CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY, ISLAMABAD

Implementation and Analysis of Adaptive Bit-rate Video Streaming Architecture

by

Muhammad Hamza Bin Waheed

A thesis submitted in partial fulfillment for the degree of Master of Science

in the

Faculty of Engineering
Department of Electrical Engineering

2019

I dedicate this work to my parents for their limitless love, care and inspiration

CERTIFICATE OF APPROVAL

Implementation and Analysis of Adaptive Bitrate Video Streaming Architecture

by

Muhammad Hamza Bin Waheed

(MEE173026)

THESIS EXAMINING COMMITTEE

S. No.	Examiner	Name	Organization
(a)	External Examiner	Dr. Saima Nazir	FJWU, Islamabad
(b)	Internal Examiner	Prof. Dr. Imtiaz Taj	CUST, Islamabad
(c)	Supervisor	Prof. Dr. Amir Qayyum	CUST, Islamabad

Prof. Dr. Amir Qayyum
Thesis Supervisor
October, 2019

Dr. Noor Muhammad Khan
Head
Dept. of Electrical Engineering
October, 2019

Dr. Imtiaz Ahmad Taj
Dean
Faculty of Engineering
October, 2019

Author's Declaration

I, **Muhammad Hamza Bin Waheed** hereby state that my MS thesis titled "**Implementation and Analysis of Adaptive Bit-rate Video Streaming Architecture**" is my own work and has not been submitted previously by me for taking any degree from Capital University of Science and Technology, Islamabad or anywhere else in the country/abroad.

At any time if my statement is found to be incorrect even after my graduation, the University has the right to withdraw my MS Degree.

(Muhammad Hamza Bin Waheed)
Registration No: MEE173026

Plagiarism Undertaking

I solemnly declare that research work presented in this thesis titled "**Implementation and Analysis of Adaptive Bit-rate Video Streaming Architecture**" is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been dully acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and Capital University of Science and Technology towards plagiarism. Therefore, I as an author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS Degree, the University reserves the right to withdraw/revoke my MS degree and that HEC and the University have the right to publish my name on the HEC/University website on which names of students are placed who submitted plagiarized work.

(Muhammad Hamza Bin Waheed)
Registration No: MEE173026

List of Publications

It is certified that following publication(s) have been made out of the research work that has been carried out for this thesis:-

1. **Bin Waheed, M. Hamza**, et al. "MMCDN: A Novel Architecture for Multimedia Content Delivery Networks." Proceedings of the 2019 4th International Conference on Multimedia Systems and Signal Processing. ACM, 2019.

2. Gilani, Syed Sherjeel A., **Bin Waheed, M.Hamza**. "QoENGN: A QoE Framework for Video Streaming over Next Generation Mobile Networks." Proceedings of the 2019 4th International Conference on Multimedia Systems and Signal Processing. ACM, 2019.

(Muhammad Hamza Bin Waheed)
Registration No: MEE173026

Acknowledgements

I hereby thank Allah Almighty to whom we belong and to whom we shall return and by his blessings we are able to contribute our part in betterment and growth of the world of science and engineering. I am thankful and greatly indebted to my teacher and supervisor **Prof. Dr. Amir Qayyum**, without whom this work could not have been done. His guidance throughout my thesis has been a great help in moving to the right direction. His sincere advices, directives and constructive criticism on my work have helped me bring out this study to a conclusion. I would also like to express my gratitude to CEO and workers in mobilevas solutions who have allowed me to conduct my study on their IPTV service. I am also thankful to my parents who have been a source of encouragement and motivation throughout my life and especially for this work.

(Muhammad Hamza Bin Waheed)
Registration No: MEE173026

Abstract

The world of technology is going through a rapid transformation, which directly impacts the life of internet users. Therefore every user may be connected to internet everywhere in near future, such type of communication demands high speed connectivity with internet that may provide data in very less response time. In addition, the demand of multimedia contents over IP network is growing exponentially with the growth of internet users. Despite of great reliability and well defined infrastructure for IP communication, Quality of Experience (QoE) for the users has remained main focus, while getting multimedia contents with flawless or smooth video streaming in less time with great availability. Failure in providing satisfactory QoE resulting in churning of the viewers. QoE depends upon various factors, such as those factors related to network infra-structure have significant effect on the perceive Quality. Moreover, Video content distribution have greater impact upon QoE, which can be improved by using specialized protocols that are responsible for video transmission over internet. Unlikely, HTTP also allows video distribution, caching and streaming.

This research presents an architecture for multimedia distribution, caching and streaming by using off-the-shelf components for applications such as Video on Demand (VoD) and live streaming services. Moreover, discussion is carried out for the factors impacting QoE and their behavior is evaluated with QoE, the statistical data is taken from real architecture and analysis is made by using simulation tools and through real-time data, the response time and the throughput is also compared with the change of segment size in adaptive bit rate video streaming. Furthermore, video transmission is done by an adaptive bit rate mechanism over HLS standard. HTTP load balancing and DNS load balancing is also introduced to enhance QoE and to mitigate the challenge of single point of failure. The proposed architecture is validated and indulged as core component for video streaming in project "3GPP-IMS compliant E2E Mobile IPTV solution for 4G/LTE Networks"

Contents

List of Figures

List of Tables

Abbreviations

ABR	Adaptive Bitrate
API	Application Programmable Interface
CDN	Content Delivery Network
E2E	End-to-End
EXOPlayer	HLS Player or Video Player
HLS	HTTP Live Streaming
HAS	HTTP Adaptive Streaming
IP	Internet Protocol
ISP	Internet Service Providers
IMS	IP Multimedia Subsystem
LMS	Learning Management System
MOOC	Massive Open Online Course
P2P	Point to Point
QoE	Quality of Experince
QoS	Quality of Service
RTP	Real Time Protocol
RTSP	Real Time Streaming Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
STB	Set-top Box
URI	Universal Resource Identifier
URL	Uniform Resource Locator
VM	Virtual Machine
VOIP	Voice Over IP

WebRTC Web Real Time Communication

Symbols

$beta$	Number of Stalling Events
Sum	Summation
To	Starting Time
Tn	Ending Time
p	Iterative Index
C_p	Bitrate of Chunks

Chapter 1

Introduction

1.1 Background

In this era of internet, the usage of multimedia content is tremendously increasing that results in a congruent increase in demands of multimedia content delivery for meeting the requirements of users. The commonly used technique for streaming of video on demand and live video is the Hypertext Transfer Protocol (HTTP). Such protocol provides a mechanism for adaptive streaming known as Hypertext Transfer Protocol Adaptive Streaming (HAS) [1]. This mechanism became popular for using HTTP as its primary method of transportation in the internet applications. Moreover, this protocol is much easier to configure while allowing Network Address Translation [2]. In present networks, Quality of Experience (QoE) is the key focus of service providers, because of similar available resources in market that create hovering of users from one service to other in order to get their resources in lesser time with high quality. In case of videos, this phenomenon is observed significantly, as user gets annoyed when videos take too much time to play. Multimedia content is revolutionizing with the passage of time. It is experiencing enhancements in quality and resolutions of videos to a greater extent that increases the bandwidth consumption, however, better the video quality more space needed to store the content and much higher bandwidth should be available for the delivery of these

videos. Commercially available content caching and delivery solutions are not only expensive but also give less control to service providers over content management. IPTV services have the same type of effect as discussed above,these services not only include Video on Demand (VoD) service but also having capability of live streaming of videos. The content management and distribution in these services should be organized in a way that may enhance the quality of experience for users, as well as cost effectiveness for the service providers. Several global vendors such as Netflix, YouTube and Hulu use the technology of HTTP Adaptive Streaming (HAS) [3] to deliver the up to mark QoE to its end clients while keeping in view network quality or Quality of Service (QoS), which includes delay, packet size, packet loss ratio, jitter, response time and throughput [4]. In HAS, the original video content is converted into several bitrates, each bit-rate is further converted into number of segments by segmenter, which may have called as chunk. Each video segment or chunk has the video of fixed durations in seconds [5]. The path or address of chunks are stored in high level language file called a media presentation description or a manifest file. Once the user has a manifest file then it is very easy to stream the video content from the original server or rather from the content delivery networks (CDNs) [6]. The main feature of the HAS is to allow a multimedia player to adapt the video stream or to demand the optimal video stream, while keeping in view the condition of network and to switch the between different representations of same video content seamlessly and effectively. Several giant company holders implemented their proprietary protocols for adaptive multimedia streaming for example Apple [7], Microsoft (MSS) [8], and MPEG-DASH as mentioned [9] [10]. Most of them have similar features while using different formats and mechanism for HAS.

Presently, modern smart phones and mobile devices have much greater computing powers, capabilities and are able to perform multiple tasks effectively and in a short time. Moreover, these devices can use different ports and interfaces to connect to global internet [11] [12]. Mobility is the most fascinating and demanding feature of these devices as everyone wants to connect with internet from everywhere whether from their home, offices and from picnic places,which allows user

to stream multimedia application in all these places in effective way. Furthermore, multimedia services are generating very high percentage of downstream traffic. Such as the bandwidth usage of Netflix, a video content provider jumps to 40 percent of the Internet traffic and consumes more bandwidth then YouTube and Amazon. As mentioned by Cisco [13] that the global mobile traffic will increase 8 percent of the current mobile traffic between 2015 and 2020.

While keeping in view all these issues, this research presents an architecture for Multimedia content cache, management and delivery in a scalable environment, which is not only cost effective for multimedia service providers but also easy to deploy and to configure for their own content delivery network. The proposed architecture is developed using off-the-shelf components. It provides HTTP caching, DNS load balancing, HTTP based load balancing, HTTP web managing of cache that ensures the maximum uptime of multimedia services. A Transcoding server is also included in proposed architecture, which has the ability to do efficient encoding and decoding of raw videos into multiple segments of short duration having bitrates for low, medium and high resolution quality. This transcoding server is responsible for increasing QoE for the users as they are able to stream videos segments with low bitrates if they do not have enough bandwidth to stream high resolution video segments. The protocol used in transcoding of videos is HLS (HTTP Live Streaming), which is developed by Apple and is widely supported in multimedia servers moreover it is based on standard HTTP transactions and easily configured with every multimedia service. HLS basically converts raw video file into multiple short time chunks having extension of .ts file, which are pointed by a playlist having path of all these chunks, playlist have an extension of .m3u8, which is further pointed by master playlist, master playlist serves as bank for each existing video qualities. Both media playlist and master playlist are UTF-8 formats and having URI and descriptive tag about every index file in case of media playlist and playlist file in case of master playlist.

1.2 HTTP Live Streaming Protocol (HLS)

Internet provides access to users in order to get connected with the people around the world and also became main source of multimedia content distribution in the recent years HLS is one of the widely used protocols that enables users to choose bitrate of the video content by monitoring current network conditions for maintaining uninterrupted playback of the video at its best quality. It is able to manage interstitial content boundaries while providing modifiable and easily manageable framework for the encryption of media contents. HLS yields multiple interpretations of the same content, for example translation of the audio. It provides compatible infrastructure to get easily connected to large scale HTTP caching servers to enable delivery of content on a bigger scale.

Streams of HLS are formed in real time and contained over the webservers mainly HTTP servers. Formation of streams includes breakdown of the video content into smaller size segments, called as chunks, these segments have .ts file extension and also formed a master file with .M3U8 extension which referred as manifest file which act as an index file for the video segments. It mainly works for mpeg-2 format. Master playlist is the main file that information about all other index files that are separate for all the different qualities. This master file even also exists, when we are broadcasting only a single quality video. The figure 1.1 represents the playlist sequence in HLS.

While describing a broader concept, HLS hardware encoder takes input of video-audio file and encode it with H.264 encoder and AAC audio encoder and generates the MPEG-2 stream, which is the further splitted into .ts files, these .ts files are pointed by the .M3U8 file. The client player will receive master file, while requesting the URL of the video. This software enables users to watch flawless video playback.

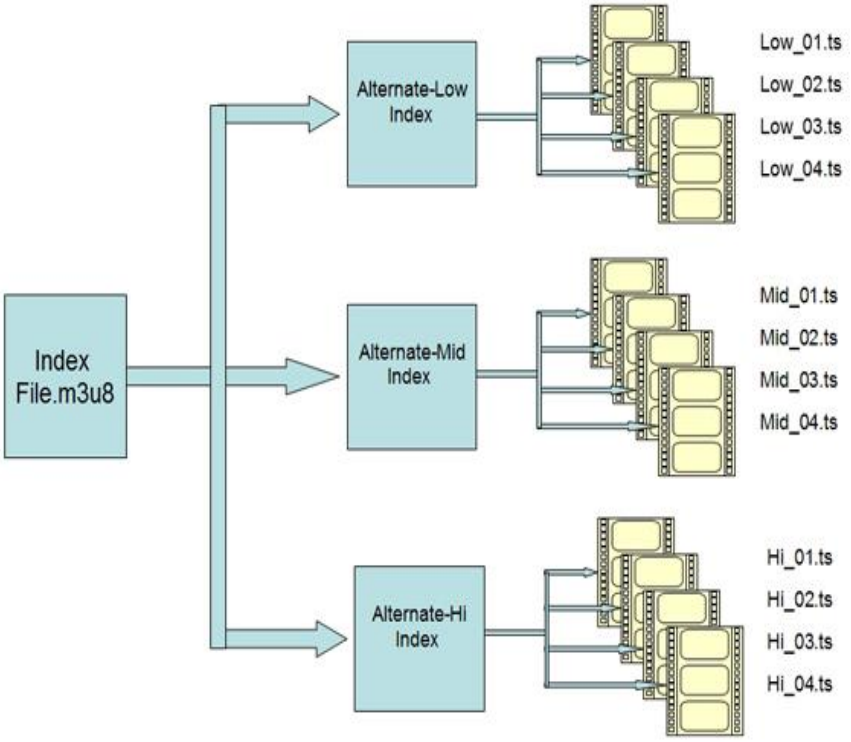

FIGURE 1.1: Playlist sequence in the HTTP Live streaming

1.3 Components of HTTP Live Streaming Protocol

There are three main components of HTTP Live Streaming Protocol:

1. Server: Responsible for encoding and decoding of the raw video file.
2. Distribution: Responsible for delivering video content to the end client software.
3. Clients: Enables users to watch video content in an unbreakable form

These are components can be shown in the figure 1.2:

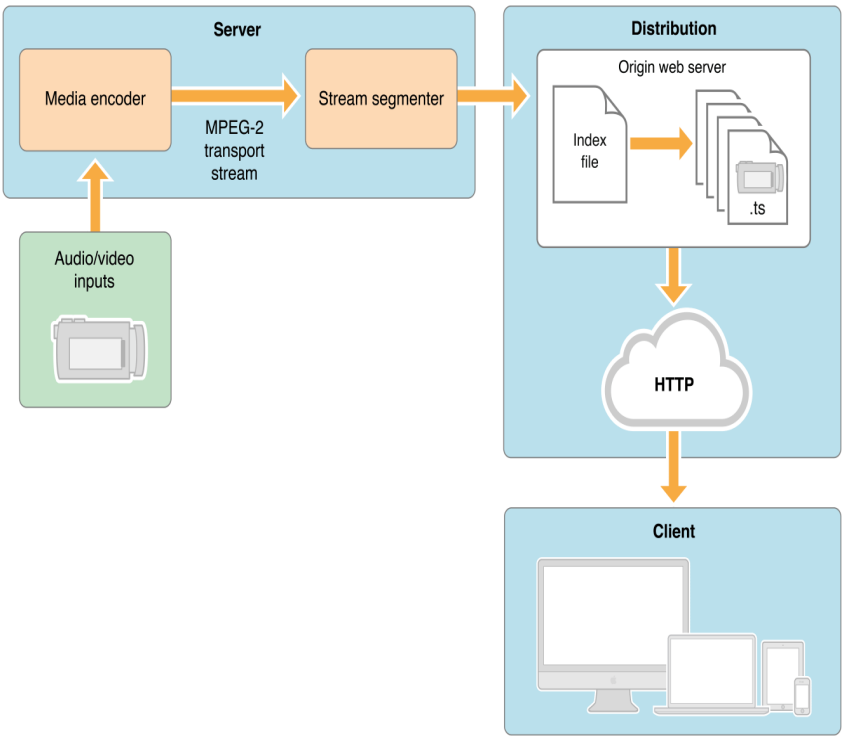

FIGURE 1.2: Block diagram of HLS components

1.4 Playlists

All the information about different quality streams and their specified paths are stored in playlists . There are two types of playlists:

1. Master playlist.
2. Media playlist.

These playlists are Unicode Transformation Format (UTF-8) text files, which contains special tags and URIs that have description about the desired media. Media playlist basically have the details about video chunks and master playlist have detail about the media playlist files.

Chapter 2

Literature Review

2.1 Introduction

Over the past few years, internet is spreading as nuclear reaction all over the world, leading towards an increase in network traffic and data causing network congestion and low response time of services over the internet. The increased request over the same servers can create hotspot on web servers [14] and the web servers are overwhelmed by demands of resources from many users at the same time, leaving the web servers in temporary unreachable state. These issues may easily be mitigated by using Content Delivery Networks (CDNs). This type of network brings data near the users and cache data in ready to use form, therefore throughput as well as uptime of the website is getting increased by using CDN technology [15]. More often, users even configured their own CDN's on technologies like Word press for their own websites. Most of the CDN providers, generating much revenues because of various undeniable features of CDN services that increase Quality of Experience for the users. These trends of functionality and revenue attracts researchers to work in this field and generate feasible and easily manageable solutions for the content providers.

An effective delivery of resources by using network elements can be declared as Content Delivery Network (CDN) [16]. It may be in many forms according to

need, as it may be centralized or decentralized, in an administrative domain or in more than one. Features that are important in CDN's are redirection of user's request, distribution among various servers, content retrieval and management of the content and servers [17]. The performance of the CDNs can be enhanced by making mirror servers of the original one and redirecting users to most suitable one on the basis of some criterion, which reduce the sudden load on the main servers. CDN can greatly reduce the response time and quality of service in video streaming architectures [18]. In CDNs, digital data is used in which static and dynamic data can be differently categorized and deal in that way to improve the performance of the service [19]. The architecture of CDN is based on three main roles that is resource provider, CDN provider and the end users. Mostly third party CDNs are used by many content provider organizations as they are unable to manage their own CDNs. Caching in CDN plays important role as the content is replicate on different servers located in different areas,to provide this cached content to nearest users of that area called surrogates or edge servers.

2.2 Related Work

This section includes both the related work of content delivery networks as well as the work for adaptive bitrate video streaming architectures.

Content Delivery Network (CDNs)

Content delivery networks remained the basic demand of every content provider company and it is increasing with the increase in multimedia content over internet, because of the much needed bandwidth. Therefore, research in CDNs becomes hot topic for the research in efficient delivery and streaming of multimedia content and several works has been proposed for the content distribution networks [20]. [21] Authors suggested architectures by using open source components in educational context and in commercial context as product for generating revenues.

In [22], authors present the architecture of distributed content delivery network (DCDN), in which they use available components on internet and proposed multi-level based architecture, while mostly focusing on load balancer and discussed the disadvantages of commercially used content delivery networks. Another architecture that presents a Hybrid content delivery network, in which authors used multiple commercially available services, combined them together and reduce the intermittencies of all individual commercial available distribution networks, and compare individual services with their Hybrid solution architecture [23], but as commercial services are used, therefore they are not specific for the traffic of single content provider and cannot cope up with the efficiency as required, moreover a commercial service is also expensive and cannot be used by organizations deal with large multimedia content. The centralized architectures are also presented and comparisons are available for the performance evaluation of content distributed networks and their limitations. Moreover, many literature is available for the different taxonomy of CDNs [24].

Adaptive Bitrate Video Streaming

Content distributors are also developing techniques and tools for their own content delivery efficiently and generating revenues from their own infrastructure. White papers presented the need and the key components that should be included in the content delivery architecture to optimize the delivery process and Quality of service is increased by enhancement of Quality of experience. Open connect is the architecture deployed by Netflix for video distribution over VOD services, not only used by their services but also they market their product to generate revenues. Another contribution in [25] that presents the architecture and gives the overview of the mobile streaming content delivery networks. This paper basically tells about a brief description of the MSM-CDN system, and presents the testbed design that is built based on these architectural principles. This research provides new platform for distribution of media content. Pedro Luis deployed a virtual architecture in the university of Quindio, in which he presents two architectures,

one for video on demand (VOD) and other for live streaming of video over HTTP for broadcast of educational material. It uses DASH protocol to test the capacity of its server by using stress tool by Apache and only provide the results of response time of server over different bitrate and number of users.

Moreover, it does not include the support of any cache server, load balancer and did not provide the effect of segment sizes. He created a virtualized environment and did not uses the simulator that actually can download the chunks created by DASH and provide optimal response time [26]. Another architecture proposed by Miran Taha, which is a virtualized network environment and created over VNX program over the single machine including cache serve support, uses DASH protocol to transcode the video into its segments and evaluate QoE metrices in his architecture by doing subjective analysis of number of stalling events, number of time video changes its resolution during playback time by conducting video session. Moreover, accumulative time and CPU usage is also evaluated in his paper. Miran suggested preferable segment size for the videos are 6-8 seconds after evaluation of mean opinion score (MOS) [27]. Anatoliy Zabrovskiy et al; presents simulations with simple architecture over mini-net and uses bit-codin service for transcoding of videos, they evaluated switching representation parameter iin his research by performing experiments on mini-net environment [28].

Hassan et al; presents research on streaming over WiMAX networks by using Opnet++ simulator upon one physical node, which have capability of easy mobility and evaluate the effect of various segment size of video on throughput and CPU computations and found that small segment sizes are useful in quality video streaming. This architecture was quite simple without any cache and load balancing servers [29]. Yae et al; deploy a real environment testbed for investigating initial delay, stall, throughput and CPU consumption in DASH video streaming. He used five nodes for his deployment in a simple network topology and analyze that small segments give better throughput then bigger segment sizes [30]. Jaehyun Hwang et al analyzes the different segment length for adaptive bitrate video streaming and conclude that smaller the segments fastest will be there reaction,

when network bandwidth oscillates or problems like network fluctuation and network congestion occurs [31]. Another study presented in [32], in which author's takes benefit from the CDN's and proposed the streaming service along with incorporating content delivery network distribution and its management and presented a study in the comparison of P2P video streaming and with CDN streaming and Analyzed the better performance is archived with CDN in contrast to P2P delivery, this work was done in simulations its real scenario is still missing yet. some studies were made on the streaming of adaptive bitrate video streaming in Educational context in which the author in [33], proposed a solution for streaming and video palyback using periscope, this solution was proposed to globalized the education of pathology. In [34], streaming is done for the education of online courses and teaching, in which good quality of experience matters for which solution is proposed for the delivery of educational notes over HTTP. A concept of Massive open online courses is presented in [35], that uses the video streaming application as main resource in delivery of educational notes. Another research is presented and validated over emulated scenario about the control of the video selection, while switching of the video is controlled by controller [36]. This adaption of video automatically done by using controllers at users end application and actuators at the server side and another control which is switch control loop performs role to control the switching between streams. The study [37], that incorporates the architecture WebRTC technology that transforms the way of communication as it enables the real time communication of the web clients. It includes the incorporating power of API's, in build video and audio codec, the web interfaces and plugin's . These were offered by some big companies like telestax [38] and Bistri, that provides the intelligent platforms for video conferences.

2.3 Comparison of Architectures

In the table 2.1, a complete comparison of architecture is provided on the basis of Protocol used, technology used and the their applications in which some state of the art architectures were compared that has been used in the broadcasting of

Comparison of architectures				
Research	Description	Technology	Protocols	Applied
F.A. Urbano [20]	Architecture for video streaming	Live555,RTSP, ffmpeg	RTSP,RTP,SDP	General
M.Y. Fuller [33]	provide architecture for mobile clients	Periscope,Andriod IOS	N/A	pathology
T. Hartsell [34]	integration of streaming in Educational purposes	MPEG,HELIX	RTSP	Educational
K. Sigama [35]	Implementation of higher education	LMS,HTML	N/A	Education
A.B. Johnston [37]	Real time communication project	HTML5,JAVA, VOIP	HTTP,SIP, RTP	Generic

TABLE 2.1: Architectural comparison.

educational material through video streaming architectures,some general purpose architectures and Architectures used in mobile IOS for enhancing the mobility has been discussed this comparison.

2.4 Existing Architectures of Content Delivery Networks

The commonly used CDN architectures are:

- Centralized
- Hierarchal
- Peer-to-Peer

When there was no or very less demand for video streaming, like it is today, the centralized architectures were adapted by CDN developers for caching and distribution of content. Such type of architectures has single point failure problem and vulnerable performance bottleneck of video servers. To reduce response time and increasing performance of video server, caching servers are introduced and bring near to users, where content is to be delivered [39]. These type of structures are easily manageable but with single point of failure as a major issue. Figure 2.1 shows the architecture of Centralized CDN's. In architectures that break down

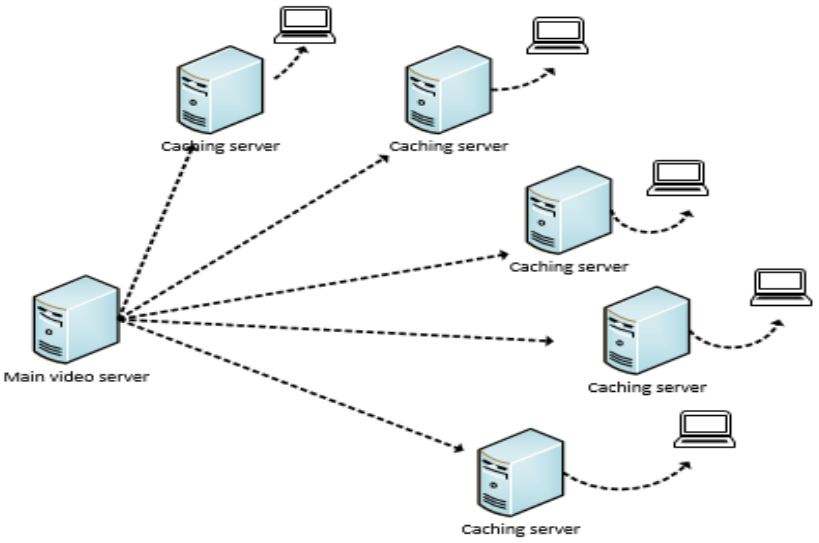

FIGURE 2.1: Centralized CDN's.

and divide the content distributed network into many levels are hierarchical CDN architecture. Which improves Quality of service, because the single point of failure is obsolete from this architecture. If one level fails to bring the requested data, some other level will bring it and provide it to users, forming multiple hierarchical levels [40], thus increasing the demand of video streaming server exponentially with the cost of this architecture. Figure 2.2 shows the architecture of Multi-level CDN's. Third architecture is Peer-to-Peer CDN architecture (Figure 2.3), in which CDN is also divided in many peer areas, where every peer connection acts as cluster of local video servers. Each local server has complete replication of original server and every peer server, serves to users in its area [41]. For VOD response time is better and the load balancing is occurring between peers to get maximum

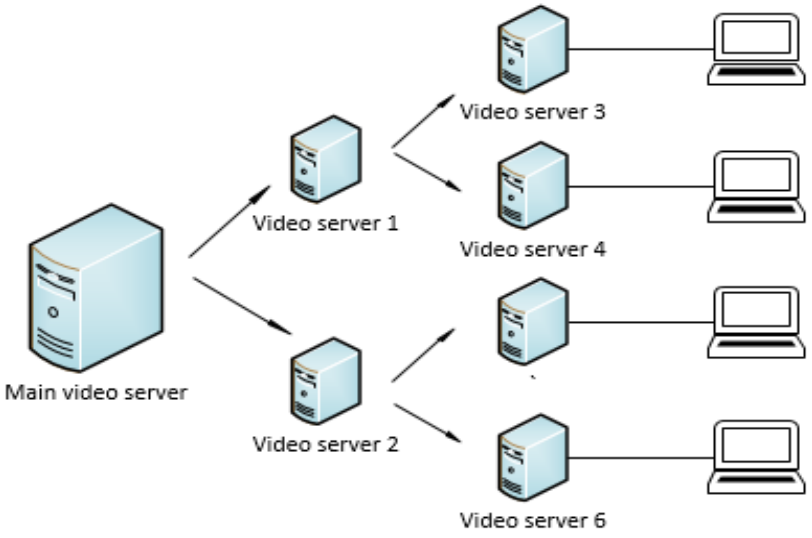

FIGURE 2.2: Multi-level CDN.

uptime. Many authors [42], proposed real as well as virtualized networks testbeds for carry out experiments [43]. The execution of both virtualized and real scenarios has been done in [44]. They provide comparison on different network scenarios and observe in the test results that packet loss and delay of real time scenario consume higher rate than simulated scenarios.

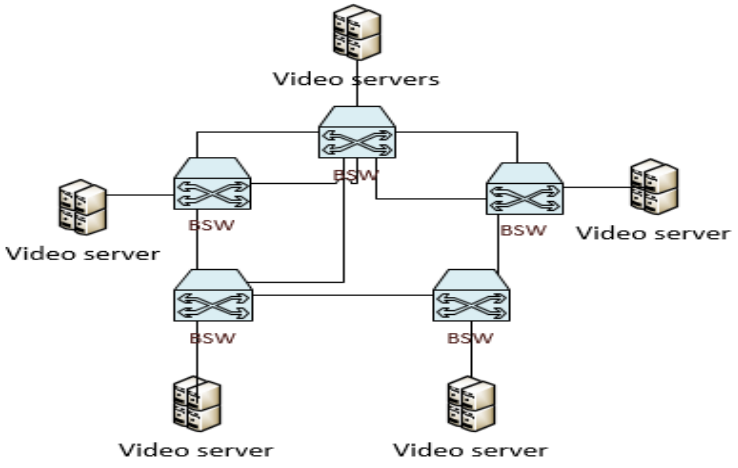

FIGURE 2.3: Peer-to-Peer CDN's.

2.5 Limitations of Existing Research

There are some limitations of existing research that are enlisted below:

1. Most of the solution discussed in the literature are theoretical and simulation based.
2. Architectures are without any real deployment while capturing the important performance metrics.
3. Many researches were carried out over ABR architecture, only some testbeds were found incorporating the effect of Content delivery networks.
4. The effect of Content Delivery Network is also judge through simulation and real deployment is still missing.
5. Researches providing the optimal segment size, but subjective analysis is missing.
6. Segment sizes up-to 15 seconds were analyzed.
7. The current available solutions over cloud or in market of Content Delivery Networks are expensive.
8. CDNs of the third party provides lesser control to the Content managing organization.
9. Most of CDNs are not designed for the one's own service.
10. Data managing and handling is difficult in third party solutions.

2.6 Problem Statement

To-date, most of the research in multimedia streaming architectures has been oriented towards analyzing their performance and quality of experience by using virtualized network testbeds or simulated environment on the basis of various

performance metrics such as delay and packet loss without incorporating content delivery networks and impact of varying segment size. However, using globally available CDNs make the multimedia streaming solutions more costly, and beyond the affordability of multimedia content providers. Moreover, a significant challenge of load balancing while streaming video contents has not been addressed by the existing multimedia architectures.

The proposed architecture analyzes the effect of RAM, throughput, response time, multiple transactions, varying segment size, and deployment over the same architecture with incorporating content delivery network in the real time application.

Chapter 3

System Description and Performance Parameters

3.1 Introduction

This section presents the more detailed information about the system and components necessary for real environment deployment. Conventionally, multimedia content transportation over the internet can be considered by using vast networks, which includes content delivery networks (CDN's), where user connects to ISP by means of wired or wireless media such as Wi-Fi. The video is delivered to users by using different network access environment through cache servers nearest to the users. When the content is not available in cache servers or replica server then the cache server, fetch the request of user from the main server and serve the user with the desired request and also save the copy of the content in itself also for the future requests of the users. The request can be fulfilled by other replica servers and main servers [45].

The principle system design contains both virtualized and real nodes. The system components which are main streaming server, cache servers and load balancers are deployed on real physical machine, every component is separate virtual machine (VM) which is deployed on single physical server. These machine are running

over Linux 16.04 operating system. Transcoding server is deployed on separate physical machine equipped with high processing power for fast transcoding of the videos. Moreover, DNS load balancing is achieved through a cloud application named dyna to provide maximum uptime of the service.

Transcoding server is configured with Transcoding Service for Live and Video on Demand content. This service takes live channels or recorded media content in any format as input and converts it to HLS format for streaming to clients and also has convert video into adaptive bit rate video. This transcoding service has following parts:

1. JSON parser: The JSON parser is responsible to parse JSON commands.
2. Demultiplexer: The Demultiplexer used for the seperation of voice and video from the video content that was inserted at the input of the demultiplexer and after seperation pass down in the pipeline for further actions.
3. Decoder: After demultiplexing of the audio and video decoding is applied on the stream and converted into raw video file.
4. Encoder: After decoding, we have raw file is used to encode it with our own desired encoder which can be H.264 and H.265.
5. Multiplexer: While before decode we demultiplexed all the content, now we multiplexed both the audio or video generated at the encoder output.
6. Adaptive bit-rate conversion: ABR feature is of much importance as after all the steps done, now we have to convert encoded media file into several content files of different resolutions and bit-rates, for this transcoding service convert them into media playlist and master playlist and make them in a format that they can be easily interpret at the clients end with specific modifiable bitrate.
7. Transcoding Instance Statistics: The service used in this research is also capable of determining important details about the service while on the run

time, this details includes ,messages about errors occurs during the transcoding, completion, not completed, time at which it starts , time at which it stopes, moreover these statistics can be used by application to automatically control all flow.

origin server is referred as main streaming server where all the transcoding media is stored, is also referred as origin server or content provider server which all the content provider transcoded media is stored all the new media will be stored there before it stream to users.

Cache servers are the replica server which serves the user with media content, if cache server do not have the desired content then it requests the content through the main server and serves the user and store a copy of the content itself on the mean time.

Load balancer/DNS load balancer are the front end application, which is configured in a way to balance the load between all the cache servers and DNS load balancing servers to save the single point of failure and provide optimal uptime of the service.

The proposed research design has ability to connect the external networks: internet or intra-net. Clients can be connected through any type of heterogeneous devices. For the objective as well as subjective analysis, proposed system have used both ways to request the video content for its performance analysis, first through simulated nodes or users by using J.meter simulator that have ability to request the adaptive bit-rate videos and generate the results of throughput, initial delay and latency by increasing the number of threads simultaneously. Second way is to generate request by real users through their mobile phones as well as through their laptops. An app is developed for the mobile users for request of videos. In which EXO player is deployed and configured, through which adaptive bi-trate video content is delivered to users without any hindrance, moreover this App is configured in a way that it able to provide all the real time statistics of the current video that user is viewing and store the information in text file and after

reading these statistics and watching the current bandwidth available for user, EXO player takes decision about the quality of the video that should be delivered to user. A script is written in the open source file EXO player to retrieve all the real time statistics of video streaming. When the bandwidth and bit-rate of the user are less then optimal threshold the EXO player switch the quality of the video in the seamless manner, which reduces the number of stalls and hence increasing the quality of Experience of the user.

3.2 Quality of Experience Metrics

This section contains information about the QoE metrics, which we have used in my research to calculate and analyse the effect of these metrics over the QoE and to get information about how there variations results in the change of overall QoE.

Initial Delay

This metric defines the period between starting time of loading a video and staring time of playing it [46]. The user end application formed a buffer length of T duration in seconds, this T depends upon the video segments size requested by the user and the time which the buffer is available for the maximum size related to QoS (Quality of service) factors such as availability of high bandwidth, variations in the bandwidth at user end and packet loss or packet drop rate. Bss is denoted as a buffer size of segment length, where Bssi ¡ Bssi+1, where I = 1 to M, and M is the maximium number of segment length. The user end application, which our case is EXO-player or HLS player will start to receive data at time To and accumulate them in the queue maintained by the application before its first frame delivered at Tn, which would have some delay or greater response time of the video.

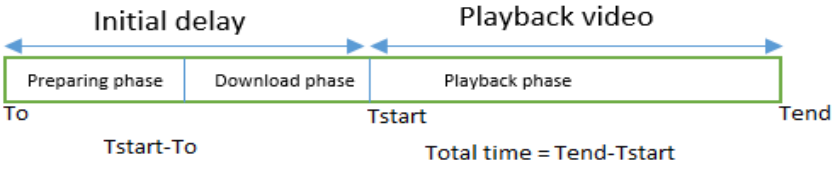

FIGURE 3.1: Time line during video playback

Quality Switching Frequency

Number of switches describes the number of times the quality changes from one bit-rate to another in the entire video session, it depends upon two factors: number of bits flowed through the channel in one second and the instability of the throughput at the user end, when these both parameter are not optimized then the number of oscillations or the number of switches increases, which will definitely reduce the QoE of the user as the greater the number of switches in the entire session the user will get irritated and may lead to switch the channel and service you are providing because when the switch occur a stall will also occur for much smaller time and also jitter and glitches are seen on the screen of the device user is watching any video, hence with the increase in number of switches the QoE will hurt, therefore we need to minimize the number of switches. The change of number of switches can be described by the following formula:

$$\sum_{p=1}^{n} g(C_p)$$

where p can be from 1-n, n is maximum time bit-rate change, C_p stands for the chunk current bit-rate, there the value of the above function can be represented as 1 if the $g(C_p - 1) \neq g(C_p)$, means if the current bit-rate and the previous bit-rate is not equal then the value of the function will be 1 and the value of quality switch will be incremented.

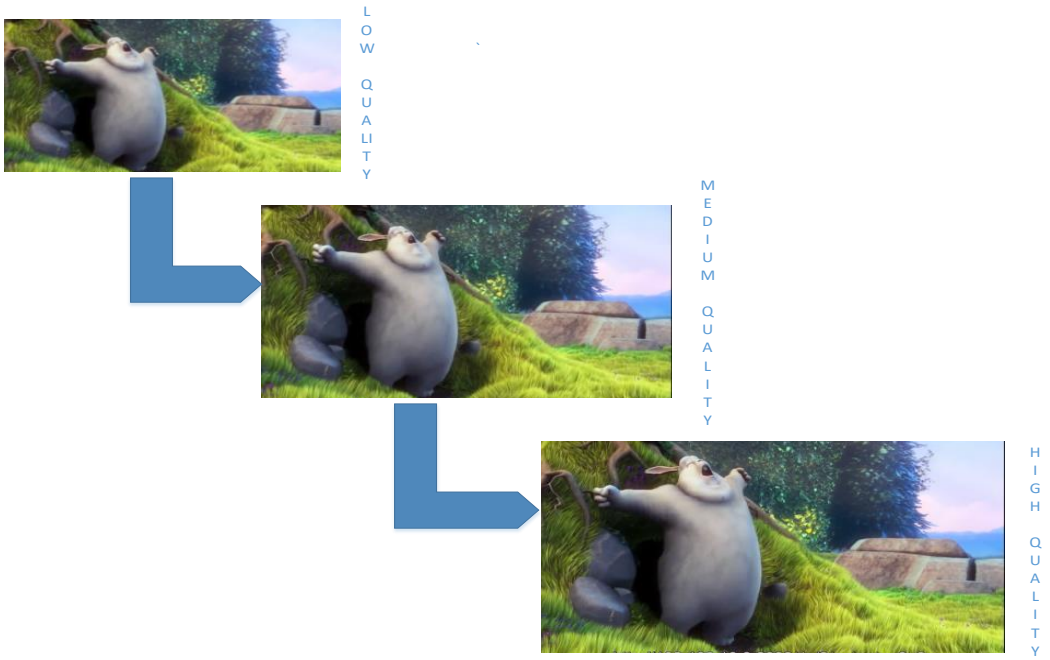

FIGURE 3.2: Quality switch diagram

Number of Stalling Events

Number of stalling events may also referred as number of pauses, stopes or number of time the video tends to buffer in the whole video sessions, this is very important parameter in the prediction of QoE as we have noticed it, as many stalling events occurs during the playback of video, the mean score for rating reduces because users get irritated as the frequency of pauses increases and average time of the video in complete video session increases.

$$Avg\_buffer\_time = \beta_0 + \beta_1 + \beta_2 + ...\beta_n$$

where β is the number of times the video stall's, therefore the Avg_buffer_time is sum of all the pauses in the video session. Figure 3.3 shows the stalling event that occurs during the playback.

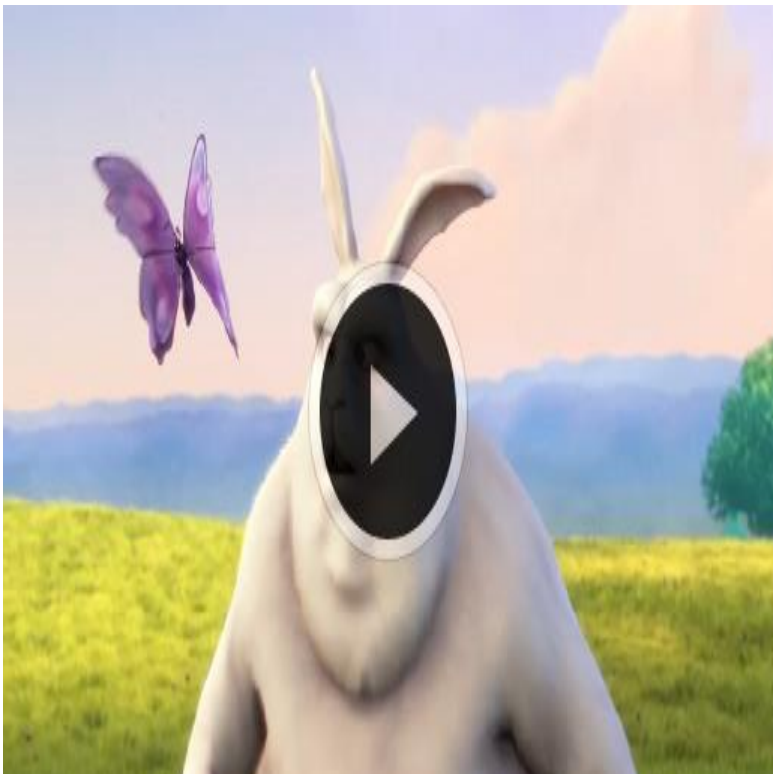

FIGURE 3.3: Stalling Event during video playback

Mean Opinion Score

Mean Opinion score is also calculated to see the effect these above mentioned parameter, that to which extent they may result in the variation of QoE. Moreover to check and analyse the behaviour of the metrics with subjective analysis and objective analysis, subjective analysis is the way to rate the Quality of Experience from people know about the video quality and have interest in generating some meaningful reviews, in this research case, i have used as defined criteria in all around the World for the rating of QoE, which can results in telling whether the user is satisfied or being annoyed. the criteria of rating can range from 1-5 , where the results from 3-5 tells about the users satisfaction and from 1-2 means that the user get annoyed.

Chapter 4

Implementation Methodology

This section presents the implementation of the overall model, in which flow chart, architectural diagram and configuration of every component is given in detail.

4.1 Introduction

This research presents the novel architecture of HTTP based content distribution network for IPTV service in figure 4.1 and its Quality of Experience analysis, which have DNS load balancing, HTTP based load balancing, caching servers, web-servers, origin server and transcoding server, that constitutes of encoding server and segmentation server, which will provide efficient video streaming to the end users. As shown in the figure 4, stream of video is coming from video source in case of Live video transmission and pre encoded videos are thrown to the input pipe of transcoding, where this encoding server will encode it into small segments of variable length and various bit-rate segments of the original video, using HLS protocol.

These adaptive bitrate segments of variable lengths are stored in origin server or video streaming server. All the content from streaming server will be pushed to webservers that are located on different areas, near to the end users, from where all the multimedia content will be cached in to caching servers near to the users

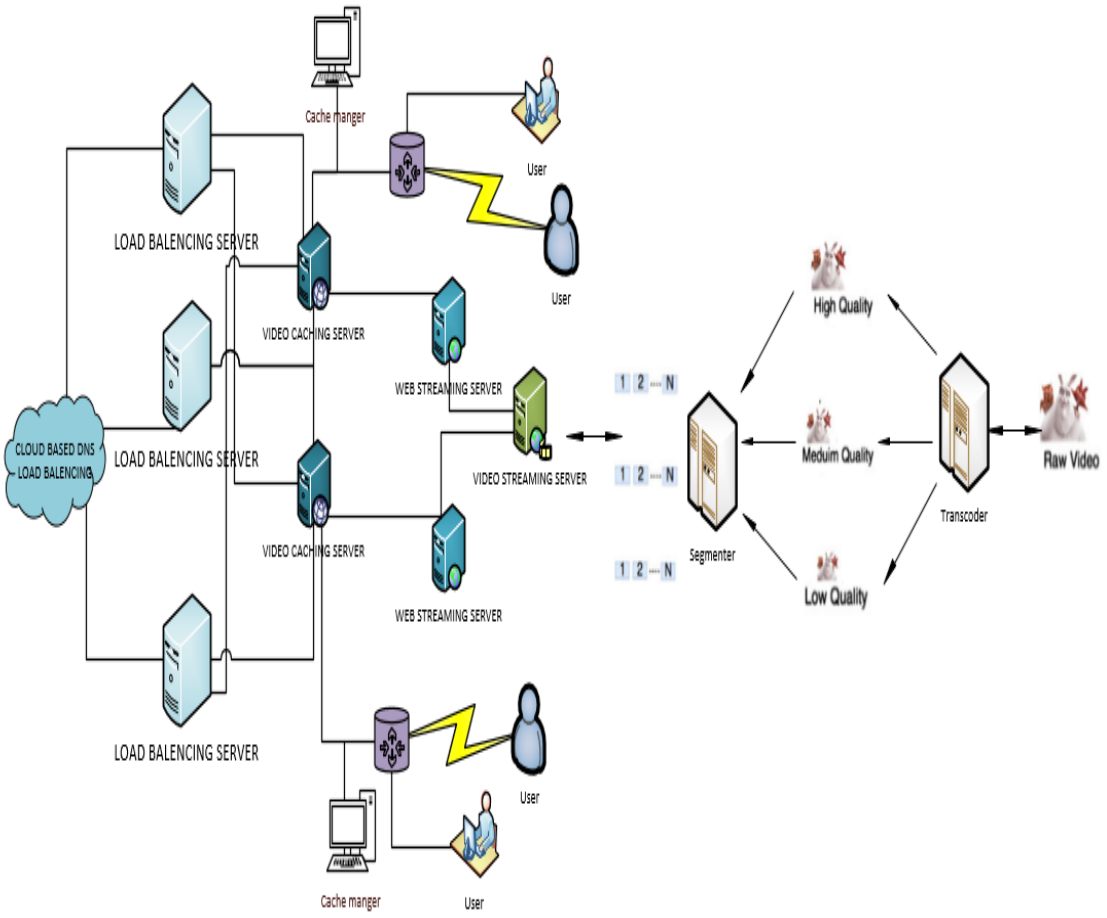

FIGURE 4.1: Proposed architecture for adaptive bitrate video streaming.

according to their location, when it is demanded by any of the user. When any content is requested by some user the caching server will check with in itself that whether, it contains respective content or not. If there is cached content, it will deliver cached content with no time to the requested user, but if the content is not cached, it will pull this content from the HTTP video servers and then store it and deliver it to the user. Two level load balancing implemented in this architecture is:

1. HTTP load balancing
2. DNS load balancing

HTTP load balancing is between multi-level architecture containing web-server and HTTP server as shown in the figure 4, which will deliver requests using round

robin fashion for mitigating the issue of single point of failure. This research provides DNS load balancing over cloud like (dyna) over which two or more IP's will be given to toggle the server, or for testing purposes another load balancer server is for load balancing of the load balancer. It will shift to other load balancer, if the current load balancer will be overwhelmed by lot of requests and becomes temporally unavailable.

The proposed architecture uses off-the-shelf components that are easily configurable, manageable and are efficient in terms of response time. Moreover, this architecture has ability to be scaled without disturbing the current configurations, only by adding more levels over cloud based DNS load balancing as well as over HTTP based load balancing. The caching server has also ability to be scaled easily just by configuring their configuration files, as IP of multiple server can be given in its configuration of caching servers. There are web mangers for every caching server over the internet, upon which the configurations, statistics can easily be seen and monitored. Videos content will be delivered to the user, using HLS players, in adaptive bit-rate that will automatically change the bit-rate of the video as the network bandwidth and bit-rate of the video content changes at the user end. If the bandwidth of user is enough to smoothly run video at higher bit-rate, then the high quality segments generated by encoding server will be delivered to the users otherwise low or medium quality segments will be delivered.

4.2 Flow chart of the Architecture

The flow chart in figure 4.2 describes the complete flow of the architecture in which the role of cache service is also demonstrated with the help of diagram. it only describe the flow of system that includes all the main servers, cache servers, network connection and the request handled by load balancer and cache servers.

FIGURE 4.2: Flow Chart of Proposed Architecture

4.3 Components and Technologies Used

The components and technologies used are explained in context with every architectural component in below section.

Load Balancer

The implementation of load balancer is in Ubuntu 16.04 LTS on open source platform called Nginx. Web-servers as well as load balancer are both implemented by using Nginx platform, the task of the load balancer is to distribute request or balance requests from the user to multiple web-server, so that we can able

to provide content to user from overwhelming the single web-server. Nginx used following algorithms for load balancing:

1. Round robin algorithm.
2. Least connected algorithm.
3. Weighted load balancing algorithm.
4. Session persistence.
5. IP hash load balancing algorithm.

The algorithm that is preferred from all of the available algorithm is IP hash algorithm in which client IP is used as hashing key for its redirection, means this algorithm generates a unique hashing key by processing the IP address of the client requesting for desired web-server, in IPV4 it see the first 3 octet as hashing key for the client IP address, for example 192.168.2.1 and 192.168.2.3, in this example it will always redirect user to same web-server because the network address of the both users are same. It means if we want to use this algorithm then the network address should be different for every user to route them to specific web-server according to the hashing key generated by this algorithm. However, for testing purposes we have used round robin algorithm, because all VM's have same network addresses, therefore have same hashing key.

The load balancer using Nginx can be configured, but before this we need to install Nginx platform in our Linux distribution, we have used virtual machine with RAM of 1GB for load balancing. Nginx can be installed using command in command line:

Sudo apt-get install Nginx After its installation completes, we can configure load balancer by going into the configuration file of Nginx which located at the path /etc/nginx/site-available, by enabling them to listen on port 80 and also defining the servers available for load balancing with proper tags, you can configure Nginx load balancer.

HTTP Caching Server

The task of reverse proxy is same as that of load balancer that to balance the traffic coming from different users, but we added this to add another hierarchy of balancing the load to increase efficiency of our Webserver. HTTP caching will cache content as requested by the user, first it will check in its log file that, whether it has content or not, if the content is missing then it will bring it in its server and maintain its log file. We have used varnish caching server; varnish is also open source tool through which Web acceleration as well as HTTP Caching is done. The strategy followed by varnish for retrieval of content is pull strategy. Multiple features of varnish that pulls us, towards its implementation in our architecture:

1. Time to cache the Web page content..
2. The allocated storage for content cache by default is 1 GB but can be changed to any number.
3. The settings, which can allow you to specifically block the data, you don't want to cache.
4. It allows to cache chunks of file, for example not to cache complete 4GB video but to cache its chunks.
5. Varnish uses a built-in tool called backend polling to check on the backend server and continue serving cached content if the backend is unreachable. In the event that Varnish detects downtime, it will continue serving cached content for a grace time.

HTTP caching server can be configured on Linux distribution; varnish uses pull based strategy to cache videos and content from the main server, pull strategy means that when any user request for desired content then cache server will cache the content in itself as well as distribute it to the user also. Varnish will be running on port 80, for its configuration, we will edit the configuration directory which is default.vcl and is available on specific path which is: /etc/varnish, therefore we can

edit configuration file for configuring backend servers using Nano or Vim editors. Moreover, varnish service file is on path /lib/system/system, where all the service settings of varnish is configured.

The figure 4.3 depicts about the caching server stats taken from the command line terminal of Linux, showing the important parameters like, cache miss, cache hit etc.

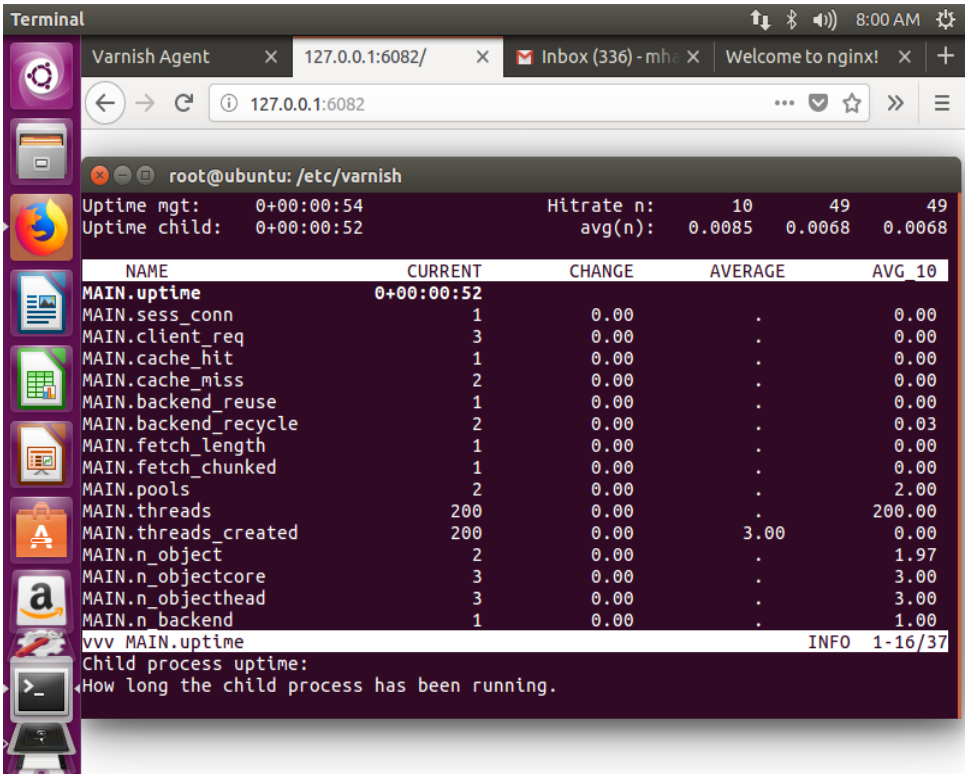

FIGURE 4.3: Caching Server stats from terminal.

Varnish Agent

All the caching server in our architecture can be monitored and the performance of the caching server can be analyzed using varnish agent, it's an open source utility, used to couple with any varnish server for monitoring and data analysis of the server's statistics. Varnish agent is web based tool and operates at 6085 port of any server in which it is configured, combined with varnish dashboard gives graphical statistics, making easier to understand the performance of the entire

caching server. The figure 4.4 shows the statistics taken from the web interface through which we can control our caching server activities without going into the VCL file of the Caching server and without using Linux interface, it can be configured and changed from server by going to this IP address.

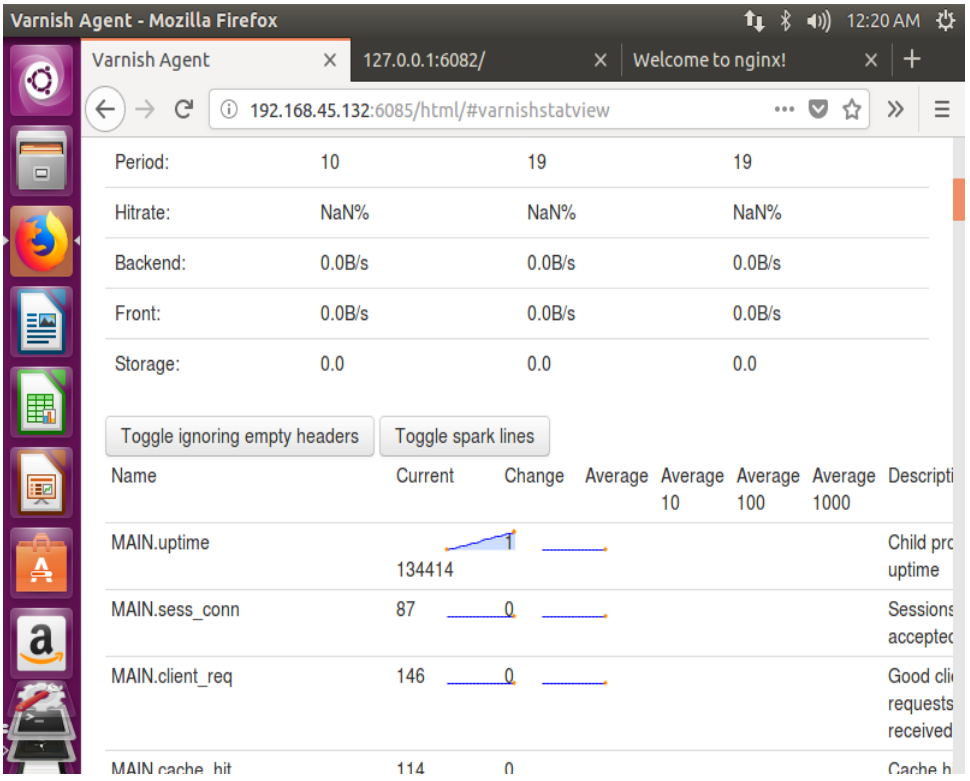

FIGURE 4.4: Caching Server stats over Web interface.

Configuration of Transcoding Server

The transcoding server is configured on separate high power Lenovo machine, and Linux distribution 16.04 LTS with RAM of 4 GB is virtually configured using virtual box on the machine for time efficient and good quality transcoding of the videos into multiple bitrates. Before the deployment of transcoding server, I had install various supporting packages and libraries by using command terminal of the Linux, first of all we need to install and configure h264 encoder and its library that for installing and configuring we need to install yasm assembler that will parse the script of the installing module of the h264. This can be install by using command: sudo apt-get install yasm

We have to download the h264 encoder library in my case I had used x264-97eaef2 version, which was in tar format means, we have to unzip this file by writing command in terminal:

tar xvf x264-97eaef2.tar.gz

this command will unzip the library and you will have extracted x264 library, for configuring it we have to go it folder with name of x264-97eaef2 by using command:

cd x264-97eaef2

After this configure this library by using command:

./configure –enable-shared –disable-static

make

by executing these commands x264 library will be installed in your distribution. Another important this is FFmpeg, FFmpeg is a multimedia framework and having capability of encoding, decoding, mux, demux, filter, stream and play the videos and everything that is between the human creations and the machines. This framework can be installed after downloading it from its source, I had used ffmpeg-3.2 version, it was also in tar format therefore I had to extract it, which was done through commands given below:

tar xvf ffmpeg-3.2.tar.bz2

sudo apt-get install libsdl2-dev

./configure –enable-shared –enable-libx264 –enable-gpl

make

sudo make install

After this had install the transcoding service, I had used in the project of IPTV, this can be easily installed using by extracting it:

Tar xvf transcoding-service.atr.gz

cd transcoding service

make

After successful installation of transcoding service, we can run transcoding service by going into the folder of transcoding service by using command line terminal. The command used for running transcoding service is:

cd transcoding-service

./trans-service –input=../input-pipe –output-pipe=../output-pipe

This command will run the transcoding service and now the input pipe is opened for the Json incoming command, after running the command which can be seen in the figure below, we have send command for the transcoding of the video through Json command, for this open another terminal and write whole command starting with echo, small portion of the command is given below to explain the functionality of some commands:

”Command”: ”START-ABR”,”Output”:”/home/hamza/Downloads/1/output.m3u8”, ”Manifests”:[”InstanceID”:”1”,”Input”:”/home/hamza/Downloads/SampleVideo-1280x720-10mb.flv”,”Output”:”/home/hamza/Downloads output.m3u8”,” Output-Format”:”HLS”,”HlsIndexEntry”:”1/output.m3u8” ”HLSOptions”:”SegmentDuration”: ”10”,”MaxPlaylistEntries”: ”10”,”BaseUrl”: ””,”SegmentPath” ”/home/hamza/-Downloads/1/”,”VideoEncoder”: ”H264”,”VideoBitrate”: ”936000”,”Width”:”320”, ”Height”:”240”,”AudioEncoder”:”AAC”,”AudioBitrate”: ”64000”,”Channels”: ”2”, ”SampleRate”: ”44100”

1. Command: This should always have the value “START” for this command.
2. Output: This shows the path of ABR .m3u8 master playlist file.
3. InstanceID: This field represents the unique Id with which a new transcoding / segmenting instance will be created. This instance Id will be used in all future commands to refer to the created transcoding / segmenting instance. For example, in INST-STATS, STOP commands etc.

4. Input: This specifies the input video source that we want to transcode, which can be in any format

5. Output: This output command is the path of the .ts file, which is playlist file, that have information of all .ts segments of the video.

6. Output format: This specifies the format of the output. Currently it can have the following values:

   (a) HLS

7. HLS options: This should only be present if the "OutputFormat" is HLS.

8. HLS options: This should only be present if the "OutputFormat" is HLS.

9. Video Encoder: This specifies the video encoder to be used for output video. It can have the following values:

   (a) H264

   (b) H265

10. VideoBitrate: Bit-rate in bits per second of the output video. This is an optional field in general.

11. Segment duration: it is the duration of the .ts segments, that how long a single segment can be of.

12. Width: Width of the output video if it needs to be scaled. This is ignored if the "VideoEncoder" is set to "COPY".

13. Height: Height of the output video if it needs to be scaled. This is ignored if the "VideoEncoder" is set to "COPY".

14. AudioEncoder: This specifies the audio encoder to be used for output audio. It can have the following values:

    (a) AAC

    (b) COPY

15. AudioBitrate: Bitrate in bits per second of the output audio. This is an optional field in general.

16. Channels: No of channels of the output audio. Ignored when "AudioEncoder" is "COPY".

17. SampleRate: Sampling rate of the output audio. Ignored when "AudioEncoder" is "COPY".

Figure 4.5 shows the whole procedure of transcoding of the video in to its segments, in which i used 2 Linux command terminal in the first terminal window whole command that is described above is passed to the transcoding service, that is running in the background in the second command line terminal, first we have to start the transcoding service which opens and input pipe for the reception of any video source, when the command is passed from the other terminal then transcoding gets starts moreover, in this figure you can also view all the three video instances named as instance ID 1,2 and 3 that starts at some specific time and stooped after the completion of transcoding. The start time of all the instances are same but the stopped time is different because of the difference in the bitrate and resolution of the video, high resolution video consumes more time then the lower one.

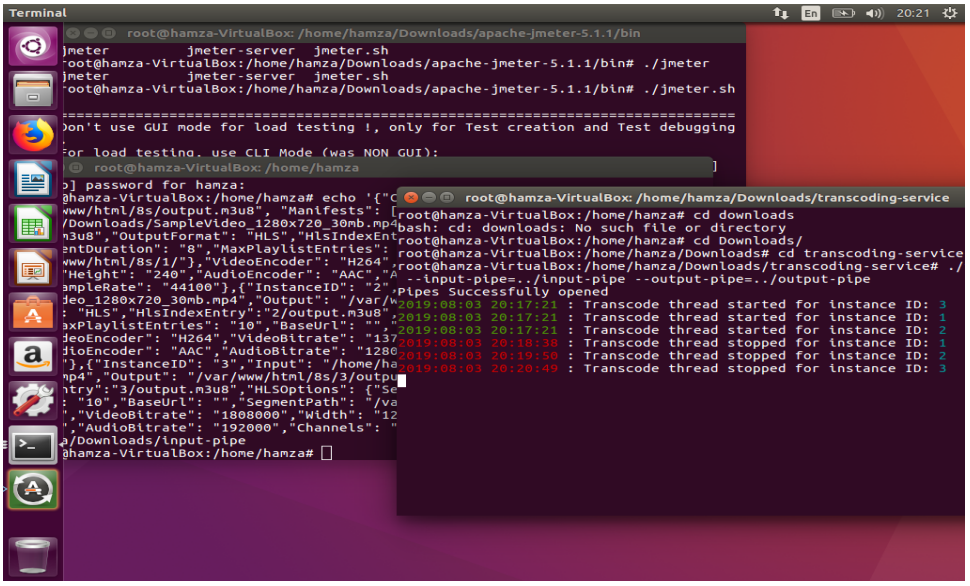

FIGURE 4.5: Transcoding server procedure window terminals.

Figure 4.6 shows the folder containing transcoding video, in which you can able to see master file with name output.m3u8 with three folders named with 1,2 and 3 having different resolution in it. Moreover folder 1 shows the palylist file with .ts segments formed in the result of segmentation.

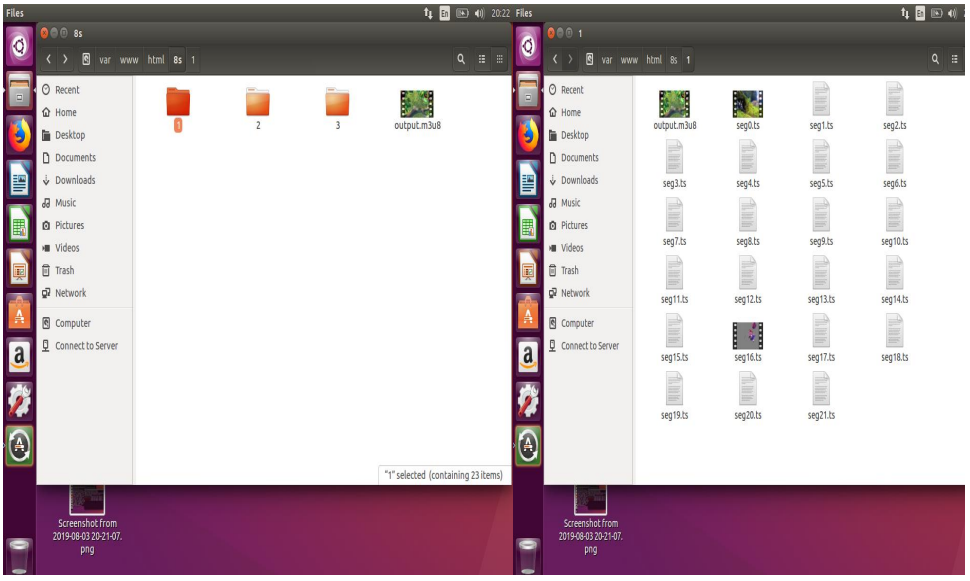

FIGURE 4.6: Folders depicting .ts files with master playlist and media playlist.

4.4 Benchmark Comparison with Existing Architectures

Testbeds	**Testbed for educational video streaming**	**Testbed for QoE analysis**	**Simulated testbed on Mininet**	**Real testbed**	**Testbed for streaming over Wi-max network**	**Proposed testbed**
Simulator used	Apache stress test tool	Not discussed	Mininet	Not discussed	Opnet	J.meter and siege stress tester tool
Nodes tested	01	01	01	05	01	06
Mobility	Not discussed	Easy	Easy	Low	Easy	High
Scalability	Not discussed	High	High	Low	Medium	High
Performanc e metrics	Response time	CPU consumption , segment size effect and MOS of staling events	Switching representations	Segment bitrate, video bitrate and initial delay	CPU consumption and throughput	Response time, throughput, effect of segmentation, initial delay, effect of RAM
Realism	Simulation	Emulation	Simulation and emulation	100% real	Simulations	Real and simulations
Network topology	Simulation	Emulation	Simulation	Real	Simulation	Real
Cache service	No	Support	No	No	No	Highly efficient
Load balancing	No	No	No	No	No	Yes
Preferred segment sizes	No	4-6 sec	Not discussed	Small segments	Not discussed	6-12 sec
Language	Not discussed	Extensive language	Python	Not discussed	Not discussed	JSON'S, html and c++.
Virtual machine	No	Linux	Linux	No	Not discussed	Yes
Network topology	Simple	Complex	Simple	Simple	Simple	Complex
Wireless tested nodes	No	No	No	No	Yes	Yes
VM chains distribution	No	Support	No	No	Not discussed	Yes
Current deployment	NO	NO	NO	NO	NO	In production of IPTV service

FIGURE 4.7: Benchmark Comparison with latest testbeds.

Chapter 5

Simulations and Results

5.1 Introduction

The Proposed solution is using Big Bunny video [47] for the evaluation of QoE parameters and its effects on Quality of Experience by changing different parameters of video. For the production of HLS content, as described earlier, transcoding server is used, that further uses ffmpeg and Lib-X264 library. This HLS content contains 24 representations that constituents of 3 levels of video quality such as: Low, Medium and High quality.

These 3 levels of video quality are encoded in different bit-rates, which are gradually increased from low quality to high quality, in this research case, a low quality video, which has a resolution of 320x240 with a bit-rate of 936000bps, a medium quality video, which has a resolution of 640x480 with bit-rate of 128000bps and in the same way high quality video with resolution of 1280x720 with bit-rate of 180800bps, with separate audio bit-rates for each quality representation gradually increased from low quality to high quality. 64000bps for low quality and 192000bps for high quality with sample rate 44100 or 44.1khz. Furthermore, this research provides 8 versions of the same video having different video segment length or chunks sizes, that are starting from 2s. These segment sizes are 2s, 6s, 8s,10s, 12s, 15s and

20s. By generating these different segment sizes, the effect over throughput, initial delay, quality switch, stalling events and QoE has been observed by changing important parameters.

In the first test, observation is made about initial delay by varying the number of users. In which users simultaneously try to fetch the same content. For performing this test, J.meter is used that have capability to retrieve HLS streams by increasing the number of threads or users simultaneously and have capability to retrieve all relevant information such as, initial delay or response time and throughput. Therefore J.meter is allowed to run over separate machine in Linux distribution platform and users are varied from 1 to 150 and response time have been observed for each quality representation from low to high. While considering this response, it has been observed that almost up-to 50 users, there is no big difference while fetching the high quality or low quality video but as users are increased from 50, one can easily observe that the gap between three video qualities increases significantly and also response time for each quality increases uniformly with increase of users. Therefore 50 users can be declared as the point of fluctuation or fluctuating point in this case, which shown in figure 5.1. The behavior can be observed in the below figure, moreover the J.meter configuration is also depicted in figure 5.2.

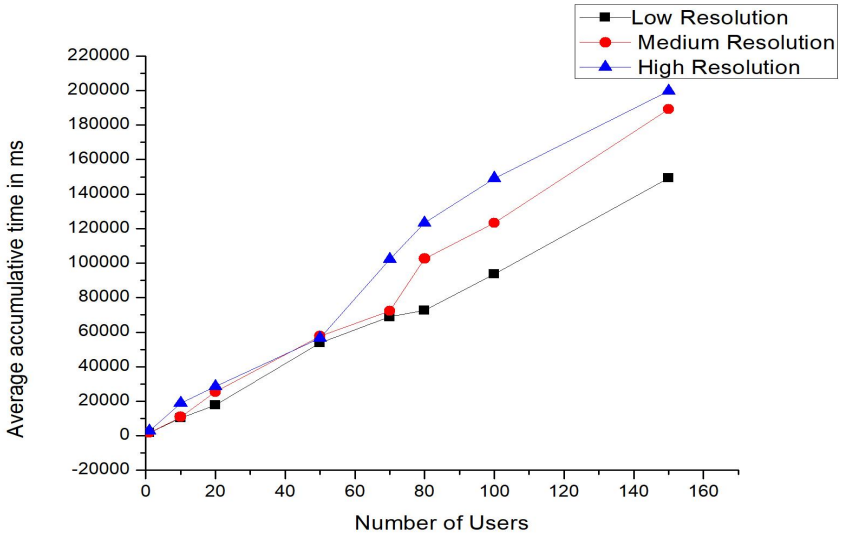

FIGURE 5.1: Response time and accumulative time graph with number of users with CDN.

The number of threads or can be seen in the J.meter console in figure 5.2, where I have put rampup period 0, which means that all users will simultaneously request for the content.

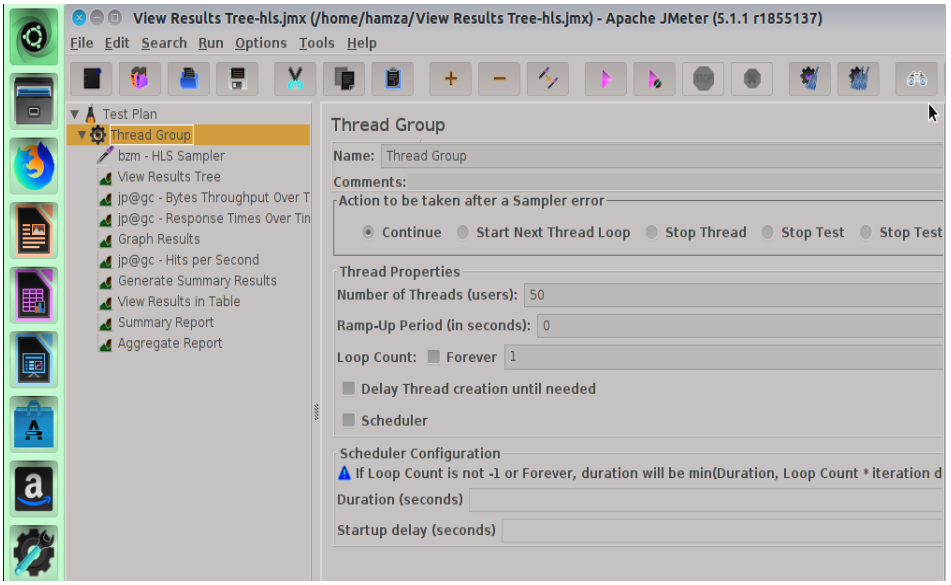

FIGURE 5.2: J.meter simulator showing to increase the number of users

The figure 5.3 shows the J.meter simulation in which, i have performed the test with 50 users simultaneously and it can be seen from the figure that how the segments of HLS protocol retrieved by the simulator, further information about the packet size and its load time with throughput is also found by j.meter.

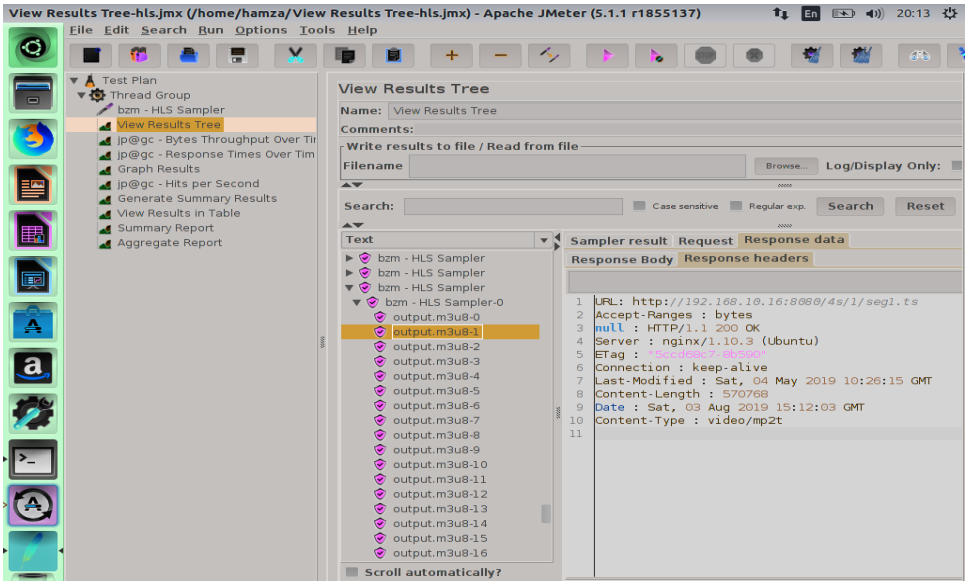

FIGURE 5.3: J.meter simulator showing Fetching of segments

Shown in figure 5.4, this research also generated the result of initial delay with variation of number of users while not using load balancer and caching servers, which in this research referred as CDN or content delivery network. Moreover,

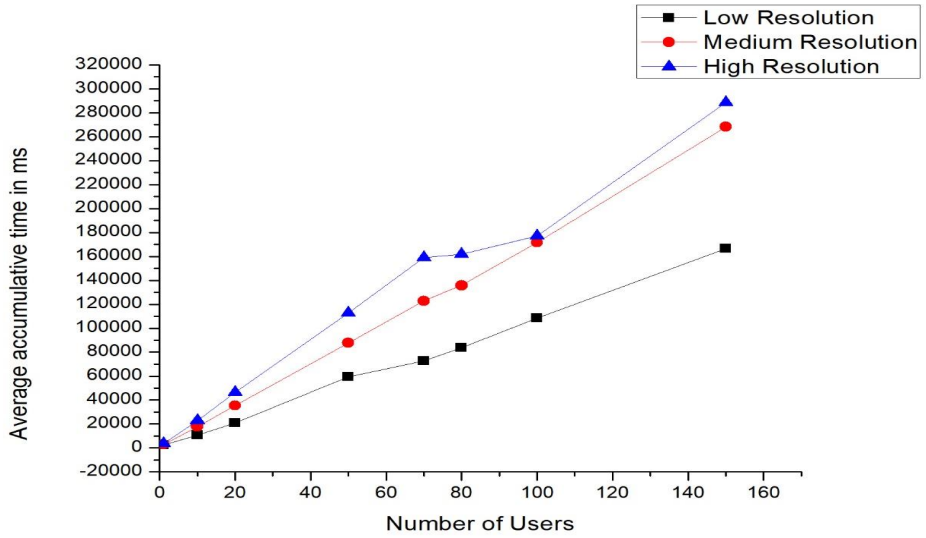

FIGURE 5.4: Response time and accumulative time graph with number of users without CDN

the throughput is also measured and evaluated against number of users by using J.meter, this research have also compare the throughput of the system without using content delivery network with by using content delivery network, it has been observed that with using CDN, the throughput increases and the difference is more interestingly seen in case of less number of users. The comparison can be seen in the figure 5.5:

More tests were run over the architecture for the evaluation of the effect of segment size by fetching the videos of different segment sizes, it is observed that the throughput is much higher for smaller size segments and after the size of 6 seconds it becomes consistent or uniform, this test was run by accommodating 50 users simultaneously. The results can be seen in the figure 5.6:

In the same way, it can be seen that response time and accumulative time for three qualities changes with the variation of segment size until 10-12 seconds it increases with the increase of segment size and then it became much uniform at

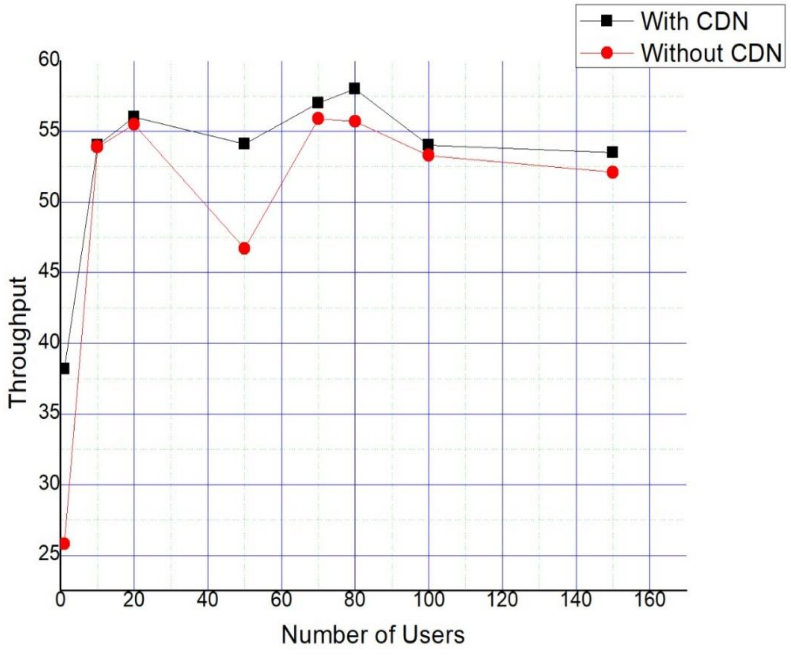

FIGURE 5.5: Throughput with number of users

the duration of 15-20 seconds. But response time is least at the segment seize of 2 seconds. The results of the above discussion are given in figure 5.7:

For encounter the device capability on which the certain video is playing and from which the certain video is requested,RAM of the same virtual machine are changed and try to find the effect on system's throughput and response time, in figure 5.8, the average throughput is much less for the 512 MB RAM and then it increases with sudden change in case of 1Mb then slight change has been observed in figure 5.8 and 5.9.

5.2 Resource Usage Evaluation

Resource usage evaluation is also done upon the proposed architecture; in which I have captured CPU load for 300 seconds for all segment sizes that are of 2s, 4s, 6s, 8s, 10s, 15s and 20 seconds. From which one can observe the behavior of the

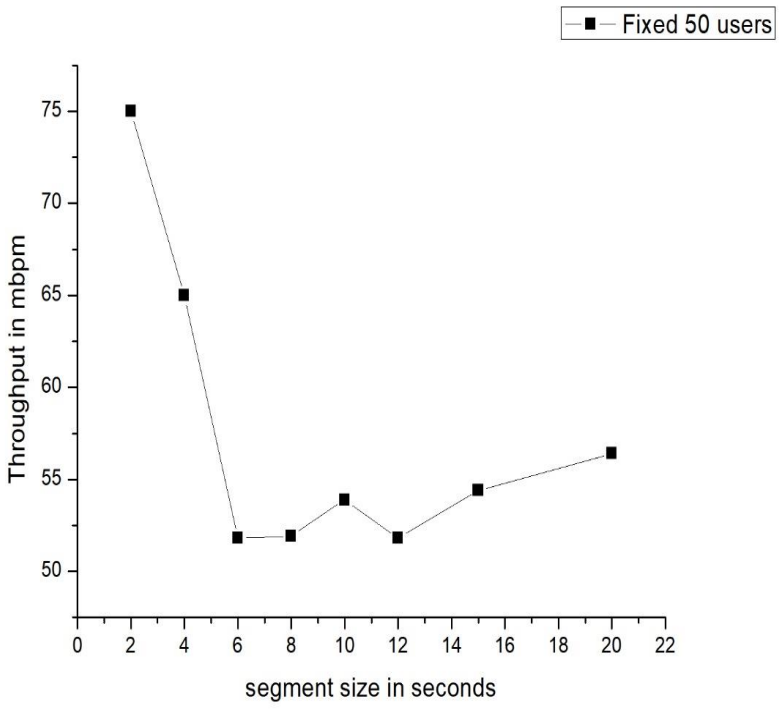

FIGURE 5.6: Throughput with variation of segment size

CPU consumption and Energy consumption. Important things that can be seen in this evaluation are:

1. Segments with short sizes do more CPU consumption then the segments of greater sizes this can be seen in the figure below.

2. In the same way short size segments appear to consume more energy than the segment having greater sizes in seconds.

This type of behavior appears because, while fetching short size segments, client have to make multiple HTTP connection to the main server and then stored into the buffer of the application for smooth playback of the desired video, as many chunks of the videos are, that much connection will be established by the client, lesser the video chunks less will be the connections established by the client or from the client end. This behaviour is depicted in figure 5.10 and 5.11.

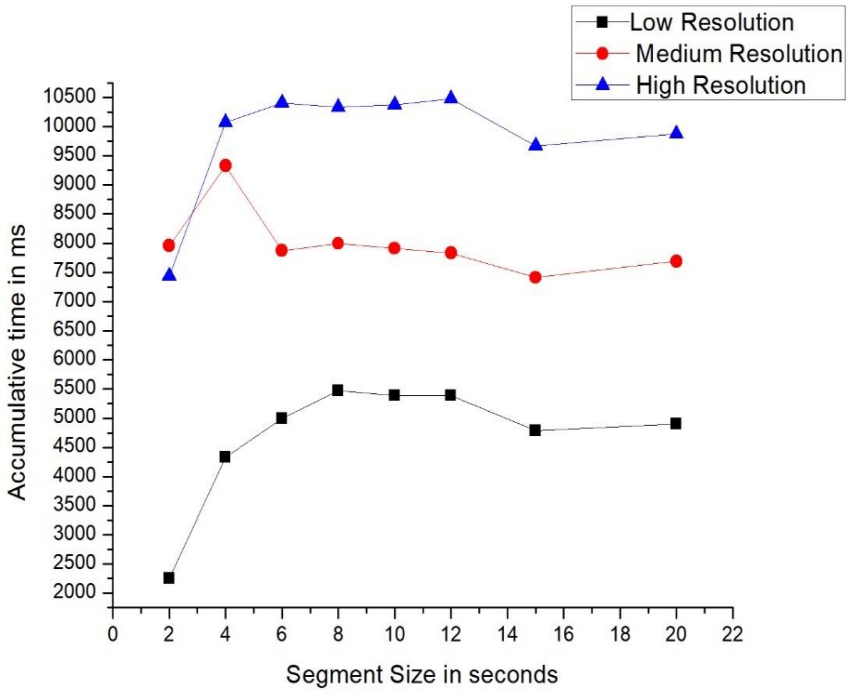

FIGURE 5.7: Throughput with variation of segment size

5.3 Implementation in the IPTV Service

The proposed architecture is also assembled in the current project of ignite, named by "3GPP-IMS compliant E2E Mobile IPTV solution for 4G/LTE Networks", and validated for both live streaming and VOD.

This project is developing up to the mark architecture for IPTV services for the telecom and internet service organizations. Because of using IMS, it can be connected in any running network of any organization. IMS is referred as IP Multimedia Subsystem, in this system a client will register itself through IMS system for became the client of the service and then its billing and charges will automatically charge through IMS based billing and charging system. This services will able to run over 4G and LTE networks and also have capability to become the part of 5G architecture that will enhance its future capability, the architecture including the architecture proposed in this research are using standard ports and tools that can be easily modifiable and flexible in nature. The complete architecture diagram of the IPTV service is shown in figure 5.12: Where the proposed design is also

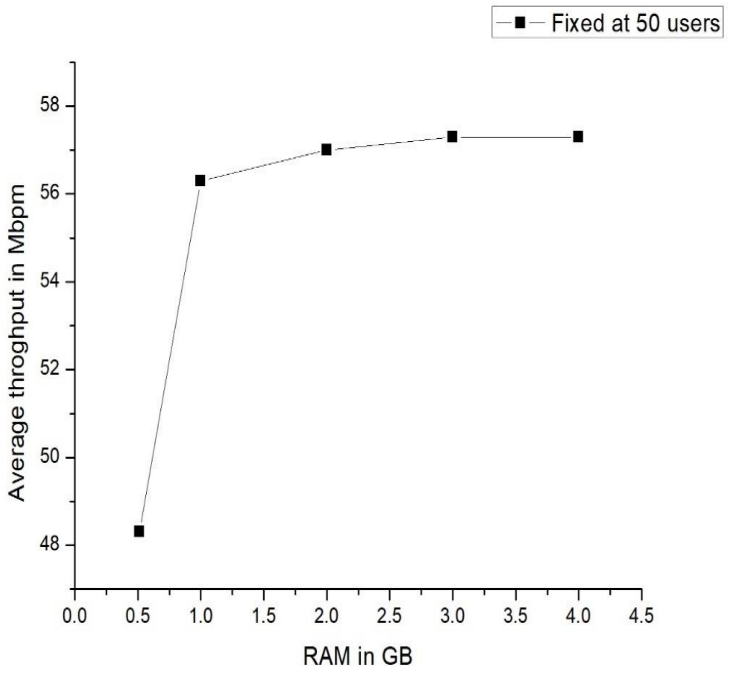

FIGURE 5.8: Effect of RAM on Throughput at the user End

shown, which is integrated in the whole system design, that includes encoding server, HTTP server, Video CDN and client end.

5.4 Testing with Seige

Another test is performed over a slightly changed architecture shown in figure 5.13, in which transcoding server is not included and it is performed to evaluate the performance of content delivery network, it is done by using Nginx benchmarking tool named as Siege, which measures the real time performance of the server by creating load over the desired server. With the help of siege, we measured the performance of our proposed architecture by requesting video file from that server with the server that has only Nginx web-server. Which shows the performance of architecture that this research had proposed that will enhance quality of experience as well as full fill the need of content delivery network for IPTV services by reducing the response time of the desired content.

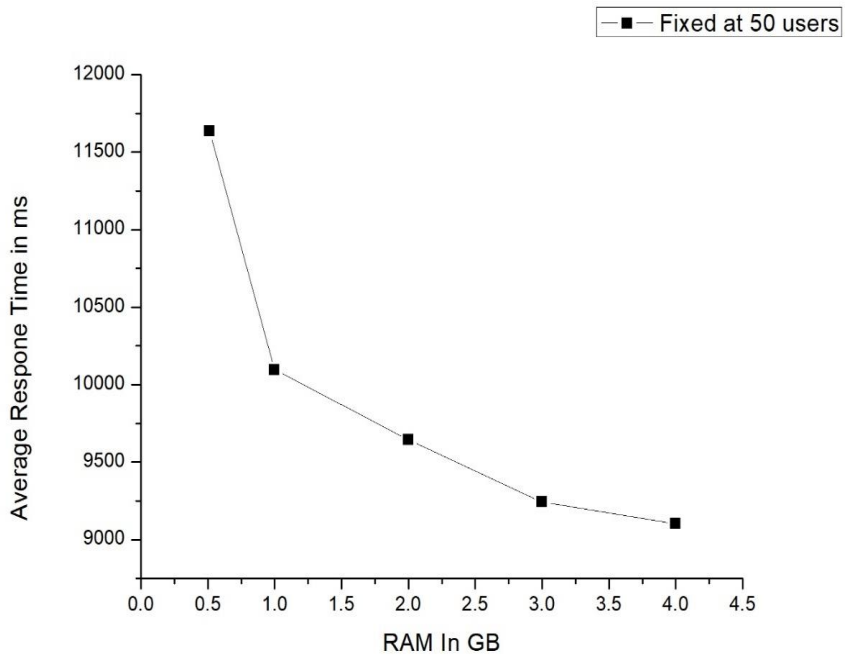

FIGURE 5.9: Effect of RAM on Response Time at the User End

The siege stress test is run over both servers one by one to check the response time of the requested video file, for elapsed time of 10 seconds and changing the number of concurrent user in one time on the server. After that we compared the response time of both our architecture and general Nginx web server through log file and interpret the data in to graphical form shown in figure 5.14.

The simulation results from the siege stress test revealed that the response time is better when the number of concurrent users is less without using proposed CDN solution, but is drastically decreases when the user load increases over the server. While CDN behaves pretty much consistent, as the response time varies between 4 seconds to hardly 6 seconds from less user load to high user load, means for the Video server, which have high traffic load, may able to give much less response while using our architectures. Services like IPTV had much load on server every time, therefore the proposed solution is fair option for such a service. The response time will get minimized by proposed architecture, resulting in improvement of QoE. Under the same scenario, another reading is taken from

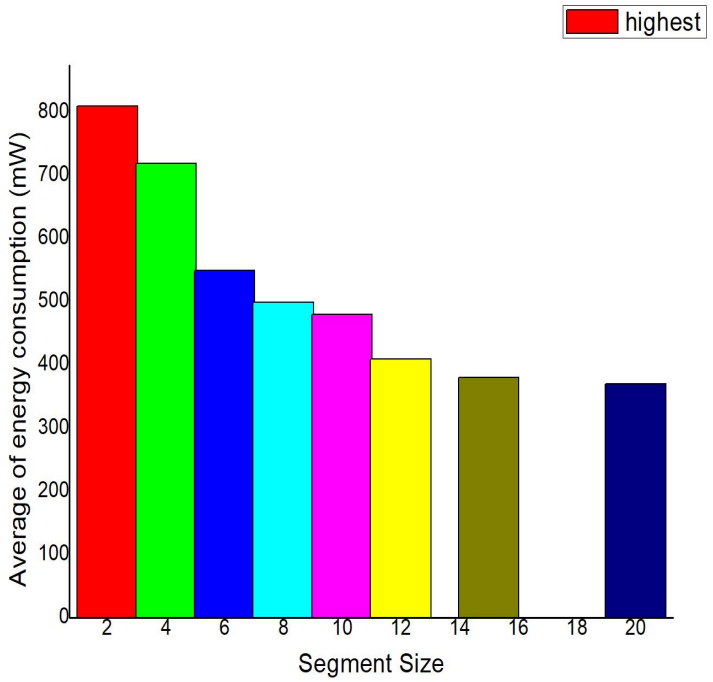

FIGURE 5.10: Effect Segment size over CPU consumption

the Siege Benchmarking log file that is the relation of throughput with number of concurrent users, the results can be analyzed from the figure 5.15.

Almost the same trend is followed in the above simulation results. Throughput of CDN solution is consistent as the number of concurrent user increases, while the throughput of other server is very low. The lowest throughput of other server with load of 1000 user is 0.98 Mbps, while our CDN solution have throughput of 153.08 Mbps. Number of transactions at different traffic load is also important criteria for judging the performance of any server, therefore we have taken results that relate number of transactions with the traffic load. After finding results, we visualized that the transaction rate of the other server is better initially but decreases as the load of user increases on the server, while our CDN solution behaves much expectedly and very little change appears in transaction rate, when load is increased. The results can be seen in the figure 5.16:

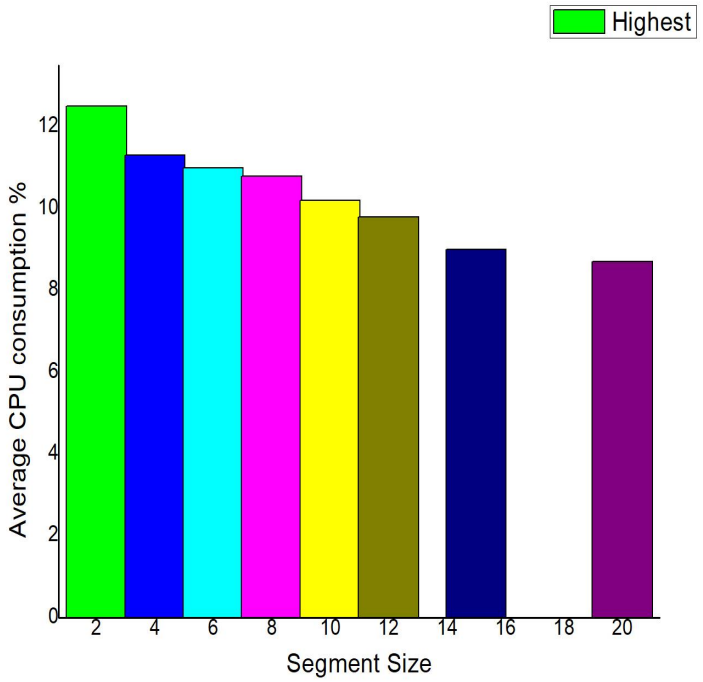

FIGURE 5.11: Effect of segment size over Energy consumption

IPTV Performance Metrics

This section contains the results of the real time measurements from the IPTV service, these measurements constitute of QoE metrics including initial delay, stalling frequency and Quality switching frequency.The measurements are enlisted in the table 5.1. It has been observed that with increase of segment size the initial delay also increases, but the stalling events and Quality frequency reduces.

5.5 Subjective Analysis

Another test is ran over the architecture, this is for subjective analysis, which aims to evaluate and estimates the Quality of Experience to an accurate one. For this purpose, 15 users were selected for the execution of the experiment , who have enough knowledge about the video perception quality. The Test was conducted in

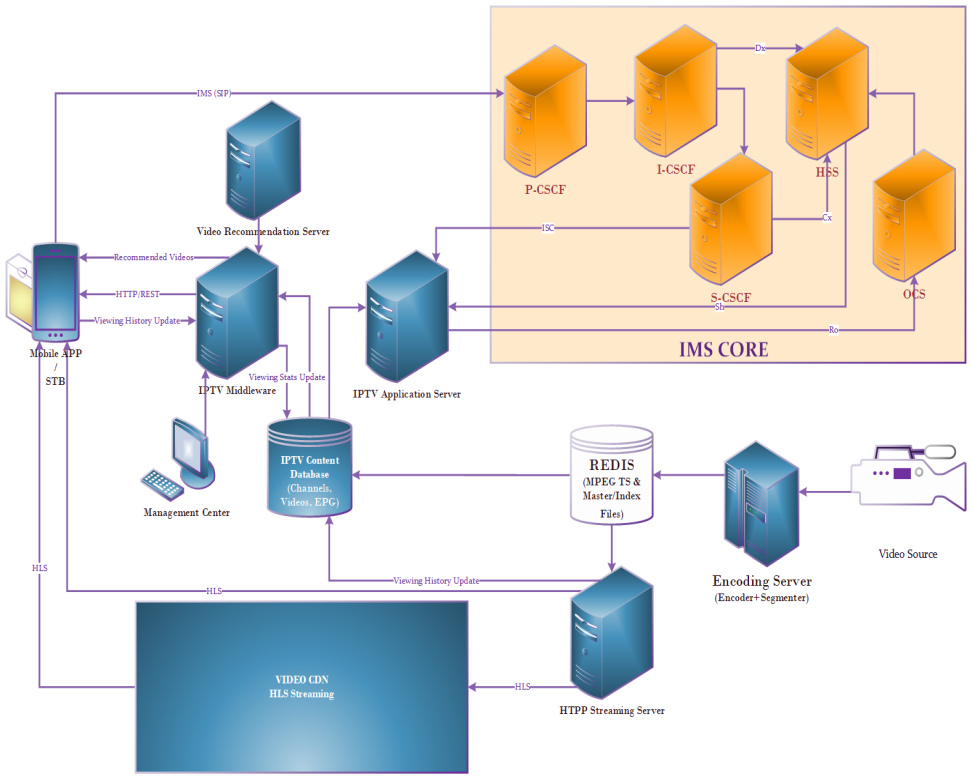

FIGURE 5.12: Complete Architecture diagram of IPTV service

Real time Results			
Segment size	Initial Delay	Number of stalling events	Number of times Quality switches
2s	1 ms	6 times	6 times
4s	1 ms	2 times	4 times
6s	2 ms	4 times	4 times
8s	2 ms	4 times	4 times
10s	4 ms	3 times	2 times
12s	4 ms	2 times	1 times
15s	5 ms	1 times	1 times
20s	5 ms	1 times	1 times

TABLE 5.1: Measurements from IPTV service.

real environment on real server, configured at Mobilevas.co, a software house in Islamabad Pakistan. For the analysis a complete dataset of videos were available consisting of different segment sizes and different qualities, in this session every video is played for 2 minutes and the participants were strictly guided for the rating of the video quality in the proper way, for this a time of 2 minutes were given after the every video session with different segment sizes to give the reviews about the three important parameters, that hurt the QoE and also rate the Quality of

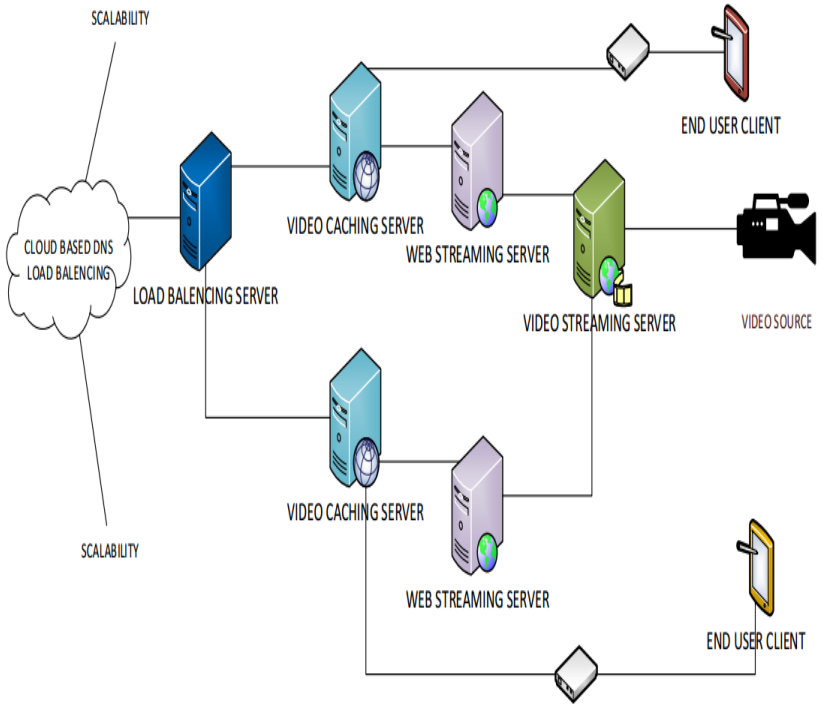

FIGURE 5.13: Content Delivery Network Architecture

MOS Results				
Segment size	Initial Delay	Number of times Quality switches	Number of stalling events	QoE
2s	1 ms	3 times	5 times	3
4s	2 ms	4 times	4 times	2
6s	2 ms	4 times	3 times	3
8s	1 ms	5 times	4 times	3
10s	3 ms	4 times	6 times	3
12s	3 ms	2 times	2 times	4
15s	4 ms	6 times	3 times	2
20s	5 ms	4 times	3 times	2

TABLE 5.2: MOS results.

Experience for every segment size on the scale of 1-5. The scale from 1-2, represents dissatisfaction of the user and the scale of 3-5, means satisfactory level, where 1 is the poorest and 5 is the excellent. The results achieved during subjective analysis is given in the table 5.2:

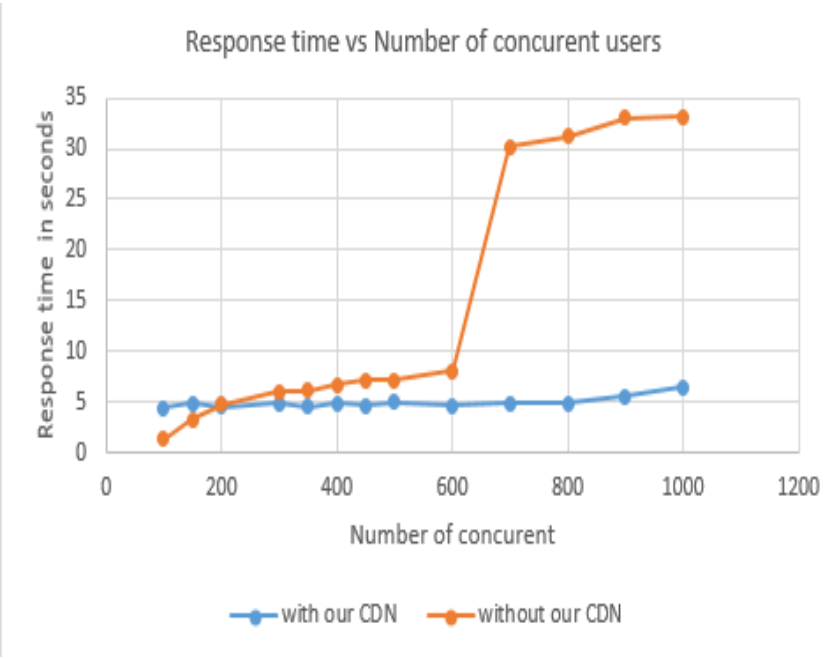

FIGURE 5.14: Results of response time VS Users load on server

After analysis of this experiment it has been seen that, the results of initial delay is much worst in the case of large segment sizes like in the range of 12- 20 seconds. Moreover other important factors has been evaluated in this experiments conclude smooth playing region in segment sizes, in which a region between 6- 12 seconds found to be much smoother than other segments that have comparatively less initial delays also. The bar chart of the reviews of the people is given in figure 5.17.

Another important finding came into attention that, the Mean Opinion score given by the reviewers or the subjective analysis and the influence factors, initial delay, segment sizes and stalling frequency (objective analysis) both have the inverse relationship with each other, as the influence factors number increases the Quality of Experience at the user end decreases exponentially, the trajectory is also given in figure 5.18.

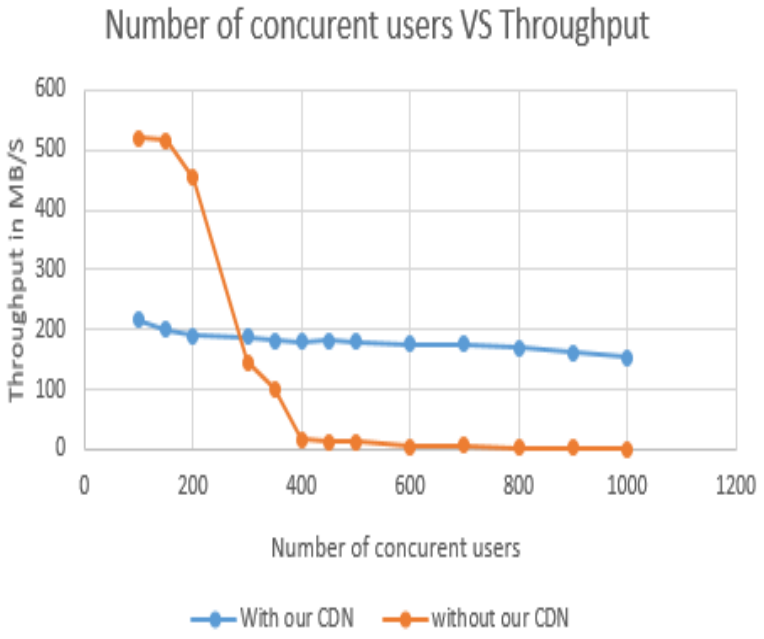

FIGURE 5.15: Results of Throughput Vs Users load on server

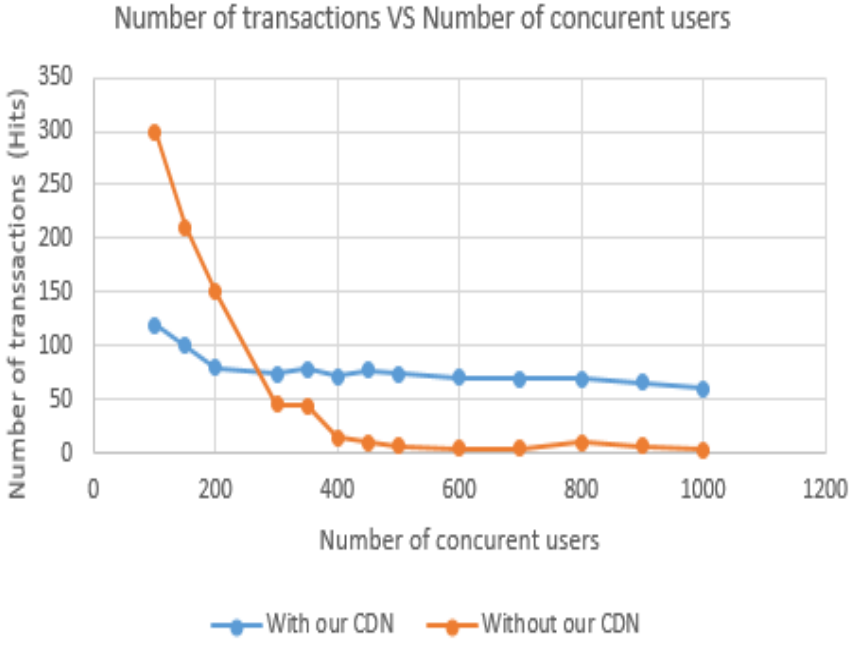

FIGURE 5.16: Results of Number of transactions VS Users load on server

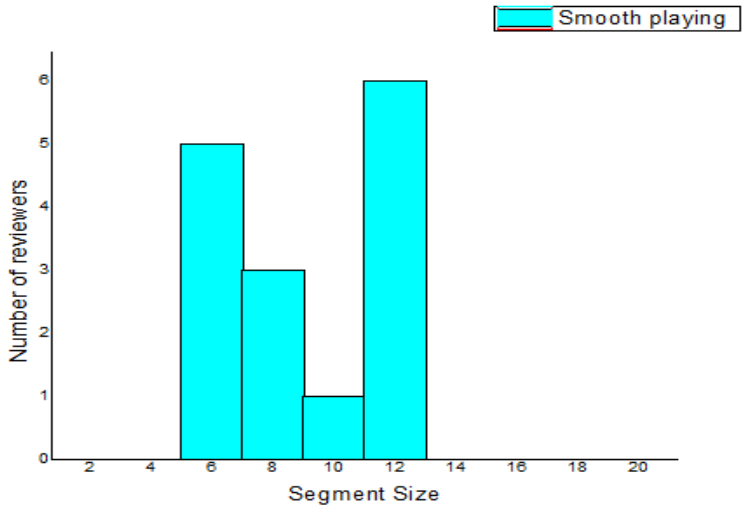

FIGURE 5.17: Region of smooth playing in segment sizes

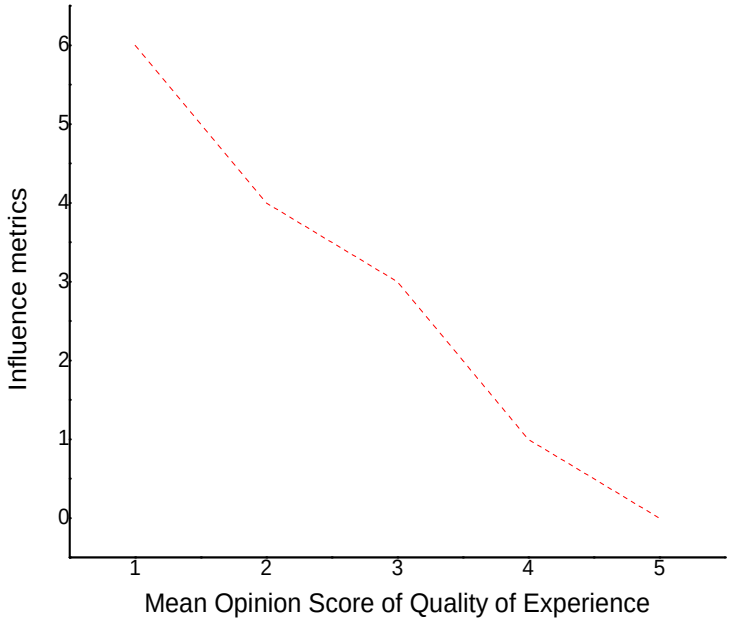

FIGURE 5.18: Relationship of influence factors with subjective analysis

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This section includes the conclusion and future work. In this research, adaptive bit-rate video delivery network has been tested to enhance the response time and throughput by varying different factors influencing it. Moreover key parameter fluctuations are also measured and analyzed by varying the segment size of the same video with multiple representations. The factors including accumulative time, stalling frequency and quality switching representation is also compared with quality of experience and inverse relationship has been observed between all the factors and the Quality of Experience from the subjective users. Furthermore, it has been observed that load balancing and caching servers have positive effect on response time, throughput and number of transactions through Seige load testing tool by increasing the number of concurrent users from 1 to 200 users at a time . The whole testbed is deployed on real machines and the real time measurements were taken through J.meter, which evaluates the performance of proposed architecture in terms of response time and throughput.

This research presents the fact that segment size is an important factor, which contributes effectively on the QoE at the end users. By analyzing throughput and response time by the users, a segment length of 6 sec - 12 sec is proposed

for the smooth streaming of the video. The energy and CPU consumption is also evaluated and observed that smaller segments result in more energy and CPU consumptions rather longer chunks as multiple connections were generated by the end client application or server to fetch the smaller segments multiple times.

6.2 Future Work

This architecture design may be able to expand by including intelligent system, for network monitoring and transcoding of videos in real time, a centralized monitoring system may be included for intelligent route optimization of the users to the concerned caching server. Moreover, by taking all the real time statistics the server can able to take decision of the encoding of the videos, to which time a specific coding schemes will optimize the QoE at the user end. In addition, for the QoE analysis, a prediction model may be implemented by applying machine learning algorithms like neural networks, that enables the prediction of the required QoE as well as current user's QoE and then comparison of the subjective and objective QoE may be made.

The proposed architecture will be implemented, to perform adaptive streaming by including monitoring points at the server side, network side, and the client side and then dynamic exchange the PQoS Measurements from STB and Head End to a centralized QoS based video streams transmission control system by introducing a policy controller for centralized decisions based on the network monitoring information. User pulls the service from server while Network QoE Monitor gets the User Device stats, Usage Stats, Network QoS and Application/Service Stats to build Video Profiles of every user. This Network QoE Monitor directs the Encoding Server on the basis of QoE Score Model. It also performs QoE Reporting. This mechanism shall implement a network monitoring mechanism both at server side and client side to monitor the status of channel, current bandwidth, the capability of user device such as computing power and screen size, available bandwidth to

transmit adaptive video streams by adjusting the video frame size, video quality and by changing the compression technique or parameters.

Bibliography

[1] O. Oyman and S. Singh., "Quality of experience for http adaptive streaming services," *IEEE Comm Magazine*, vol. 50, no. 4, pp. 20–27, 2012.

[2] S. I, "The mpeg-dash standard for multimedia streaming over the internet." *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.

[3] H. F. Adhikari VK, Guo Y and et al, "Measurement study of netflix, hulu, and a tale of three cdns." *IEEE ACM Transactions*, vol. 23, no. 6, pp. 1984–1997, 2015.

[4] Z. T. H. T. Seufert M, Egger S Slanina M and T. P, "A survey on quality of experience of http adaptive streaming," *IEEE Comm Surveys Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[5] T. Stockhammer, "Dynamic adaptive streaming over http: standards and design principles." *" In Proceedings of the second annual ACM conference on Multimedia systems, San Jose, CA, USA*, vol. 2, pp. 133–144, 2011.

[6] V. H. M. S. Vijay Kumar Adhikari Yang Guo Fang Hao, Matteo Varvello and Z. Zhang., "understanding and improving multi-cdn movie delivery," *In INFOCOM*, vol. 25, no. 30, pp. 1620–1628, 2012.

[7] R. Pantos and W. May, "Http live streaming," 2017.

[8] M. Levkov, "Video encoding and transcoding recommendations for http dynamic streaming on the adobe® flash® platform," *White Paper, Adobe Systems Inc*, 2010.

[9] A. Zambelli, "Iis smooth streaming technical overview," *Microsoft Corporation*, vol. 3, p. 40, 2009.

[10] L. García, J. Lloret, C. Turro, and M. Taha, "Qoe assesment of mpeg-dash in polimedia e-learning system," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2016, pp. 1117–1123.

[11] J. Lloret, M. Garcia, M. Atenas, and A. Canovas, "A qoe management system to improve the iptv network," *International Journal of Communication Systems*, vol. 24, no. 1, pp. 118–138, 2011.

[12] M. Garcia, A. Canovas, M. Edo, and J. Lloret, "A qoe management system for ubiquitous iptv devices," in *2009 Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*. IEEE, 2009, pp. 147–152.

[13] C. V. N. Index, "Forecast and methodology, 2014-2019 white paper," *Retrieved 23rd September*, 2015.

[14] S. Adler, "The slashdot effect: An analysis of three internet publications (1999)," *Disponıvel em:¡ http://ssadler. phy. bnl. gov/adler/SDE/SlashDotEffect. html¿. Acesso em*, vol. 17, 2001.

[15] N. Bartolini, E. Casalicchio, and S. Tucci, "A walk through content delivery networks," in *International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. Springer, 2003, pp. 1–25.

[16] A.-M. K. Pathan and R. Buyya, "A taxonomy and survey of content delivery networks," *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, vol. 4, p. 70, 2007.

[17] D. C. Verma, "Content distribution networks," *A Wiley-Interscience Publication*, 2002.

[18] M. Aguilar, "Aprendizaje y tecnologías de información y comunicación: Hacia nuevos escenarios educativos," *Revista Latinoamericana de Ciencias Sociales, Niñez y Juventud*, vol. 10, no. 2, pp. 801–811, 2012.

[19] D. Zhang, L. Zhou, R. O. Briggs, and J. F. Nunamaker Jr, "Instructional video in e-learning: Assessing the impact of interactive video on learning effectiveness," *Information & management*, vol. 43, no. 1, pp. 15–27, 2006.

[20] F. A. Urbano, G. Chanchí, W. Y. Campo, H. F. Bermúdez Orozco, and E. Astaiza Hoyos, "Entorno de pruebas para el soporte de videostreaming usando herramientas libres," *Revista Científica Ingeniería y Desarrollo*, vol. 34, no. 2, pp. 333–353, 2016.

[21] P. L. Agudelo, W. Y. Campo, A. Ruíz, J. L. Arciniegas, and W. J. Giraldo, "Architectonic proposal for the video streaming service deployment within the educational context," in *Colombian Conference on Computing*. Springer, 2017, pp. 313–326.

[22] J. P. Mulerikkal and I. Khalil, "An architecture for distributed content delivery network," in *2007 15th IEEE International Conference on Networks*. IEEE, 2007, pp. 359–364.

[23] A. Alfa, F. Ogwueleka, E. Dogo, and M. Sanjay, "A hybrid web caching design model for internet-content delivery," *Covenant Journal of Informatics and Communication Technology*, vol. 4, no. 2, 2016.

[24] J. Apostolopoulos, T. Wong, W.-t. Tan, and S. Wee, "On multiple description streaming with content delivery networks," in *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3. IEEE, 2002, pp. 1736–1745.

[25] S. Wee, J. Apostolopoulos, W.-t. Tan, and S. Roy, "Research and design of a mobile streaming media content delivery network," in *2003 International Conference on Multimedia and Expo. ICME'03. Proceedings (Cat. No. 03TH8698)*, vol. 1. IEEE, 2003, pp. I–5.

[26] X. Wang, T. Kwon, Y. Choi, H. Wang, and J. Liu, "Cloud-assisted adaptive video streaming and social-aware video prefetching for mobile users," *IEEE wireless communications*, vol. 20, no. 3, pp. 72–79, 2013.

[27] M. Taha, J. Lloret, A. Ali, and L. Garcia, "Adaptive video streaming testbed design for performance study and assessment of qoe," *International Journal of Communication Systems*, vol. 31, no. 9, p. e3551, 2018.

[28] A. Zabrovskiy, E. Kuzmin, E. Petrov, and M. Fomichev, "Emulation of dynamic adaptive streaming over http with mininet," in *2016 18th Conference of Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT)*. IEEE, 2016, pp. 391–396.

[29] Y. M. Hassan, A. Helmy, and M. M. Rehan, "Effect of varying segment size on dash streaming quality for mobile user," in *2014 International Conference on Engineering and Technology (ICET)*. IEEE, 2014, pp. 1–4.

[30] Y. Liu, S. Dey, D. Gillies, F. Ulupinar, and M. Luby, "User experience modeling for dash video," in *2013 20th International Packet Video Workshop*. IEEE, 2013, pp. 1–8.

[31] J. Hwang, J. Lee, and C. Yoo, "Eliminating bandwidth estimation from adaptive video streaming in wireless networks," *Signal Processing: Image Communication*, vol. 47, pp. 242–251, 2016.

[32] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li, "Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky," in *Proceedings of the 17th ACM international conference on Multimedia*. ACM, 2009, pp. 25–34.

[33] M. Y. Fuller, S. Mukhopadhyay, and J. M. Gardner, "Using the periscope live video-streaming application for global pathology education: a brief introduction," *Archives of pathology & laboratory medicine*, vol. 140, no. 11, pp. 1273–1280, 2016.

[34] T. Hartsell and S. C.-Y. Yuen, "Video streaming in online learning," *AACE Journal*, vol. 14, no. 1, pp. 31–43, 2006.

[35] K. Sigama and B. M. Kalema, "Conceptualizing moocs implementation for higher education in developing countries," in *2018 IEEE 6th International Conference on MOOCs, Innovation and Technology in Education (MITE)*. IEEE, 2018, pp. 14–18.

[36] L. De Cicco and S. Mascolo, "An adaptive video streaming control system: Modeling, validation, and performance evaluation," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 2, pp. 526–539, 2014.

[37] A. B. Johnston and D. C. Burnett, *WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web*. Digital Codex LLC, 2012.

[38] T. Stockhammer, "Dynamic adaptive streaming over http–: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 133–144.

[39] M. Elkotob, "Architectural, service, and performance modeling for an ims-mbms-based application," in *2010 IEEE International Conference on Communications*. IEEE, 2010, pp. 1–7.

[40] G. Fortino, C. E. Palau, W. Russo, and M. Esteve, "The comodin system: A cdn-based platform for cooperative media on-demand on the internet," in *Proceedings of the 10th International Conference on Distributed Multimedia Systems (DMS'04)*, 2004.

[41] M. Li and C.-H. Wu, "A cost-effective resource allocation and management scheme for content networks supporting iptv services," *Computer Communications*, vol. 33, no. 1, pp. 83–91, 2010.

[42] M. Taha, "A novel cdn testbed for fast deploying http adaptive video streaming," in *Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications*. ICST (Institute for Computer Sciences, Social-Informatics and ..., 2016, pp. 65–71.

[43] M. Garcia-Pineda, S. Felici-Castell, and J. Segura-Garcia, "Using factor analysis techniques to find out objective video quality metrics for live video streaming over cloud mobile media services," *Network Protocols and Algorithms*, vol. 8, no. 1, pp. 126–147, 2016.

[44] J. M. Jimenez, J. O. Romero Martínez, A. REGO MAÑEZ, and J. Lloret, "Analyzing the performance of software defined networks vs real networks," *International Journal On Advances in Networks and Services*, vol. 9, no. 3-4, pp. 107–116, 2016.

[45] A. Zambelli, "Iis smooth streaming technical overview. microsoft corporation, 2009."

[46] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," *IEEE/ACM Transactions on Networking (ToN)*, vol. 22, no. 1, pp. 326–340, 2014.

[47] A. Zabrovskiy, C. Feldmann, and C. Timmerer, "Multi-codec dash dataset," in *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM, 2018, pp. 438–443.