be supported by social network information.

The first contribution of this chapter lies in a systematic design of a middleware for P2P collaborative application in a heterogeneous, mobile/fixed environment. We show that the SIP protocol could be an important substrate in the implementation of such a middleware, and analyze which middleware functions could be supported by SIP (and which must be implemented separately). We give a comprehensive overview of the emerging standard of P2PSIP. Another contribution bases on the concept of vertical integration of the collaborative middleware and P2P overlay layers. We review related work that proposed solutions that were vertically integrated to some extent; then, we describe a new concept of SocialCircle, a P2P overlay that is much more closely integrated with the implicit social network provided by the collaboration middleware layer. We give preliminary results of experiments with a prototype of SocialCircle.

The chapter is organized as follows: the first section discusses the functionality of the collaboration middleware, with an emphasis on the most challenging performance and security problems that can require new research. This section also describes the information that can be provided by the collaboration middleware to the underlying P2P overlay layer for cross-layer optimization. The section includes an overview of related work on P2P middleware for mobile collaborative applications.

In the next section, we describe protocols and frameworks that can be used to implement the desired collaboration middleware functions: SIP and P2PSIP. The section discusses the current state-of-the-art in this area, identifying the most difficult research problems.

The last section concerns how the P2P overlay layer can use information provided by a vertically integrated collaboration middleware layer in order to optimize its operation, and how the overlay can be specially designed to better support collaborative applications. This section gives an overview of some recent relevant developments the area, and indicating the way ahead and open research issues to realize a P2P infrastructure tailor-made to support collaborative work in a mobile environment, and also using meta-information from the upper layers in order to make the P2P infrastructure more resilient and efficient.

2 Applications for Collaboration and Creativity Support

Research and development on Computer Supported Collaborative Work (CSCW) is an old area (there exist mature, commercial technologies such as LotusNotes and Microsoft Outlook) that has recently accelerated due to the use of the Internet. Wikipedia, SourceForge, CVS and SVN, iGoogle are examples of successful applications of Internet-based CSCW. Use of the P2P computing model in this area is yet in its infancy. Still, applications such as Groove, Skype and many others show high potential. In 2005, the first International Workshop on Collaborative Peer-to-Peer

Information Systems (COPS 2005) has taken place in Linkoping, Sweden. Since then, the workshop is held yearly.

Creativity support is a younger research area that has been developed mostly by psychologists and management scientists. Computer science has been applied to creativity support since the beginning of the XXI-st century. The first applications, like Oasen, Serious Creativity, Mindmanager, Maxthink, Axon, Groupsystems&Groupintelligence, focus on using knowledge management for idea generation and recording, as well as on enhancing creativity through computer-supported collaboration [17, 19, 33]. Many of these applications apply advanced visualization and human-computer interaction technology, and do not have a collaborative character, but focus on the individual creative process. On the other hand, creativity support is strongly connected to CSCW, since it requires support for debating, brainstorming, and group decision making.

A step towards the development of Peer-to-peer groupware has been the technology of Groove Networks [1]. It concentrates on providing support for collaboration of a team on a project. Groove offers workspaces for aggregation of documents, messages, and application-specific data, that are synchronized among all peers in a workspace; Groove also offers chat sessions, notepads, a calendar, and a file archive. Groove also has a comprehensive security model. Yet, Groove does not offer knowledge management, semantic social networks and functions for creativity support; also, while it uses the P2P computing model, it has not been designed for a mobile environment. Groove users must have access to an infrastructure of "relay servers", which means that the system is not well suited to nomadic use.

There has not been much research on collaboration and creativity support on mobile devices. Shark [32] supports management, synchronization and exchange of knowledge in mobile user groups. Shark uses a semantic meta-data hierarchy for routing of messages in its P2P overlay network. However, Shark does not posses features that support creative processes; Shark is also not suitable for teamwork. The reason why Shark does not have these capabilities is that it has been designed for a low-bandwidth mobile ad-hoc network (Bluetooth).

Two interesting examples of creativity support software that could be ported to the P2P model and, in the process, extended with collaboration capabilities, are Freemind and JabRef. Freemind is an open-source application that enables visual mindmapping. JabRef enables the management of bibliographic references using the BibTex standard, as well as searches and automatic acquisition from popular Web-based bibliographic databases such as CiteSeer or DBLP. Both FreeMind and JabRef are really useful for support of the individual creative process; the applications are opensource (written in Java), and could be used on mobile devices (because of low resource requirements). Yet, both applications would benefit from an extension that would enable collaborative work. FreeMind could enable co-authoring of mindmaps, and JabRef could support sharing, searching and recommendation of references. Both of these functions could be simply supported through a central server (like the popular service of Bibsonomy), but supporting them in a heterogeneous, fixed/mobile environment for nomadic teams of users requires the use of the P2P computing model. The development of both extensions is underway in the mTeam project [3].

We believe that the development of applications for collaboration and creativity support on mobile P2P systems is still difficult because of the lack of middleware. While recent advances in technology of mobile devices and wireless networks enable the use of such technology for collaboration and creativity support, little is known about supporting such applications without centralized servers. For this reason, development of P2P middleware capable of working on mobile devices is a crucial technological and research challenge.

3 Collaboration Middleware Functionality

3.1 General-Purpose Mobile P2P Middleware

Several research and development projects have proposed general-purpose middleware for developing arbitrary applications on mobile devices using the P2P model. Most of these efforts aim to provide a common framework for the rapid development of applications.

Mobile Chedar [5] is an extension to the Chedar framework that provides an API to application developers. Mobile Chedar uses Bluetooth and J2ME, and uses a superpeer model. A special gateway node (a PC with fixed Internet connectivity) manages the other Mobile Chedar nodes and provides Internet access.

The oldest and best-known open-source P2P middleware is JXTA [25]. JXME [23] is a lightweight implementation of JXTA for mobile devices using the J2ME platform. JXME is aimed at devices that are too resource-constrained in order to support the full JXTA protocol; rather, JXME relies on a relay proxy to provide full JXTA functionality.

JMobiPeer [7] is a middleware for J2ME that offers interoperability with JXTA. It provides support for discovery and group management.

Proem [24] aims to support collaborative applications, but is in fact a general-purpose P2P middleware for mobile devices in an ad-hoc network. Proem is implemented in Java Standard Edition, and can therefore be run only on powerful PDAs or UMPCs.

MOBY [20] enables P2P exchange of services and data, dynamic service location and client mapping to achieve higher performance and reliability. MOBY relies on JINI (which is not compatible with J2ME).

While all of the general-purpose middleware discussed above can be used to support collaborative applications, these proposals *do not address the specific problems of collaboration and creativity support*. Another drawback of the proposed generic approaches is that most of them *are tightly integrated with a specific P2P overlay network*. This makes it impossible to port the generic middleware to various types of overlay networks. Only JXTA supports, to some extent, different overlay variants, although it mostly relies on a hybrid superpeer architecture.

In this part of the chapter, we shall describe functional requirements for a middleware that would be dedicated to the support of collaborative work. We shall assume that these functions are independent of the specific P2P overlay network, although their operation could influence how the overlay works through vertical integration. Later, we shall discuss how elements of this middleware could be implemented.

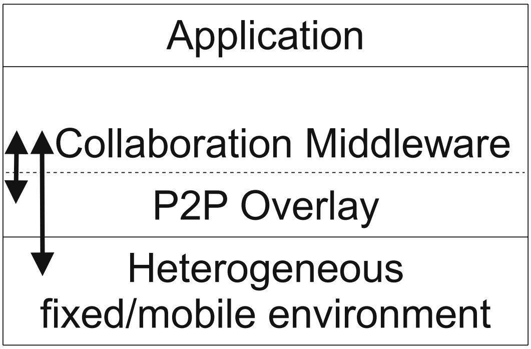


Fig. 1 Vertically integrated collaboration middleware

3.2 Overview of Middleware Functions

The purpose of developing middleware is the support of application development. In the process of middleware design, it is therefore necessary to identify and describe functionality that should be useful for supported applications – in our case, collaborative applications. We have based the functions presented below on a review of the functionality of collaborative applications. It is usually hard to become convinced that the chosen middleware functions are indeed comprehensive, until the middleware has been used to create several applications. Therefore, a larger set of functions must be chosen that can support a variety of applications. However, at the design stage it is possible to reduce certain middleware functions to others, and to choose the most generic versions of required functionality.

An important problem is that the chosen functions of a middleware should satisfy several requirements of efficiency or security. For the purpose of vertically integrated middleware, it also becomes significant what information can be obtained from middleware function execution for the purpose of optimizing lower layers and how this information can be stored, updated, and communicated.

An abstract design of the collaborative middleware is presented on Fig. 1. The interface between the P2P Overlay and the Collaboration Middleware layers should be generic and will be discussed in the next sections. Functions of the Collaboration Middleware include:

3.2.1 Session Management

This function enables the initiating, modifying, and ending of sessions of multiple users. Session management requires discovery of the participants, and therefore the use of the P2P overlay layer routing function. A session describes the communicated media, including information about the encoding and transport method, between all participants of the session (each participant can receive a different set of media streams). Therefore, session establishment and modification requires signaling.

3.2.2 Conference Management

This function enables the establishing and updating of conference state. The state can include the conference membership. Also, the conference state can include floor control (allowing only one user to speak at a time), or another form of control (for example, moderation). Conferences can have several modes of establishment: for example, dial-out mode, where a single user invites all other conference participants, or join-in mode, were new users join the conference whenever they want.

The conference management function requires use of the event management function (an event is that a user has joined a conference). The conference itself uses the P2P overlay layer multicasting function for the delivery of conference media.

The session and conference management functions can be the source of usage statistics that can determine the level of acquaintance between persons to be used in a *social network*. Each user could maintain these statistics about his own communication, enabling the creation of an *egocentric social network*; on request, users could share their egocentric networks with other, authorized or trusted users. In this way, a global view of the social network could be constructed by trusted superpeers.

3.2.3 Data Synchronization

Data used by arbitrary other services, or by the application layer, will need to be synchronized among a group of authorized devices (that can belong to a single or multiple users). This synchronization can have several models – from a strong synchronization (all changes are propagated to all devices, WYSIWIS – What You See Is What I See), to a relaxed synchronization that allows different versions of the data to exist at the same time on different devices (concurrent work). Two special cases of data synchronization are: document synchronization and workspace synchronization.

3.2.4 Data Sharing

Unlike in data synchronization were the data is intended to produce a single version from all participants (in relaxed synchronization, data could have different versions, but the intention is to produce a single version in the end), some forms of data should be shared only with some (authorized and interested) users and are not intended to produce a single version. Furthermore, the data itself could be controlled by an authorization policy that governs their distribution. Using this function, a user of a collaborative application can make certain information available to other, interested users who can copy and distribute this information without needing to synchronize it with the original version.

3.2.5 Presence

State and profiles of users in the system should be made available to other, authorized users. The user (or device) profile can include available services or resources. Overall, the presence service would be responsible for the maintenance of all available context information about the user, including location information (if available). For example, consider a context that describes what kind of device is currently in use by the participant; this kind of information would be part of the service profile maintained by the presence service, and could be obtained automatically. The presence service requires the use of the overlay layer publish-subscribe service.

Presence information can be the source of important statistics. Users who have a high availability (are nearly always present) are good candidates for superpeers, and can be considered more reliable for various other functions (network management, storage). The availability statistics can be maintained by the peer who stores presence information about another peer in the overlay and communicated to trusted superpeers on demand.

3.2.6 Workspace Awareness

When users form a group that works in a shared workspace (or on one document), other users must be aware (to the extent possible) of what the others are doing, in order to enable coordination and planning of work (for example, who is currently editing a certain document). This function is a specialized form of data synchronization.

3.2.7 Group Management

Group management is the function of creating, modifying, deleting of groups of users, including the possibility of establishing hierarchical groups. The group can be working in a shared workspace. Groups can be of many kinds: they can be open (anyone can join), closed (only the group owner can join new members).

3.2.8 Access Control

This function is crucial for the enforcement of authorization policies in a collaborative environment that allows users to share information. The policies can be intended to protect sensitive information, a user's privacy, or business policies. An access control method for the collaborative application would have to support not only the first access to shared information, but all subsequent sharing of the provided information. Access control can be based on various models, but all of them require the use of an authentication service that should be provided by the P2P overlay layer.

3.2.9 Trust Management

The function of trust management is required in open environments, were the users must depend on the actions of others and are uncertain about the outcome of such interactions. The purpose of trust management is decision support in such circumstances. For example, consider dynamic teams that include participants who do not know each other. In this case, participants can rely on information from trust management for making decisions that require dependence on unknown team members. In return, after gaining knowledge about the new team members, participants will give feedback to the trust management system that can be used to update trust or distrust. Note that trust or distrust need to be maintained in a context that can, in the simplest form, be represented by a value out of a finite set of contexts.

3.2.10 Event Management

This service allows the subscription to events, notification of subscribed users about events, and publishing of events. It uses the P2P overlay layer publish-subscribe service. In particular, the event management service should also include an ordering of events (especially important for synchronization).

3.2.11 End-to-End Security Functions

While overlays may provide some hop-by-hop security functions, these may not be sufficient for the purposes of the collaborative application. For that reason, end-to-end confidentiality should be provided by the middleware. In addition, authentication between the communicating peers (session participants) is required.

3.3 Required P2P Overlay Functions

3.3.1 Routing

Routing of information to a destination (a key that can be a user or service identifier).

3.3.2 Publish-Subscribe

This service allows the subscription to an arbitrary topic, notification of subscribed users about changes to the topic, and publishing changes to a topic.

3.3.3 Multicast

A special case of the publish-subscribe service where a user subscribes to a group. Notifications are any messages from the group, and publishing changes means sending a message.

3.3.4 Overlay Adaptation Functions

These functions enable vertical integration by receiving data from the collaboration middleware layer and adapting overlay structure using the received information. Mostly, this means adapting the local view of the social network that is used to support the P2P overlay network.

3.4 Research Issues Related to Middleware Function Design

The proposed middleware functions should satisfy strong requirements of performance, scalability and security. Furthermore, in a mobile P2P environment their design will have to differ from designs used in fixed P2P networks. These considerations point to the conclusion that new research issues will need to be solved in the middleware design and implementation. It is not the purpose of this chapter to present solutions of the described problems, but merely to identify the most difficult and important issues for future research.

The *session management* service, and therefore the routing service, should have a low call setup latency. This is especially difficult in a mobile P2P environment. Therefore, this requirement poses new research challenges; a special kind of routing may be required for session setup messages. Another requirement is the reliability of session management; how often would a session establishment fail because routing failures due to churn?

The *event management* and *presence service*, and consequently the overlay publish-subscribe service will be required to optimize the load on the notifier, network traffic for notification, and delay of notification delivery. In addition, the requirement of delay tolerance or reliability can be imposed on these services; for example, if a peer departs from the network, it may need to receive all missed notifications after it joins the network again.

In a mobile P2P environment, the *data synchronization* service would have to operate in the concurrent work mode. A question arises how this service should implement a distributed transaction that would produce a single version out of the existing, concurrent versions. This may be achieved through a pre-commit stage (a “code freeze”), but the duration of this pre-commit stage during high churn and long peer disconnections may need to be high. Also, the data synchronization function may need to implement work scheduling decisions that have been made by the team, allowing only certain changes to be made by particular users.

The *data sharing* function bases on the information delivery mechanism of the P2P network. The multicast or publish-subscribe service may be used for that purpose. These services would have to satisfy efficiency and reliability requirements. Also, the associated function of *access control* would have ensure that the data is sharing in agreement with a specified policy. The fundamental problem here is that when a user shares data with another, authorized user, the receiving user could share the data with unauthorized users. How to control this second (and further) degree of sharing is a difficult research problem.

The *trust management* service will need to use some expression of trust context, which may be based on a semantic description of the problem that the collaboration is trying to solve. Also, the kind of feedback information used by the trust management system, and the algorithms used to calculate trust or distrust measures are the subject of ongoing research. For example, the feedback information could be based on edit history or on an explicit evaluation of team partners.

4 SIP and P2PSIP

The Session Initiation Protocol (SIP) and its extensions already implement a number of the functions required by the collaboration middleware (although not all). Moreover, SIP is implemented and available to programmers on most modern mobile phones, and SIP is specially treated by many firewalls and NATs. Since many P2P protocols today are implemented on top of an application layer protocol (such as HTTP), these observations can lead to the conclusion that the collaboration middleware for mobile P2P environments should be implemented using SIP. The question remains how, and to what extent, SIP can be used to implement all required middleware services.

In this section, SIP is briefly introduced (with a focus on its relevant services). In the next part of the section we discuss how the dependence on server infrastructure can be removed through the use of P2PSIP. Research and standardization of this extension is continually advancing. Yet, for the purpose of a collaboration middleware, all required middleware services would have to be implemented in a distributed, P2P environment. This requires a new design effort, and raises issues of security and performance.

4.1 The Session Initiation Protocol

The Session Initiation Protocol (SIP) [30] is an application-layer control (signaling) protocol for creating, modifying and terminating sessions with one or more participants. SIP sessions can be used for any kind of application data, and SIP does not standardize or attempt to control session contents. Thus, the first required service of the collaboration middleware that can be implemented using SIP is the Session Management service.

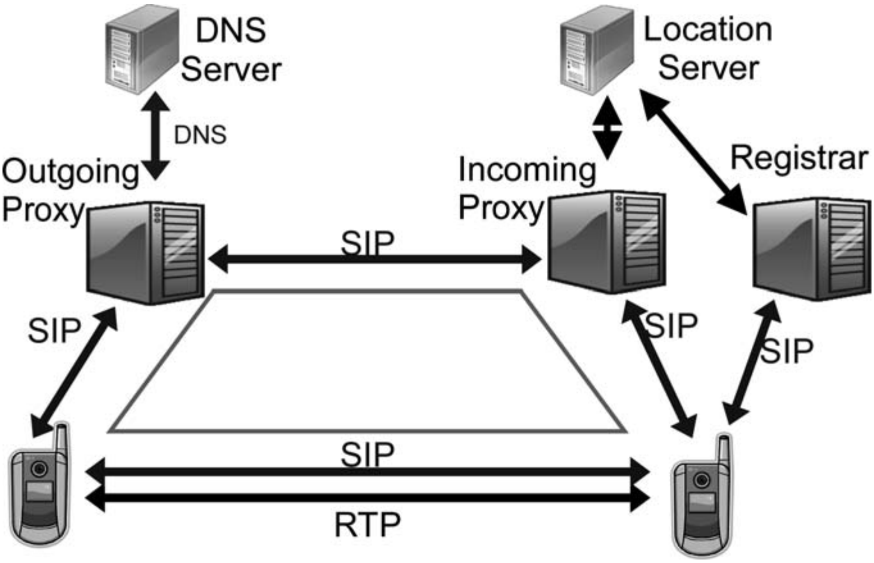


Fig. 2 SIP signaling using proxy and registrar servers

SIP introduces its own addressing of users (or services), with the intension of achieving independence of the user's location in the network (IP address). Moreover, SIP session management services allow to search for users at several locations during session initiation, and to specify redirection policies that can depend on the user's current state and will redirect a session to another location.

SIP can use either TCP or UDP for message transport; due to the possibility of using UDP, acknowledgement messages are required and a three-way handshake is used during session initiation. SIP uses an infrastructure of servers for the resolution of its addresses and for the control of the session. SIP servers can be of four kinds (the proposed distinctions are logical, and server roles can be combined): registrar servers, proxy server, redirect servers and location servers. Another type of SIP server is a gateway proxy server. Proxy servers can be thought of as analogous to local DNS servers that take over responsibility for SIP address resolution. Proxies can also work as gateways to other networks or domains, and SIP signaling can be forwarded by arbitrary chains of proxies. Proxy servers can also implement a variety of services, such as lists of missed calls, call forwarding or screening. Proxy servers may also implement call admission control and report usage. For the purpose of some service, proxy servers may operate in a stateful model and maintain state information for the duration of session initiation (not during the session).

The registrar server is analogous to an authoritative DNS server that contains the definite information required for SIP address resolution. A SIP user must register with a registrar server in order to be able to receive sessions.

Figure 2 shows the SIP signaling flow between two user agents, two proxy servers and a registrar server. Note that the user agents exchange signaling messages only

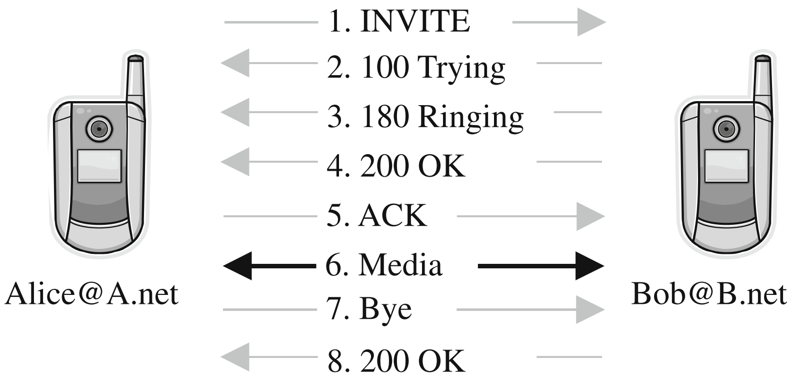


Fig. 3 SIP message flow for simple session

Table 1 SIP methods

Basic methods	Extended methods
INVITE	INFO
ACK	NOTIFY
BYE	PRACK
REGISTER	PUBLISH
OPTIONS	SUBSCRIBE
	UPDATE
	MESSAGE

with their local proxies. Note also that when the roles of a proxy server, registrar server and location server are distinct, the proxy server does not signal the registrar server directly using SIP; rather, both of them interact with the location server using a non-standardized protocol. The signaling flow is simplified when proxy, registrar and location servers are integrated.

The SIP protocol is a text-based protocol that resembles HTTP. It can use several methods, which form two sets. Table 1 summarizes the basic and extended methods of SIP. SIP messages use mandatory headers that resemble mail messages, and can include a MIME body and Session Description Protocol (SDP) information [18] (Fig. 4).

Figure 3 shows a basic SIP message flow during a simple session initiation. Note that the response codes of message exchanges resemble HTTP. Figure 2 shows the signaling message flow during a SIP registration procedure.

For the implementation of the Session Management service of the collaborative middleware, SIP provides comprehensive negotiation functions that allow users to agree on received media formats and transport, as well as updating session state by adding or removing participants, or changing their session parameters. Such

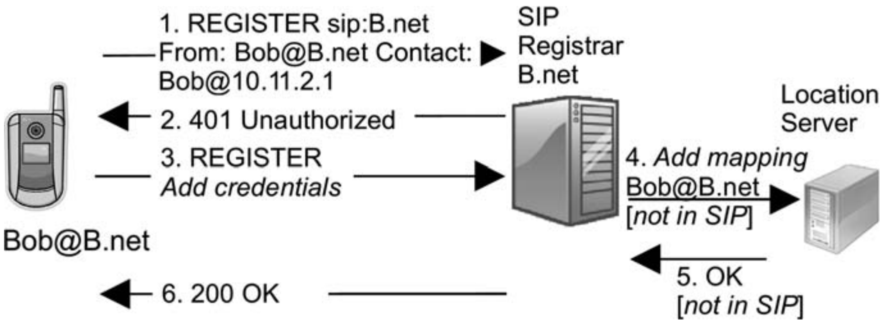


Fig. 4 SIP registration

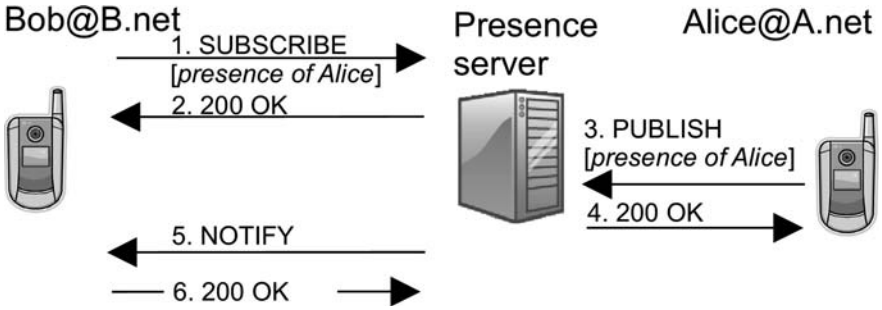


Fig. 5 SIP presence service

functionality is essential for session management and using SIP seems the easiest way to implement a comprehensive Session Management service.

SIP has been extended with several functions that allow to implement various services of the collaboration middleware. To a limited extent, the Conference Control service can be implemented using SIP; however, SIP does not have features for floor control in a conference. The Data Sharing service can also be partially implemented, since SIP has an extension for file sharing (SIPPING) [15]. SIP has further extensions that support challenge-response authentication based on a shared secret (although a SIP proxy can also implement RADIUS authentication).

SIP provides the most comprehensive support for the Presence and Event Notification services [27]. Figure 5 shows the SIP message flow that implements a presence services. Note that the implementation is based on a server and uses three new SIP methods: SUBSCRIBE that allows an agent to indicate that it is interested in receiving information about an event (in this case, the presence of another user), PUBLISH that is used to inform the server about new information, and NOTIFY that is used by the server to notify a client. The figure is simplified because it does not show a notification message that automatically follows the response to the SUBSCRIBE message.

Presence in SIP can be extended to include information about changes in buddy lists, by adding authorization policies that control who can receive presence information (using the XCAP standard) [28]. Another extension concerns presence data; using the Presence Information Data Format (PIDF) or the Rich Presence Extensions to PIDF (RPID) [31, 36] it is possible to express detailed status of users, using information that could be automatically extracted from calendars. PIDF and RPID are XML dialects. The SIP extensions for presence and instant messaging [11] are developed by the SIMPLE IETF working group [29].

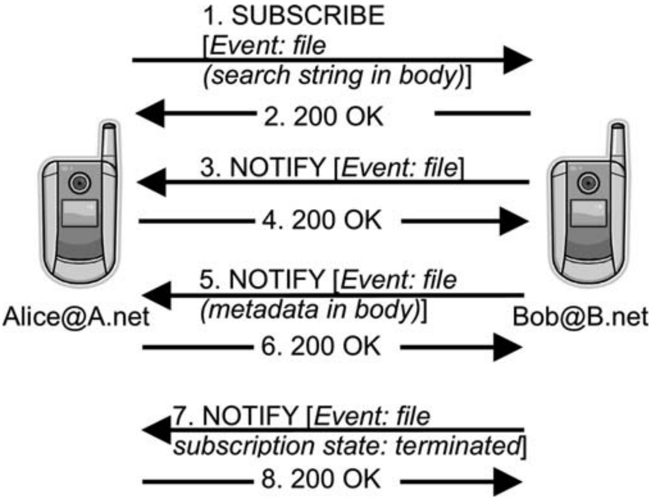


Fig. 6 The SIP event notification

Presence is a special case of SIP's event notification framework that is shown on Fig. 6. SIP allows the subscription to arbitrary events, and the notification about these events will be sent to all subscribed users. This design of the event notification service can create severe performance and scalability problems.

4.2 P2PSIP

SIP can be considered today as a stable and mature technology that has seen wide adoption in Internet-based application. Additionally, the choice of SIP as a signaling protocol for IMS and its consequent implementation by 3G mobile phones has further increased the importance of SIP as a signaling protocol. In the previous sections, we have shown how SIP can be used to implement several services of the collaboration middleware. Additionally, IMS is a powerful middleware that facilitates easier development of many types of applications, although it is not specially designed for collaboration services. On the other hand, applications developed using IMS will depend on the operator's 3G network. In an environment that does not have access to the 3G network, IMS applications cannot function.

The developing P2PSIP standard aims to enable the use of SIP without the dependence on a server infrastructure. The first step in this effort is to distribute the SIP registrar servers using a P2P overlay. Following this, other SIP functions such as publish-subscribe for event management and presence can also be distributed, aiming to create a P2P framework that would support full SIP functionality (a kind of P2P-IMS). The full set of requirements for a P2PSIP system would include [34]:

1. **Zero configuration**. The system should be able to automatically configure itself, e.g., detecting NAT and firewall settings, discovering neighboring peers and performing initial registration.
2. **Heterogeneous nodes**: It should be able to adapt to available resources and distinguish between peers with different capacity and availability constraints. This favors the distinction between nodes and super-nodes as in Kazaa.
3. **Efficient lookup**: The system should be able to execute a lookup of a SIP identifier efficiently, which can be achieved using a structured P2P network.
4. **Advanced services**: The system should support advanced telephony services such as offline voice messaging, multi-party conferencing, call transfer and call forwarding as well as advanced Internet services such as presence and instant messaging. For the support of collaboration, the system should support all collaboration middleware functions described in the first section.
5. **Interoperability**: It should easily integrate with existing protocols and IP telephony infrastructure.

The conceptual model of a P2PSIP system [8] consists of *peers that are coupled with SIP agents* such as user agents, proxies, redirect or gateway servers. In other words, it is possible to think of peers as executing a separate functionality but being coupled with ordinary SIP entities. All peers should be able to participate in the P2P overlay network using the *P2PSIP peer protocol*. However, there may exist SIP user agents that wish to use the P2PSIP system, but are not coupled with a peer capable of participating in the overlay. These peers can use the *P2PSIP client protocol* to contact proxy peers that participate in the overlay. Finally, it is also possible for a SIP agent that does not understand the P2PSIP protocols to communicate with the P2PSIP system directly using SIP.

A high-level overview of the steps that a peer has to carry out in a P2PSIP system is as follows:

1. In order to join an existing overlay, the peer needs to locate a P2PSIP peer that already participates in the overlay, which is referred to as a bootstrap node. IP addresses of bootstrap nodes can be obtained in many ways: from a cached list, using multicast discovery, using manual configuration or public bootstrap nodes.
2. After bootstrapping, the peer should authenticate itself, traverse NAT and firewalls, and register itself in the overlay (or register as a P2PSIP client to some P2PSIP proxy peer). Authentication methods or NAT traversal are not defined yet.
3. After joining, the peer uses the P2PSIP peer protocol to search for other peers or resources, share own resources or insert new resources in the system. Resources can be SIP identifier-IP address mappings, context information, meta-

data, or files. There are several candidates for a P2PSIP peer protocol, all of them are structured P2P overlay networks.

4. After P2PSIP peers discover other peers, they can initiate SIP sessions or send other SIP messages independently of the P2PSIP peer protocol, but possibly using resources of the P2PSIP system such as proxy or gateway peers (also needed for NAT traversal).

At a first glance, the task of distributing the functions of SIP registrar servers that implement a simple directory function seems easy. Structured P2P overlays are capable of providing directory functions. Yet, there is to date no standard of a P2P overlay. Also, P2P traffic often struggles to pass NATs and firewalls, often hiding using another application layer protocol. These considerations raise the first question concerning the design of P2PSIP. Should P2PSIP use a P2P overlay as an external component (requiring SIP clients to implement another protocol and raising issues of choosing the best overlay and of firewall traversal), or can a P2P overlay be designed to form part of SIP? In other words, can a P2P overlay be created and maintained using the SIP protocol itself?

For these reasons, one of the first proposals of a P2PSIP system advocated the use of the SIP protocol as a signaling protocol for overlay setup and management [8, 9]. This proposal specified the mechanism for user registration and location in a P2PSIP overlay. The overlay was created using a modified SIP REGISTER message that could be used to join the overlay and to update overlay neighbors. The proposed approach was implemented using Chord as the P2PSIP overlay. Special SIP identifiers were used to identify peer nodes (a SIP identifier included a Chord node identifier and could specify whether the nodes' IP address was known).

4.2.1 P2PP

The other approach to developing a P2PSIP system requires the use of an additional protocol for overlay management. A current proposal of the P2PSIP working group advocates the development of a new standard for a general-purpose protocol for P2P overlay network management that would also offer the basic services required by P2PSIP [6]. This protocol, called simply the P2P Protocol (P2PP), is independent of the overlay routing and allows the construction of arbitrary overlays (structured or unstructured).

Figure 7 shows the reference architecture of P2PP [6]. The protocol provides an API to the SIP user agent or other applications. The API consists of three separate interfaces: the *service interface* provides functions to join or leave the overlay, functions for routing table maintenance and updates. The *data interface* provides functions to insert, lookup or delete an object by its key. The *diagnostic interface* provides functions for gathering statistics, like response time or relay bandwidth of a peer.

The P2PP protocol does not specify how the routing and network management of the overlay works. In fact, it is possible for the overlay to be structured or unstructured, and for the structured overlay to use any routing scheme. However, this

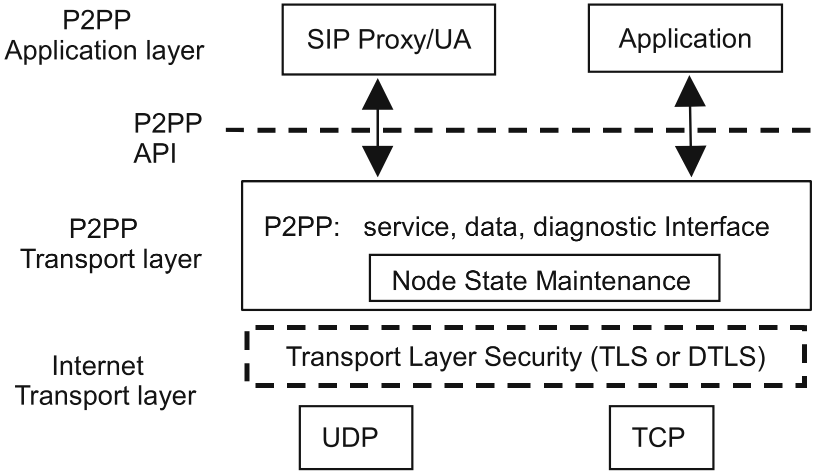


Fig. 7 P2PP reference architecture

does not imply that P2PP specifies just the interface part of the overlay. The protocol specifies how peers maintain state. The P2PP node state contains the following information:

- The P2P overlay routing algorithm (can be any)
- Overlay ID
- Hash algorithm (if any)
- Routing type (iterative or recursive)
- Routing table (table of other peers in overlay)
- Neighbor table (table of immediate neighbors)
- Both tables include for any peer: Peer ID, IP address, RTT, Uptime
- Number of neighbors (sum of routing and neighbor table sizes)
- Resource table (objects the peer is responsible for)
- Replicated resource table (replicated objects)

Overlay routing can be iterative or recursive, which needs to be decided at the time of overlay creation. Peers can issue *parallel iterative routing requests*; then, one copy of the request needs to be designated as primary. Requests use TTLs. The protocol also supports request transactions, which can add reliability to requests if UDP is used for transport. Reliability is provided as a hop-by-hop mechanism. An interesting feature of P2PP is that it can combine the use of TCP and UDP within one overlay. With recursive routing, TCP and UDP can be combined using hop-by-hop reliability for UDP.

P2PP supports the use of security using hop-by-hop TLS or DTLS. An authentication server can be used that issues certificates to users. The P2PP specification describes adversary models and the protocol is protected against some routing and storage threats, as well as replay attacks.

P2PP is a candidate for the generic interface between the Collaboration Middleware and P2P Overlay layers, described on Fig. 1. The reason for this choice is that P2PP is by itself generic, allowing the implementation of arbitrary overlays while keeping the same protocol and service interfaces. However, P2PP does not support all of the functions of the P2P Overlay layer, described in the previous section. In particular, P2PP does not support publish-subscribe or multicast. Of course, the P2PP interfaces do not include overlay adaptation functions. Research on an adaptation of P2PP to support all of the required functions is continuing in the mTeam project [3]. The P2PP interface has already been extended to support publish-subscribe and multicast – a proposal of revision of the standard is forthcoming.

4.3 A Case Study: SharedMind

SharedMind is a collaborative version of FreeMind [2], the most popular open-source mind mapping software. Our motivation is to enable the group to work on a shared mind map. Each member of the group can interactively add, delete or modify nodes in the mind map. These modifications are seen almost in real time by the whole group. If the network connection between groups' members is broken, the members can continue to work independently (or in subgroups, that have connectivity). After the connection is restored, the files are merged. SharedMind is an ongoing work that prototypes and tests the design assumptions of the collaborative middleware.

The original FreeMind follows a standard, model-view-controller architecture. The mind map is represented internally as a tree. Nodes of the tree correspond to the nodes of the mind-map. The text of a node and its style information are stored as properties of the node. The persistent version of the tree is stored as an xml file. User's modifications (such as adding a node, editing node's description or its attributes) are represented as "commands". These commands are executed on the model by the controller.

In our implementation, each member of the group has a local copy of the shared data (the mind map edited by the group). The main issue in SharedMind is to provide seamless *data synchronization* of changes made to local copies. Other functions defined in Section 3.2, especially those related to the security and group management, will be implemented by the middleware and do not require substantial modifications to the FreeMind code. We use P2PP as a generic interface to a P2P overlay. All the members subscribe to a common publish-subscribe topic. Whenever one of the members modify her local copy, a message containing the description of the modification (an xml string containing the marshalled action) is published on (sent to) this shared topic. The middleware propagates the modifications, by the standard subscribe mechanism, to the other members of the group. A member, after having received such a remote modification, unmarshalls it, checks for possible conflicts and, in case there are no, applies it on her local copy.

In this collaborative editing scenario, two main types of conflicts may occur. Firstly, a remote modification may concern a node that is currently edited by the local user. Secondly, modifications may be concurrent. In this case, local editing has finished and the notification message is sent, but the message has not reached the author of the remote modification, before she sent her message. Both types of conflicts are easy to detect. The first one by a simple state variable that stores the currently edited node. The second one, by using logical vector clocks for messages. However, both conflicts have to be resolved by the user, on the level of the UI.

We also plan to support delay-tolerant editing, in which group members continue to work on shared data, even if they do not have any network connection. Delay-tolerant editing relies on presence (provided by the middleware), versioning and checkpointing (specific to FreeMind's data structures, thus implemented over FreeMind's code). The group chooses a random number (using a bully selection algorithm) that becomes the current version number of the data. The presence function of the collaboration middleware periodically checks whether all the group's members are reachable. Group composition changes when a member disconnects, or when a persistent network failure divides the group into subgroups. In such event, each member locally reverts to the last common version of the shared data (at the moment of the last successful presence protocol invocation), and checkpoints this version with the previously elected version number. Afterwards, a member re-applies the modifications that followed and continues to work with the reachable part of the group (electing the new version number). After the connection is restored, the re-united group should produce a common version of the data from the modifications done in each subgroup. The presence mechanism notifies group's members. Then, each subgroup elects a member responsible for merging the data into a one, common version (a *merger*). Mergers exchange lists of version numbers. Then, each merger determines the last common version number (the last number that appears on both lists). Each merger produces a list of modifications its subgroup made over the common version, using the previously checkpointed revision. These lists of modifications are then send to the other merger. Next, mergers merge all the non-conficting changes and resolve conflicts (on the level of the UI). Then, an elected merger applies changes that were made by other, non-merger group's members during the process of merging. This process may also involve possible conflict resolution. Finally, the elected merger's data is considered as the common group's data.

To conclude, the complete version of SharedMind will use most of the middleware functions (Section 3.2). Security-related functions are mainly independent of the original code, and thus do not require substantial modifications. Data synchronization turns out to be most complex to implement, as the data structure is tightly coupled with the original code. Currently, data is synchronized on the application layer (and not by the middleware), using functions provided by the collaboration middleware and the P2PP protocol, such as presence, publish-subscribe and event management. In future, we plan to provide generic synchronizing data structures at the middleware layer, such as a tree or a table (a collection of records). These data structures will be extended by application developers to host application-specific data.

4.4 Research Issues Related to SIP and P2PSIP

SIP and IMS are both client-server systems that have performance and scalability problems. In particular, the presence and event management services of SIP are prone to severe performance problems when they have many users. Another problem is the intolerance of the service to network connection loss, and the general problem of coping with event loss. In the publish-subscribe framework, event delivery is not guaranteed in the case when the intended receiver is not available, and there is no mechanism to obtain the lost events.

The P2PSIP protocol is currently in the early design stage and there is no agreement on how it would operate. However, most of the challenging research problems described in the section on collaboration middleware that concern the session management, presence and event management service would also apply to P2PSIP, since this protocol should support the full functionality of SIP over a P2P overlay network. Another challenging problem would be the transparent integration of P2PSIP with Internet-based SIP infrastructures or IMS.

The performance of P2PSIP will depend on the operation of the underlying overlay. The overlay will also support all other functions of the collaboration middleware. Therefore, a good overlay design is crucial for the support of collaboration functions. Here, the opportunity for using information from the collaboration middleware layer to optimize overlay operation becomes important, as will be shown in the next section.

5 Vertically Integrated Overlay

Structured overlays are touted to play the role of an application layer internet, potentially bridging the gap among heterogeneous networks by providing a standard API based interface for applications, and managing the underlying operations such as routing, transparently. However, in mobile (either logical or physical) environments structured overlays are harder to realize and maintain. A simple approach – called the *layered design* – is to assume that the networking layer takes care of any and all intricacies arising from mobility, and thus just deploy an overlay on top, treating the underlying layer as a black box. Such a layered design risks losing out on cross-layer optimization opportunities, which affect performance, besides potentially jeopardizing the feasibility of such a system to start with. Experiments in mobile ad-hoc environments do confirm this intuition, demonstrating that a cross-layer approach performs better, where the underlay's routing mechanisms are integrated with the overlay's routing mechanisms. A good summary comparing layered and integrated overlay designs can be found in [21]. Recent efforts have pushed this idea even further, using overlay like approach to infact perform the underlay's routing itself [10], simultaneously realizing a DHT by default.

In this part of the chapter we will first review some recent *standalone* ideas, and then propose a speculative mechanism of how these apparently different ideas together potentially can help realizing a structured overlay in a highly dynamic environment as is witnessed in mobile environments. The standalone ideas have been tested in isolation. While the proposed speculative mechanism is one of the best bets given current understanding of overlay networks, it nevertheless currently lies in the realms of future work.

We next provide a summarized survey of these stand alone mechanisms, followed by our vision of how several of these can be combined together to provide the basic routing and indexing infrastructure for the upper layer middleware. We also will point out how information from the upper layer can be exploited to make the underlying overlay more robust and reliable.

5.1 In the Realm of Ringless Routing

Structured overlays, e.g., Distributed Hash Tables (DHTs) provide essential indexing and resource discovering in distributed information systems. Typically, structured overlays are based on enhanced rings, meshes, hypercubes, etc., leveraging on the topological properties of such geometric structures. The ring topology is arguably the simplest and most popular structure used in various overlays. In a ring based overlay network (like Chord [35]), nodes are assigned to distinct points over a circular key-space, and the ring invariant is said to be hold if each node correctly knows the currently online node which succeeds it (and the one which precedes it) in the ring. The ring is both a blessing and a curse. On the one hand, an intact ring is sufficient to guarantee correct routing. Hence, historically, all existing structured overlays have de facto considered it necessary.

Traditionally DHTs implicitly assume that any pair of nodes can interact directly with each other. Under such an assumption, the ring is relatively straightforward to realize. However, in some networking environments, particularly mobile, wireless and ad-hoc, assuming that a network layer routing mechanism provides such pairwise connectivity and then realizing the structured overlay on top of the network is at least an inefficient approach since it ignores cross-layer optimization opportunities. Moreover, even on the internet infrastructure and over even a moderately small and dedicated infrastructure like PlanetLab researchers have observed non-transitive connections [14], which violates the all-pair connectivity assumption. To make things worse, large scale networks are prone to failure of individual participants, and in peer-to-peer networks, peers often go offline and come back online, leading to churn in the network. Each and all of these factors make it harder to maintain the ring invariant, becoming a performance bottleneck in the best case, and jeopardizing the functionality altogether in the worst. There are of-course many self-stabilization mechanisms which try to ensure ring integrity, with varying degree of success, but these do not fundamentally change the fact that traditional DHT routing strategies require the ring integrity as a necessary condition, even while actually it is just a sufficient condition. Next, we describe some recent works which try a different approach, investigating whether the routing strategy can be changed so that the DHT works even when the ring invariant *is not* or *can not* be met. In some sense, these approaches complement self-stabilization mechanisms, reducing the DHTs reliance on the ring-invariant on the one end, even while of-course the performance of the system improves if and whenever the ring invariant is actually met.

5.2 Virtual Ring Routing

Virtual ring routing (VRR)[10] is a DHT style "overlay" layer approach used to define the underlying network's routing mechanism. It is implemented directly on top of the link layer and provides both traditional point-to-point network routing and DHT routing to the node responsible for a hash table key, without either flooding the network or using location dependent addresses. While traditional DHTs take for granted point-to-point communication between any pair of participating nodes, VRR extends the idea, using only link layer connectivity. Essentially this means that the VRR scheme relaxes the traditional DHT assumption of a completely connected underlying graph.

Each node in VRR has an unique address and location independent fixed identifier, organized in a virtual ring, emulating Chord style network. Each node also keeps a list of r/2 closest clockwise and counter-clockwise neighbors for the node on the virtual ring. Such a set of neighbours is called the node's virtual neighbour set (*vset*). Typically, members in a node's *vset* won't be directly accessible to it through the link layer. Thus each node also maintains a second set called the physical neighbour set (*pset*), comprising nodes physically reachable to it through the link layer. Quality of the link is taken into consideration in choosing members of *pset*, and is updated regularly with changing conditions.

VRR sets up and proactively maintains routing paths called *vset paths* to each member in its *vset*, and these paths are typically multi-hop using *pset* nodes. This provides a routing table at each node, comprising of routes to its own *vset* nodes, as well as *vset paths* that pass through it. This is enough to support point-to-point communication between any two nodes in the system, since one can follow the *vset* neighbors to reach the destination along the ring. In practice, VRR performs significantly better, because a node can exploit the routing entries from other *vset paths* that pass through it. Figure 8 shows an example virtual ring, and illustrates how VRR works.

While proposed originally for network routing, where *pset* is determined by link layer connectivity, the idea of a VRR is potentially applicable even for other kinds of connections, including of-course hybrid physical networks, but also abstract ones like social network graphs. For routing and indexing services for a collaboration middleware, using such a social network to determine *pset* is particularly alluring, given that social network links can inherently provide security and robustness, as will be briefly explained later when we describe how ideas from *VRR*, *Fuzzynet* and

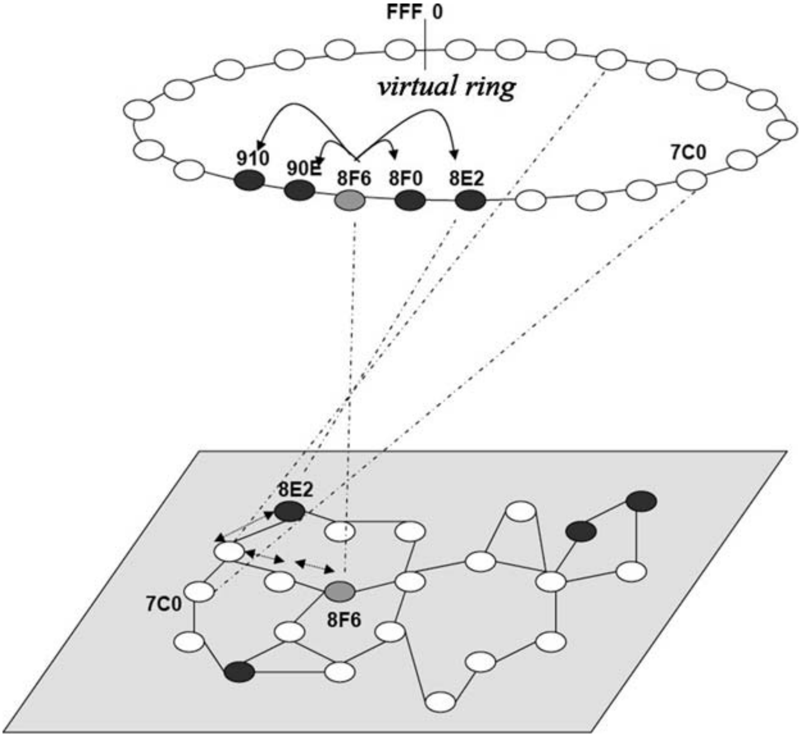


Fig. 8 The actual pair-wise connectivity of peers is used to establish route to the virtual ring predecessors and successors. For example, 8F6 has neighbors 8F0, 8E2, 90E and 910 in the virtual ring, though it is has actual connections with some nodes which are located in other arbitrary points on the virtual ring. As per the original VRR paper link state connections are used to determine pair-wise connectivity, however any other kind of abstract connections like social acquaintance may also be used instead. Now if 8FC wants to communicate with 7C0, it will choose to forward the messages greedily towards 8E2, however on the way, this message will go through a node which has direct connection to 7C0, and hence the message will be forwarded to its destination. This figure is reproduction the virtual ring example provided in the original paper (Caesar et al., 2006)

social networks can all be put together to realize a new vertically integrated overlay which we tentatively call *SocialCircle*.

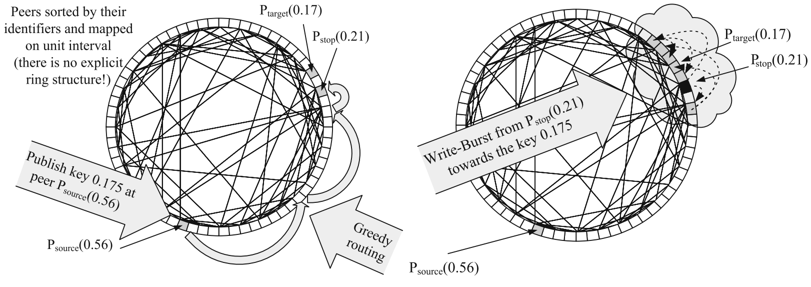
5.2.1 Fuzzynet

Traditional DHT designs rely on the existence of a ring in order to deterministically guarantee that any node can be reached from any other node by traversing the ring sequentially. While the ring ensures connectivity among any pair of nodes, ensuring the ring itself is challenging and expensive. In a system with large number of nodes and under continuous churn, guaranteeing the ring invariance continuously is unrealistic. Thus, even though it is true that if the ring exists then it guarantees routing, in practice since the ring invariance itself is often not met, routing failures are also relatively frequent. Fuzzynet [16] is a recent approach which uses a probabilistic approach for both routing and data placement in the DHT, but manages to successfully perform DHT operations even when the ring invariant is not met.

The essential idea in *Fuzzynet* is as follows. As in ring based DHTs, Fuzzynet assigns nodes unique identifiers on a ring. However Fuzzynet drops the requirement for every peer having a predefined deterministic responsibility range on the identifier space. Instead, it uses a probabilistic responsibility "fuzzy" approach, where a data item will most likely be stored on a peer whose key in the identifier space is close to the hash value of that data item (data key). Secondly, each peers need to know some other peers in its neighborhood in the ring. This is however different from knowing the precise successors or predecessors on the ring, but just some nodes in the vicinity, and hence there is no need to actively maintain this information. Data is replicated in Fuzzynet, by gossiping the data replicas in the vicinity of the data position on the identifier space. Such dissemination of data replicas does not significantly increase the overheads of Fuzzynet, since all the realistic systems (which use the ring structure) employ replication for fault tolerance and persistence anyway. An useful consequence of such data replication in Fuzzynet is the fact that a simple greedy routing query will find one of the replicas w.h.p. given sufficient network connectivity, even if the ring invariance is violated.

5.2.2 Lookup (Read) Operations on Fuzzynet

Lookup routing employs a greedy routing algorithm, where messages are forwarded everytime minimizing the distance to the target. The routing terminates if a data item D, which was looked-up is found. However, since there are no ring links and no predefined responsibility ranges, a peer might end up in a situation where it does not have any links which would lead the query closer to the target, nor does the node hold the requested data locally. In such a case the lookup query terminates


Peers sorted by their identifiers and mapped on unit interval (there is no explicit ring structure!)
P$_{target}$(0.17)
P$_{stop}$(0.21)
Publish key 0.175 at peer P$_{source}$(0.56)
P$_{source}$(0.56)
Greedy routing
Write-Burst from P$_{stop}$(0.21) towards the key 0.175
P$_{target}$(0.17)
P$_{stop}$(0.21)
P$_{source}$(0.56)

(a) Greedy-Approach (b) Write-Burst

Fig. 9 Schematic example of the two phases for publish (write) in Fuzzynet: (a) Greedy-Approach to reach in the vicinity where the data should be stored and (b) Write-Burst to disseminate the data in that region in a fuzzy manner

unsuccessfully. Nevertheless, analysis proves and experiments demonstrate and validate that with realistic parameters (e.g., in the networks with ***O(log N)*** degree and typical peer replication cost), data is found w.h.p. if it was published before.

5.2.3 Publish (Write) Operations on Fuzzynet

The high guarantees for Fuzzynet lookups lie in the exploitation of a particular Small-World property, namely the clusterization property, during the data writing phase. The write operation is performed in two stages and stores data *D* on *r* peers (replicas) in the vicinity of the data key *D(key)*. The first stage is similar to the lookup (read) phase and uses greedy routing to find one of the peers which are close enough to the data key *D(key)*. Once the write operation reaches the vicinity of the targeted key location, the data is seeded in the nearby peers by the self-avoiding multicast (a controlled "Write-Burst". The underlying idea is to use the clusterization property of the network and to reach as many peers as possible in the *D(key)* vicinity. The multicast has two parameters $-f_n$ *fanout* and *depth*. A peer contacts its fn closest neighbors to *D(key)* and requests to store the data item *D* as well as to continue the multicast process with reduced *depth*. The multicast avoids the peers which have been visited (already store data *D*) and terminates when depth reaches zero. Data *D* is seeded (stored) on all the peers reached by the multicast-burst. The two phases of the write process are shown on Fig. 9.

Experimental results show Fuzzynet to provide routing success which is upto an order of magnitude better than ring based routing (as in Chord). Fuzzynet's ability to realize DHT's indexing service and data retrieval without requiring the ring invariant makes it particularly suitable for systems not only with high churn, but also potentially in systems incurring mobility. Realizing the Fuzzynet rather than ring based DHTs on top of a social network is again relatively simpler since social network links need not be sufficient to guarantee connections to establish and maintain a DHT ring. Thus, we speculate that a hybrid approach using Fuzzynet and VRR can potentially be a good underlay for supporting collaboration middlewares, since they can explicitly take into account the nuances of social networks (implicitly present in collaboration activities) and mobility. This motivates us to explore a new DHT style overlay network which we will call *Social Circle*, the design of which and associated research issues will be elaborated at the end of this chapter.

Before doing so, we will first take a brief detour to explore some other identity related issues relevant in realizing a decentralized collaboration middleware.

5.3 Identity Crisis in a Large-Scale Decentralized World

In large-scale decentralized systems, there are two distinct identity related issues that may arise. Firstly, because of node mobility and dynamic IP address assignments, nodes' physical/IP address change over time. Many peer-to-peer networks and applications do not care about the specific peers, as long as they are connected to some peers. However other applications can benefit or even need to relocate an existing peer. For example, in a peer-to-peer storage system, if a peer goes offline temporarily and returns back, it is useful to be able to access the content stored in it, allowing for more efficient storage system redundancy maintenance algorithms. In other situations, being able to locate back a specific peer may be even more crucial, because of shared past interactions, and the trust and social relations established among the peers. This is particularly important in supporting a collaboration middleware. A simple solution to this issue can be to use a logically centralized directory service storing the up-to-date peer-to-address mappings. However, given that many of the peer-to-peer networks themselves work as decentralized directory services, one can also imagine using the peer-to-peer network itself as a self-referential self-contained directory service to store meta-information about the participants, including their current physical address. This basic idea has been proposed independently (and varying in details) in several academic as well as deployed peer-to-peer networks, including P-Grid [4], Microsoft's Peer Name Resolution Protocol (PNRP) and Skype. A similar self-referential directory can of-course in turn also be used to support *presence*, particularly in conjunction with a publish-subscribe mechanism!

Another aspect of identity in decentralized settings is that users can create bogus identifiers. A major security threat in such systems is if a resource rich adversary creates many bogus identifiers (Sybil attack [13]), then it can disrupt the functionalities of the system (denial-of-service), as well as more actively hurt the other genuine users. While this is a serious concern in general, in a social network context (which is inherently present – explicitly or implicitly – among collaborating individuals), this is easier to thwart, because social links are established based on either pre-existing real life interactions or referrals, or even if the link is established virtually, still it is sustained and rated based on real interactions among the players. Thus, social links can be a good mechanism to thwart multiple bogus identifiers. Notice that a limited number of identifiers for the same individual is not a threat of the same scale as a Sybil attack. Also, it really does not matter in many situations as to who is the real person behind the virtual persona, but what matters more is how this persona behaves. Trust and reputation based on the social network can thus further limit the effect of spurious as well as misbehaving peers. For example, SybilGuard [38] is one recently proposed approach which uses the social network to limit the effects of Sybil attacks in a decentralized system.

5.4 DHT Based on a SocialCircle

The social network can be useful both to improve the performance of normal routing, and to provide new functionality. Imagine that a user wishes to find a new team member with certain desired knowledge or experience. Ordinarily, the social network would be used in such a case as a medium for a broadcast query. In this work, we show how this function can be achieved much more efficiently.

Previous attempts have used social network links to bolster DHTs [26], prefering social links whenever possible, but nevertheless using the "normal" random DHT links otherwise. Such an approach still relies on using the "untrusted" links most of the time, but was perhaps as good as it gets under the older regime of DHT designs, where a completely connected underlying graph and ring invariance were considered necessary. Since then, *VRR* and *Fuzzynet* have respectively demonstrated overlay designs which work even otherwise. A hybrid of these two overlays has the potential to be a suitable substrate for collaboration middlewares in a decentralized setting. Another attempt to use social networks for routing in mobile ad-hoc networks [12] used centrally located "bridge nodes" that exchanged information about their egocentric social networks. However, in this approach, routing was still done using a network-layer ad-hoc routing protocol that just benefited from the information from the social network. In our approach, the social network itself is the basis of routing decisions, eliminating the need to exchange summary information. We would like to highlight that unlike the previous portion which was a summary of current state of the art, the following material in this chapter is currently speculative, and comprises mainly of open research issues.

5.4.1 VRR Adaptation for SocialCircle

Besides the *pset* and *vset* in *VRR*, we need another metalayer in SocialCircle, which comprises of all the social network links of a node, and call this the node's *sset*. In a wired network, one can assume that the *sset* of SocialCircle is used the same way as the *pset* in original VRR proposal. In a mobile or heterogeneous environment, whenever *sset* nodes can not be reached efficiently and directly using network level routing, the *pset* can be used for multi-hop routing in a manner similar to the original VRR. The *sset* will define a multi-hop route to the next *vset* node. Corresponding to each such point-to-point routes to *vset* members, a routing table entry will be maintained at each node. And similar to VRR, SocialCircle will also use the routing entries of other routing paths that go through a node to avoid navigating the virtual ring sequentially.

SocialCircle is developed as part of the mTeam project[3]. We present results of preliminary evaluation of the proposed system. The purpose of our experiment was to examine the VRR functioning in social networks. We have made tests on real social network graphs and randomly generated social network graphs. We have measured average stretch, which is an average ratio between the number of hops traversed by a message from a source to a destination and the length of the shortest path between this source and destination in the social network [10]. Stretch is a measure of how well the overlay network uses the underlying social network topology. A broadcast query would be wasteful in terms of overall messages used, but could discover a shortest path between the querying node and the intended receiver. SocialCircle routing is much more efficient, but can increase the length of the end-to-end path.

In the experiment we run VRR on a social graph. When the network gains a stable state, each node starts sending messages to 3 random nodes. This scenario assures that every node will be used and also ensures that even distant nodes will be on end-to-end paths, which are usually longer than average network shortest paths. The test results show the impact of increasing the number of nodes on average stretch and a comparison of stretch for shortest paths of different length. Each test was performed 3 times and average results were calculated. Random social networks were generated with the same parameters, using the "Scale Free" algorithm in the Pajek software (Pajek).

5.4.2 Average Stretch of SocialCircle

Figure 10 shows the average stretch for different network sizes. Stretch in SocialCircle on random social networks increases with the number of nodes. However, it does not exceed 1.7 up to 200 nodes, and stays below 1.5 for network sizes up to 170 nodes.

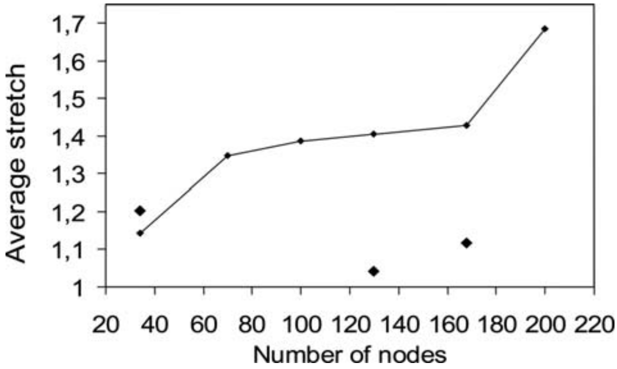


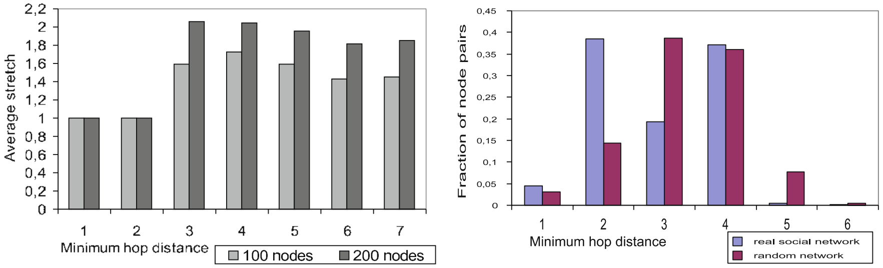
Fig. 10 Average stretch. Unconnected points show results for real social networks

Points connected with a line are test results on random social networks. The three remaining points that are not connected with lines are results of tests on real social networks. Results are quite similar at a small number of nodes (test on a sample network of 34 nodes). For real social networks with a higher number of nodes, stretch is much smaller than for random social networks.

To conclude, SocialCircle works well overall, with stretch below 70%; the approach works even better on real social network graphs, where stretch has been calculated as: 21% for a 34 nodes network, 5% for 130 nodes and 12% for 168 nodes. The reasons for these differences are discussed below.

5.4.3 Average Stretch Distribution

Figure 11a presents a distribution of average stretch for shortest paths of different length, for two different network sizes: 100 and 200 nodes.



(a) average stretch

(b) minimum distances

Fig. 11 Results of experiments. (a) presents average stretch per minimum distance. (b) presents distribution of minimum distances on real and random graphs

In both examples, there stretch is equal to 1 for shortest paths of one or two hops. These results prove that SocialCircle does not make mistakes for reaching close nodes. Stretch begins to increase at 3 hops. In the 100 nodes network, stretch is about 1.6, whereas for the 200 nodes network it is almost 2.1. For the 100 nodes network, the highest stretch is at 4 hops. In the 200 nodes network, stretch is highest at 3 and 4 hops.

For longer shortest paths, stretch is decreasing, and the differences between both networks are constant. These results are quite similar to results obtained for VRR [10].

5.4.4 Distribution of Minimum Distances

Figure 11b presents a distribution of minimum distances for a real and random social network. In the real network, most paths are up to 4 hops distance, and shortest paths of 2, 3 and 4 hops make up 94% of all paths. In the random social network, the distribution of paths is shifted towards higher distances, mainly on 3 and 4 hops. However, there are also relatively many shortest paths of 5 hops. This difference, together with the results presented on Figure 9, explains why SocialCircle performs better for real social networks. Since paths in real social networks are shorter, and stretch of SocialCircle for shortest paths of 1 and 2 hops is equal to 1, the results of SocialCircle routing on real social networks will be much better than for random social networks.

The presented preliminary results of the evaluation of SocialCircle indicate that for real social networks, the stretch of SocialCircle is likely to remain very low. This means that SocialCircle is able to route queries quickly, which would be of a special significance for session establishment and event notification in collaborative applications.

5.5 The Outlook for Future Works

5.5.1 Fuzzynet Adaptation of SocialCircle

The *SocialCircle* can be made further robust by reusing the ideas from *Fuzzynet*. Specifically, the *vset* need not comprise of precise predecessor/successors on the virtual ring, but instead use just a random set of neighbors from the vicinity and a fuzzy data placement based on gossiping over the *vset*. Such extensions are part of our future work, and has not been done yet.

5.6 Good and Better Peers

Using cooperative applications in a heterogeneous environment of mobile and fixed devices imposes a natural requirement for the diversification of peers. In fact, even P2PSIP foresees that possibility, as some SIP user agents may connect to P2PSIP peers using the P2PSIP client protocol. This solution is highly similar to a superpeer architecture. Some mobile peer may become superpeers just because they have Internet connectivity and are willing to share it with others. On the other hand, these peers may not be capable of running all superpeer functions that could be handled by a fixed node. For that reason, a three-tier hierarchy is a natural choice for the stratification of the peers according to their capabilities.

On the other hand, the question remains "how should different kinds of peers communicate?". A typical solution is to let superpeers communicate using a structured overlay, and peers communicate only with superpeers. However, such an architecture may be unsuitable in a mobile environment, where it is easy for a peer to lose contact with his superpeer. A higher connectivity among ordinary peers may be more desirable. In a different solution, all peers would use SocialCircle for communication without superpeer services, or for superpeer discovery (at the first or second tier). Third-tier superpeers receive service requests from their children in the superpeer hierarchy and serve this requests using SocialCircle routing.

An issue that remains is to determine which peers are capable to serve as third-tier superpeers. Based on information from the presence or trust management service, the overlay can automatically choose best candidates for superpeers or for peers needed for a special function (such as trusted routing). Note that this choice is not simple, since the overlay contains information about the availability and trust of individual peers, but does not contain an ordering of all peers. Obtaining all values and ordering them is not scalable. To solve this problem, it is possible to apply distributed sorting based on gossiping in the SocialCircle [22].

Another method of selecting superpeers would be to choose peers from the centre of community clusters in the social network. This method would have the advantage of supporting efficient message delivery for services such as publish-subscribe, as the superpeer could be used to receive publications and issue notifications to all peers in the cluster. The clustering would ensure that few messages would have to be sent to distant peers. To create community clusters and select appropriate superpeers, distributed clustering (community detection) protocols based on gossiping may be applicable [37].

6 Conclusions

Collaboration is one of the most promising applications of mobile network technology. The very nature of collaborative applications makes them suitable for a Peer-to-peer model, since many of these applications rely on sharing of information (expert knowledge, work results, new ideas) by people. Such sharing is inherently peer-to-peer.

Yet, in order to enable sharing, collaborative applications must have a number of functions that may require support of an infrastructure. The functional requirements for most collaborative applications have been described in this chapter. The required functions can be partially implemented using the Session Initiation Protocol and its P2P extension, P2PSIP. The emerging standard of P2PSIP points out a direction of enabling collaboration in mobile P2P environments. However, the full support of collaborative P2P applications requires further research work in the area of overlay design. In this chapter, we have explored a particular dimension of overlay design: the use of additional social network information from the collaborative middleware layer by a vertically integrated P2P overlay.

We outline how several currently stand-alone innovations can be put together to realize a peer-to-peer overlay for mobile environments, and also identify the outstanding research challenges to achieve the same. A common theme of the discussed innovations is the use of information from the collaboration middleware for the optimization of overlay operation. We have presented *SocialCircle*, with early results of a speculative approach that integrates these research ideas by using only the social network directly to create a structured overlay, which is incidentally also a first of its kind in terms of realizing the idea, even though many researchers in the community have speculated the need and possibility of such an approach [38], and devised partial solutions [26]. Preliminary results indicate that SocialCircle is likely to route messages very efficiently on real social networks, enabling efficient session establishment and event notification. These results prove the optimization potential of a vertical integration. Research that will further validate SocialCircle is already underway, together with several projects worldwide that attempt to exploit the powerful trends that bring ubiquitous, nomadic collaborative applications almost within our reach.

Acknowledgements

Most of the SharedMind implementation was carried out by Sally Ang and Michał Karwowski. The work on FuzzyNet [16] was carried out in collaboration with Sarunas Girdzijauskas, Wojciech Galuba, Vasilios Darlagiannis and Karl Aberer.

References

1. Groove networks. `http://www.groove.net/` (2004)
2. Freemind. `http://freemind.sourceforge.net/` (2008)
3. mteam: A creative environment for mobile knowledge workers. Research project supported by the Polish-Singaporean research program, `http://mTeam.pjwstk.edu.pl/` (2008)
4. Aberer, K., Datta, A., Hauswirth, M.: Efficient, self-contained handling of identity in peer-to-peer systems. IEEE Transactions on Knowledge and Data Engineering **16**(7), 858–869 (2004). DOI 10.1109/TKDE.2004.1318567
5. Auvinen, A., Vapa, M., Weber, M., Kotilainen, N., Vuori, J.: Chedar: Peer-to-peer middleware. In: Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS) (2006)
6. Baset, S., Schulzrinne, H., Matuszewski, M.: Peer-to-peer protocol (p2pp). Internet Draft, draft-baset-p2psip-p2pp-01 (work in progress) (2007)
7. Bisignano, M., Di Modica, G., Tomarchio, O.: Jmobipeer: a middleware for mobile peer-to-peer computing in manets. In: Proc. of the 25th IEEE Int'l Conf. on Distributed Computing Systems Workshops (ICDCSW 2005), vol. 791 (2005)
8. Bryan, D., Jennings, C.: A p2p approach to sip registration and resource location. draft-bryan-sipping-p2p-01 (work in progress) (2005)
9. Bryan, D., Lowekamp, B., Jennings, C.: Sosimple: A serverless, standards-based, p2p sip communication system. In: Proceedings of the 2005 International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA 2005) (2005)
10. Caesar, M., Castro, M., Nightingale, E., O'Shea, G., Rowstron, A.: Virtual ring routing: network routing inspired by dhts. In: SIGCOMM, Proceedings, vol. 36, pp. 351–362. ACM New York, NY, USA (2006)
11. Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., Gurle, D.: Session initiation protocol (sip) extension for instant messaging. IETF RFC 3428 (2002)
12. Daly, E., Haahr, M.: Social network analysis for routing in disconnected delay-tolerant manets. In: MobiHoc, Proceedings, pp. 32–40. ACM Press New York, NY, USA (2007)
13. Douceur, J.: The sybil attack. In: Peer-To-Peer Systems: First International Workshop, IPTPS, Revised Papers. Springer (2002)
14. Freedman, M., Lakshminarayanan, K., Rhea, S., Stoica, I.: Non-transitive connectivity and dhts. In: WORLDS, Proceedings (2005)
15. Garcia-Martin, M., Matuszewski, M., Beijar, N., Lehtinen, J.: Sharing files with the session initiation protocol (sip). Internet Draft, draft-garciasipping-file-sharing-framework-00 (work in progress) (2007)
16. Girdzijauskas, S., Galuba, W., Darlagiannis, V., Datta, A., Aberer, K.: Fuzzynet: Zero-maintenance ringless overlay. Tech. Rep. LSIR-REPORT-2008-006, EPFL (2008)
17. Greene, S.: Characteristics of applications that support creativity. Communications of the ACM **45**(10), 100–104 (2002)
18. Handley, M., Jacobson, V., Perkins, C.: Sdp: Session description protocol. IETF RFC 2327 (1998)

19. Herbjornsen, S.: Software support for creativity. Tech. Rep. TDT 4735, System Engineering, Department of Computer and Information Science, Norwegian University of Science and Technology (2003)
20. Horozov, T., Grama, A., Vasudevan, V., Landis, S.: Moby-a mobile peer-to-peer service and data network. In: International Conference on Parallel Processing, Proceedings, pp. 437–444 (2002)
21. Hu, Y., Das, S., Pucha, H.: Peer-to-peer overlay abstractions in manets. In: J. Wu (ed.) Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Networks, pp. 845–864. Auerbach Publications (2005)
22. Jelasity, M., Kermarrec, A.M.: Ordered slicing of very large-scale overlay networks. In: Proc. Sixth IEEE International Conference on Peer-to-Peer Computing P2P 2006, pp. 117–124 (2006). DOI 10.1109/P2P.2006.25
23. Kawulok, L., Zielinski, K., Jaeschke, M.: Trusted group membership service for jxme (jxta4j2me). In: Proc. IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob'2005), vol. 4, pp. 116–121 (2005). DOI 10.1109/WIMOB.2005.1512958
24. Kortuem, G.: Proem: a middleware platform for mobile peer-to-peer computing. ACM SIGMOBILE Mobile Computing and Communications Review **6**(4), 62–64 (2002)
25. Maibaum, N., Mundt, T.: Jxta: a technology facilitating mobile peer-to-peer networks. In: Proc. International Mobility and Wireless Access Workshop MobiWac 2002, pp. 7–13 (2002). DOI 10.1109/MOBWAC.2002.1166946
26. Marti, S., Ganesan, P., Garcia-Molina, H.: Dht routing using social links. In: The 3rd International Workshop on Peer-to-Peer Systems. Springer (2004)
27. Roach, A.: Session initiation protocol (sip) – specific event notification. IETF RFC 3265 (2002)
28. Rosenberg, J.: The extensible markup language (xml) configuration access protocol (xcap). IETF RFC 4825 (2007)
29. Rosenberg, J.: Simple made simple: An overview of the ietf specifications for instant messaging and presence using the session initiation protocol (sip). IETF draft draft-ietf-simple-simple-02 (work in progress) (2008)
30. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.: Sip: session initiation protocol. IETF RFC 3261 (2002)
31. Schulzrinne, H., Gurbani, V., Kyzivat, P.: Rpid: Rich presence extensions to the presence information data format (pidf). IETF RFC 4480 (2006)
32. Schwotzer, T., Geihs, K.: Shark-a System for Management, Synchronization and Exchange of Knowledge in Mobile User Groups. Journal of Universal Computer Science **8**(6), 644–651 (2002)
33. Shneiderman, B.: Creativity support tools. Communications of the ACM **45**(10), 116–120 (2002)
34. Singh, K., Schulzrinne, H.: Peer-to-peer internet telephony using sip. In: NOSSDAV, Proceedings, pp. 63–68. ACM New York, NY, USA (2005)
35. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, M., Dabek, F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup protocol for internet applications. Networking, IEEE/ACM Transactions on **11**(1), 17–32 (2003)
36. Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., Peterson, J.: Presence information data format (pidf). IETF RFC 3863 (2004)
37. Yoneki, E., Hui, P., Chan, S., Crowcroft, J.: A socio-aware overlay for publish/subscribe communication in delay tolerant networks. In: MSWiM, Proceedings, pp. 225–234. ACM New York, NY, USA (2007)
38. Yu, H., Kaminsky, M., Gibbons, P., Flaxman, A.: Sybilguard: defending against sybil attacks via social networks. In: SIGCOMM, Proceedings, pp. 267–278. ACM New York, NY, USA (2006)

Index

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking*,
DOI 10.1007/978-0-387-09751-0, © Springer Science+Business Media, LLC 2010