Master Thesis
Electrical Engineering
October 2015

Performance analysis of transmission protocols for H.265 encoder

AKELLA UMESH

Department of Communication Systems
Blekinge Institute of Technology
SE-371 79 Karlskrona
Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Masters in Electrical Engineering with emphasis on telecommunication systems. The thesis is equivalent to 20 weeks of full time studies.

This master thesis is typeset using LaTeX.

Contact Information:
Author(s):
Akella Umesh

E-mail:
umak14@student.bth.se,
umesh.akella@gmail.com

University advisor(s):
Dr. Patrik Arlos
Department of Communication Systems

University examiner(s):
Dr. Kurt Tutschku
Department of Communication Systems

School of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona
Sweden

Internet : www.bth.se/com
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

In recent years there has been a predominant increase in multimedia services such as live streaming, Video on Demand (VoD), video conferencing, videos for the learning. Streaming of high quality videos has become a challenge for service providers to enhance the user's watching experience. The service providers cannot guarantee the perceived quality. In order to enhance the user's expectations, it is also important to estimate the quality of video perceived by the user. There are different video streaming protocols that are used to stream from server to client. In this research, we aren't focused on the user's experience. We are mainly focused on the performance behavior of the protocols.

In this study, we investigate the performance of the HTTP, RTSP and WebRTC protocols when streaming is carried out for H.265 encoder. The study addresses for the objective assessment of different protocols over VoD streaming at the network and application layers. Packet loss and delay variations are altered at the network layer using network emulator NetEm when streaming from server to client. The metrics at the network layer and application layer are collected and analyzed. The video is streamed from server to a client, the quality of the video is checked by some of the users.

The research method has been carried out using an experimental testbed. The metrics such as packet counts at network layer and stream bitrate at application layer are collected for HTTP, RTSP and WebRTC protocols. Variable delays and packet losses are injected into the network to emulate real world.

Based on the results obtained, it was found at the application layer that, out of the three protocols, HTTP, RTSP and WebRTC, the stream bitrate of the video transmitted using HTTP was less when compared to the other. Hence, HTTP performs better in the application layer. At the network layer, the packet counts of the video transmitted were collected using TCP port for HTTP and UDP port for RTSP and WebRTC protocols. The performance of HTTP was found to be stable in most of the scenarios. On comparing RTSP and WebRTC, the number of packet counts collected were more in number for RTSP when compared to WebRTC. This is because, the protocol and also the streamer are using more resources to transmit the video. Hence, both the protocols RTSP and WebRTC are performing better relatively.

Keywords: H.265, HTTP, Network Emulator, RTSP, VoD, WebRTC

Acknowledgments

Firstly, I would like to thank the Almighty God for blessing me with knowledge, strength and vision. I would also like to thank my beloved parents and other family members for unconditional love and support without which I would not have been able to achieve anything.

I express my deepest gratitude to my supervisor Dr. Patrik Arlos for help, encouragement and support offered throughout the thesis with valuable suggestions. I have learnt a lot in the entire process from various meetings and crucial discussions with him, which will also help me in my future works and career. He also helped me in providing feedback during the meeting which is the most important part for successful completion of my master thesis, which also helped me in following the right track. It gives me immense pleasure to thank him and express my heartfelt appreciation for the support and guidance.

Last but not the least, I would like to thank all my classmates, well-wishers at JNTU, Hyderabad and BTH, Karlskrona and my cheerful group of friends (Raywon, Tulasi Priyanka, Navya, Sankaran) for their encouragement and full support towards me.

–Akella Umesh

Contents

Abstract i

Acknowledgments ii

List of Contents iii

List of Figures iv

List of Tables v

Acronyms vi

1 Introduction 2
- 1.1 Aims and Objectives . . . 3
- 1.2 Scope of the Thesis . . . 4
- 1.3 Research Questions . . . 4
- 1.4 Expected Outcomes . . . 5
- 1.5 Research Methodology . . . 5
- 1.6 Outline of the Thesis . . . 6

2 Background and Related Work 8
- 2.1 Related Work . . . 8
- 2.2 Background . . . 10
  - 2.2.1 Video Streaming . . . 10
  - 2.2.2 Video Compression . . . 10
  - 2.2.3 Video Codecs . . . 10
  - 2.2.4 VLC Media Player and its functions . . . 11
  - 2.2.5 H.265/HEVC Encoder . . . 14
  - 2.2.6 Video Transmission . . . 14
  - 2.2.7 Transmission Control Protocol (TCP) . . . 15
  - 2.2.8 User Datagram Protocol (UDP) . . . 15

3 Methodology 17
- 3.1 Experimental Setup . . . 17
- 3.2 Setup Design . . . 17

3.2.1 Measurement Point 19
3.2.2 MArC 19
3.2.3 Consumer 19
3.2.4 VLC Media Server 19
3.2.5 VLC Media Client 19
3.2.6 NetEm 20
3.2.7 Packet Loss 20
3.2.8 Delay 20
3.2.9 Bitrate 20
3.2.10 Packet Counts 21
3.3 Experimetal Procedure 21

4 Results and Analysis **24**
4.1 Mean Value calculations 24
4.2 Application Layer 26
4.3 Network Layer 27

5 Conclusion and Future Work **31**

References **34**

Appendix **38**
A APPENDIX A 38
B APPENDIX B 39

List of Figures

2.1 Video Codec 11

3.1 Experimental Setup 18

List of Tables

3.1 Hardware Specifications 17

4.1 Demux bitrate values for variable loss ratios 26
4.2 Demux bitrate values for variable delays. 27
4.3 Packet count values for variable loss ratios 28
4.4 Packet count values for variable delays 28

B.1 Demux bytes read values for variable loss ratios 39
B.2 Demux bytes read values for variable delays 39
B.3 Input bitrate values for variable loss ratios 40
B.4 Input bitrate values for variable delays 41
B.5 Input bytes read values for variable loss ratios 41
B.6 Input bytes read values for variable delays 42

Acronyms

API	Application Program Interface
AVC	Advanced Video Coding
CB	Coding Block
CDDA	Compact Disc Digital Audio
CTB	Coding Tree Block
CTU	Coding Tree Unit
DAG	Data Acquisition and Generation
DPMI	Distributive Passive Measurement Infrastructure
DVB	Digital Video Broadcasting
GNU	General Public License
GUI	Graphical User Interface
HEVC	High Efficiency Video Coding
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
LTS	Long-term Support
MArC	Measurement Area Controller
MP	Measurement Point
MPEG	Motion Picture Expert Group
NAT	Network Address Translation
QoE	Quality of Experience
QoS	Quality of Service
RD	Rate Distortion
RIP	Routing Information Protocol
RTMP	Real Time Messaging Protocol
RTP	Real Time Transport Protocol
RTSP	Real Time Streaming Protocol
TC	Traffic Control
TCP	Transmission Control Protocol
TS	Transport Streams
UDP	User Datagram Protocol
VLC	Video LAN Client
VoD	Video on Demand
WebRTC	Web Real-Time Communication Protocol

Chapter 1
INTRODUCTION

Chapter 1

Introduction

In recent years, there has been an increase in the number of video streaming services. Streaming high quality videos has become a challenge for the service providers. There are several protocols available for streaming the video. The service providers are mainly focused on the perceived quality of video for the users. There has been a huge development of applications for the minimization of the errors to enhance the user's experience. Choosing an appropriate streaming protocol is a challenging task for the service providers. The performance of the protocols depends on the network parameters such as packet loss, delay, available capacity and jitter. This indicates that people can view the same content at the same time over the internet.

Video on Demand (VoD) is defined as a service in which high-speed of data services are used to provide customers with the streaming content[13]. The data which is sent to the users travel across the network, is received and played on user's device and is then discarded. Some of the applications of VoD include Media entertainment distribution on demand and Media entertainment broadcast distribution.

There are different protocols on the Internet that are used for streaming the video. There has been an enormous development and research in the field of different video streaming protocols like HTTP (Hyper Text Transfer Protocol), RTSP (Real Time Streaming Protocol), RTP (Real Time Transport Protocol), RTMP (Real Time Messaging Protocol) and WebRTC (Web Real-Time Communication Protocol).

HTTP (Hyper Text Transfer Protocol) has grown a lot in terms of its usage for live streaming, whereas RTSP (Real Time Streaming Protocol) has been familiar in the market for a long time and is also a good alternative for streaming videos. HTTP has countless implementations on both server and client, whereas RTSP is used for controlling media session between end points. HTTP protocol doesn't require additional special proxies or caches. It is a stateless protocol and behaves as a system without a feedback control for multimedia transmission. When an HTTP client requests the data, the server responds to the request by delivering the demanded data. Each HTTP request is handled completely independently. HTTP media streaming is the cheapest and the easiest to deploy

as it does not require specialized servers at network nodes. It is relatively easy to traverse middle boxes such as firewalls and NAT (Network Address Translation) devices, keeping up the essential information on the server side, which makes HTTP servers more scalable than conventional streaming servers.

RTSP is an application level protocol for control over the delivery of real time data to address the needs for efficient delivery of the data over the Internet. It is specifically designed to minimize the overhead of multimedia transmission over IP network[20][21]. This protocol is intended to control multiple data sessions and provide means for choosing different delivery channels such as UDP, multicast UDP and TCP[3]. RTSP works on top of a well-established Real time Transport Protocol (RTP), RTSP establishes a TCP connection between the server and the client. When a client sends out controlling requests such as PLAY, PAUSE, etc., to the server, it achieves real time control of playback of the media files from the servers. RTSP is a stateful protocol and it tracks the data and records the media transitions[2][5].

WebRTC (Web Real-Time Communication Protocol) is an API, which is developed by the World Wide Web Consortium (W3C) that supports different applications like browser to browser applications for sharing the files, video calling and transmission of video. It also provides high security for the end users. This protocol has been designed for adapting dynamic link characteristics and to deliver the best possible video quality in any scenario[4][25]. The WebRTC API is currently being used by Mozilla Firefox and Google Chrome and it contains the following functions like RTCPeerConnection, Media Stream and RTCDataChannel. WebRTC protocol can be streamed by sending up using RTP Protocol. The Real Time Transport Protocol (RTP) provides a framework for delivering of audio and video signals and other time media applications.

The High Efficiency Video Coding (H.265/HEVC) is a newly standardized encoder that saves up to 50% of the bandwidth when compared to Advanced Video Coding (H.264/AVC) when streaming the video content over wireless networks [10][14][19]. The main improvement made in this encoder is that it can support high-resolution video.

Metrics like delay, bandwidth and packet loss, etc. are important for measuring the performance of the video streaming. Among these, packet loss and packet delay variation are the important factors that are used for calculating the performance of the video stream as it may result in degradation of the quality of video[15][16][17][18].

1.1 Aims and Objectives

The aim of the thesis is to evaluate the performance of HTTP, RTSP and WebRTC protocols for H.265 encoder. The performance metrics at application

and network layers will be analyzed for different protocols and will be compared among considered protocols in the thesis. The results will be documented in our thesis report. The objectives that are implemented to achieve the goal are mentioned below:

- Perform a literature review for various protocols and also different methods that are implemented by the researchers using HTTP, RTSP and WebRTC protocols.
- Perform a thorough study on HTTP, RTSP and WebRTC video streaming protocols and H.265 encoder to identify relevant features to the experiment.
- Identify the performance metrics to perform the experiment.
- Compare the metrics of different streaming protocols used and specify which protocol to be used under specific scenarios.
- Giving recommendations on the performance analysis of the protocols based on the results obtained.

1.2 Scope of the Thesis

This thesis work describes about the performance of different video streaming protocols for H.265 encoder. HTTP, RTSP and WebRTC protocols have been taken and to evaluate the performance of these protocols, metrics like packet counts at network layer and demux bitrate at application layer have been considered. The performance of these protocols have been evaluated, when variable delays and packet losses are injected into the network.

An experiment test bed was setup to perform the research work. VLC Media Player being a popular open source media player was used to evaluate the performance of different protocols. Various tools such as NetEm (Network Emulator), MP (Measurement Point), and DPMI (Distributed Passive Measurement Infrastructure) were used to facilitate the research. VLC Media Player and H.265 encoder were configured on the Linux Operating System to implement the above mentioned approach. Moreover, the results obtained from the above approaches were analyzed and compared.

1.3 Research Questions

1. What is the packet count measured at the network layer when HTTP, RTSP and WebRTC protocols were used to stream video over the network?

2. What are the streaming statistics measured at the application layer when HTTP, RTSP and WebRTC protocols were used to stream video over the network?

3. How does the performance of the above two cases (R1 and R2) vary when subjected to variable delays and packet loss in the network?

1.4 Expected Outcomes

- Various methods and schemes were proposed by the researchers to support these protocols which would be studied and documented.
- The laboratory test bed will be setup and experiments will be conducted to calculate the performance of the protocols.
- Performance data for different protocols will be collected and analyzed.
- By analyzing the obtained results for different protocols, recommendations can be given.
- The results are presented in the form of tables containing the statistical analysis performed by the author.

1.5 Research Methodology

1. In the early stage of our research, a detailed literature review is studied on the streaming protocols such as HTTP, RTSP and WebRTC and then a detailed study on the various schemes proposed by the researchers on the specified protocol.

2. Study and analyze the impact of different video streaming servers like VLC, Wowza, Red5 and Adobe flash media for live video streaming. Select the suitable video streaming server for streaming the video.

3. In the next stage we choose the videos recommended by standard groups that are encoded to H.265 for conducting experiments using HTTP, RTSP and WebRTC protocols.

4. The performance metrics that are necessary to perform the experiments are to be identified.

5. Once the literature review is done, Experiments are conducted with the appropriate video streaming servers for HTTP, RTSP and WebRTC using standard videos.

6. During the experimentations delays and packet loss are introduced by the appropriate traffic shaper in the network.
7. The streaming of the videos are conducted over HTTP, RTSP and WebRTC protocols. Streaming process is done from server to the client in a controlled environment.
8. When stream is received at the client side, the values are noted at both application level and network level for different protocols.
9. The performance of the HTTP, RTSP and WebRTC protocols are observed and analyzed.
10. Based on the final results and analysis, conclusions and recommendations are provided between the protocols.

1.6 Outline of the Thesis

This report is organized as follows. Chapter 2 describes about background and research work related to this thesis. Chapter 3 contains methodology and it describes about the experimental setup and procedure. Chapter 4 contains the results and analysis and Chapter 5 contains the conclusion and the future work.

Chapter 2
BACKGROUND AND RELATED WORK

Chapter 2

Background and Related Work

2.1 Related Work

Authors of [2] compare three different protocols like HTTP, RTSP and DASH on a smartphone. They have calculated switch delay, which is the time interval between the user sending a switch command and the client screen begins to show frames of new perspective at the normal video quality, switch rendering, which is described as how the screen acts after the client stops showing the old perspective and before it shows the new perspective and limited the bandwidth. They only used H.264 encoder and implemented in the media player of the smartphone. The authors have concluded that DASH based gave best performance as it provided seamless switching with a delay of at most 2 seconds while only incurring 10% overhead. Authors of [1] have conducted an experiment using network simulator on HTTP based video transmission and analysis on how the network impairment influence the streamed video. Authors validated the experiment against a network emulator supplied with real network traces. In conclusion, the authors stated that the buffering strategies implemented by a video player are in many cases able to mitigate unfavourable network conditions that allows the streamed video to play smoothly. Authors of [5] have presented a QoE instrumentation for video streaming, VLQoE on a smartphone. They have added a functionality to the VLC media player to record a set of metrics from the user interface, application-level, network-level and from the available sensors of the device. A tool has been used to present a two state model based on the inter-picture time for the HTTP and RTSP based video streaming via 3.5G. A user based study has been done on the influence of inter-picture time on the user's perceived quality. In conclusion, the authors have concluded that the inter picture times varies from user to user (max 2880ms, min 40ms). The authors have also stated that the maximum inter picture time increases as the user rating decreases and the highest mean of maximum inter-picture times have matched with the "Freeze" indications. They have also analysed the overall user response time of the subjects in two scenarios such as short and long freezes. In conclusion, they have found that the user response time for long freezes was exponentially distributed. In paper [11], the authors have explained the technical

and non-technical parameters of QoE. In the technical parameters, the authors have considered network level and application level QoS and for the non-technical the authors have considered users perception, experience and expectations.

Authors of [6] have analysed the feasibility of implementing live video streaming protocols into web applications with the use of WebRTC. In conclusion, the authors stated that their implementation showed peer to peer streaming is currently possible for a larger set of clients. However, the experiments have also shown that with the current limitations of WebRTC implementation is currently not practical. The authors have also stated that, Browser vendors have to undo their performance restrictions, so that the sophisticated peer to peer protocols have to be developed for WebRTC. Authors of [4] have analysed the performance of DASH and WebRTC protocols while moving at walking speeds through a WiMAX network. The authors have collected the data from the application, network and physical layers under various wireless environments, they have also identified characteristics that directly impact the quality of video service in cellular data network. The authors have examined the effect of video metrics affecting the video quality on a wireless network for DASH and WebRTC protocols. In conclusion, to the despite attempts to adapt to the channel conditions they have found that, these services did not achieve acceptable service quality for the mobile users under different network conditions.

In paper[7], describes about the overview of the new emerging HEVC standard that has been developed and standardised collaboratively. In paper[8], describes about the performance evaluations of different implementations on HEVC. The authors have compared the encoders based on the perceived video quality and encoding speed. In conclusion, the x265 encoder provided higher encoding speed among tested encoders, though the video quality can be slightly worse compared to other implementations under similar conditions. In the paper[10], the authors have compared H.265/HEVC encoder and H.264/AVC in terms of video traffic and statistical multiplexing characteristics. They have also illustrated traffic characteristics and statistical multiplexing of the video encoded with SVC extension of H.264/AVC as well as 3D video encoded with MVC extension for H.264/AVC. Comparison of rate-distortion (RD) and rate variability-distortion before and after smoothing as well as link bitrate requirements have been evaluated for H.264/AVC and H.265/HEVC for a single video. The authors have found that with the advancement of the video coding standards has led to increased RD efficiency that is higher video quality for the prescribed mean video bitrate and also increased traffic variability at the encoder output.

2.2 Background

2.2.1 Video Streaming

Video streaming is a process wherein a video is being transmitted from a source to a destination/multiple destinations. In this thesis, a video is streamed from a server to client

Video streaming is basically based on two fundamental activities:

- Creating digital content using compression techniques.
- Transmitting content over the network.

Compression techniques are used to avoid transmitting a raw video over the network as it is extortionate. Transmitting a raw video consumes more network and storage resource. Hence, compressing a video plays a vital role in video streaming.

2.2.2 Video Compression

In the video compression, raw video is compressed using disparate algorithms and mechanisms. These are done to reduce the video and file size, which in turn reduces the consumption of network resources while in transmission. In wired as well as wireless networks, any video which is uncompressed consumes more bandwidth and storage when compared to a compressed video. In order to eradicate this problem, video compression is a must.

Video compression can be achieved in two methods:

- Lossless Compression - In this method, the original video is compressed to a smaller size wherein there isn't any loss of data. This method is not recommended for large sized videos, as it requires more network resources.
- Lossy Compression - In this method, the original video is compressed to much smaller size, which results in loss of information. This reduces the quality of the video to a great extent.

The main attributes involved in video compression are to maintain balance in between video quality and the size. Various codecs have been developed to serve this purpose.

2.2.3 Video Codecs

Video compression involves applying an algorithm to the source video to create a compressed file which is ready for the transmission or storage. The two

components involved in compression of video are Encoder and Decoder, together called as CODEC, as shown in the Figure 2.1. A codec has the capability of encoding and decoding. The raw video is compressed into small sized video that is transmittable over a network. Some of the commonly used video codecs are H.265/HEVC, H.264/AVC, H.263, H.262, etc. Due to the increase in the communication of the videos, many video coding techniques have been developed to provide high quality video streams using the available bandwidth [17][31].

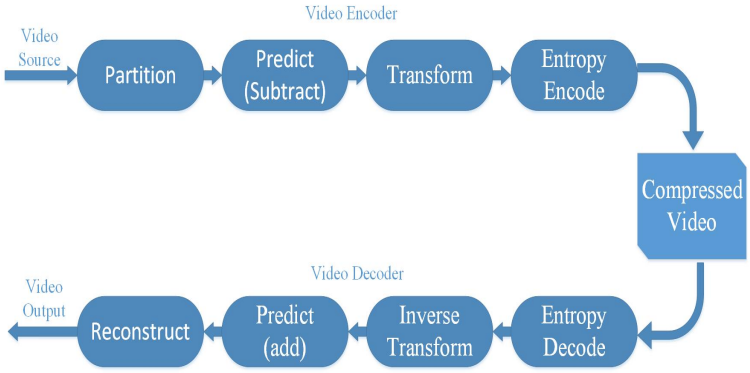

Figure 2.1: Video Codec

2.2.4 VLC Media Player and its functions

The Video Lan Client (VLC) media player is an open source which is released under the General Public License (GNU) and written by the Video Lan Project. VLC is a free software and any licensed version can be used without any legal restrictions. VLC has started in the year 1996 as an academic project. Its main purpose is to stream the videos between server and client across a network. The main purpose for the development is for the research purpose rather than entertainment. This media player can be installed from the flash drive and can be extended using the Lua scripting language. The GUI (Graphical User Interface) is based on QT4 for both Linux and Windows. QT4 is delegated Widget toolbox. The QT4 is the default, plain, graphical interface to VLC made utilising QT library to streamline the formation of use to windows, Mac OS X and Linux. Presently, the Video Lan Project was produced by benefactors all around the world and facilitated by Video Lan non-benefit association. VLC 1.1.0 has more than 380 modules [12].

2.2.4.1 Functions

VLC is a packet based media player which has the capability to play damaged, fragmented and unfinished video content. It can bolster a few record

arrangements including AVI, ASF, AAC, OGG, AIFF, MPEG, FFMPEG, REAL and so forth. VLC has the point of preference to dispatch one or more interfaces i.e. it can play several audio or video files at the same time. It has a particular outline to incorporate modules for new record designs, codec or gushing system. The VLC player can incorporate components like media and codec data. From this, one can know the codec data such as frame rate, bit rate, aggregate number of frames in the file and so forth. Snapshots of the video can also be taken in VLC.

VLC has a remote control interface and it acts as a streaming server by means of the media player. It can be controlled from a remote area and distinctive interfaces i.e., by means of HTTP. At point when a video is streamed by utilising remote control interface the receiver can play or stop the video as required by the receiver but not by the person. From, the alternative playback, the feature can be seen at quicker or slower rate. Moreover, the video can be viewed frame by frame by using advanced options, and subtitles can be added to the video. We can utilise firewire link to screen HDV camera feature amid playing. VLC has also the feature to play ISO image files from disk image. VLC can likewise play all audio and video formats supported by libavcodec and libavformat. VLC can support file formats like FLV or MXF and different encoders such as H.264 or MPEG-4 or H.265. It has modules for codec which are not based on FFMPEG libraries. The fundamental purpose behind utilising VLC player is that it accommodates several file formats, transcoding techniques and supports different protocols.

VLC Media Player can be utilised for streaming the video by using different protocols such as HTTP, RTSP, RTP and UDP etc. It is equipped for playing Motion Picture Expert Group (MPEG), Transport Streams (TS) and these streams were utilised as a part of this work to accumulate obliged data from the player. It has a filter that can distort, rotate, split, de-interlace and mirror videos. VLC can stream live from link boxes to PC utilising firewire association with screen or HDTV. In our proposal we are using the H.265 codec with the HTTP, RTSP and RTP protocols [12].

2.2.4.2 VLC Player Modules

VLC utilises a measured framework, which permits including effortlessly new functionalities and configurations. Taking after substance gives a presentation of some VLC modules. There are more than 380 modules for VLC 1.1.0. They assume a critical part for streaming, converting, sharing and so forth[12].

Access Modules: VLC can be used for streaming as it provides accessibility to different protocols. For each protocol a module with that name is characterised. VLC pursues a stream from distinctive sources using these modules, for instance: HTTP, UDP, RTP, FTP, and RTSP. The CDDA (Compact Disc Digital Audio) is utilised for sound CD data. The DVB (Digital Video

Broadcasting) module permits pursuing DVB-T, DVB-S and DBC-C satellite, physical alternately link cards. The DVD info module has been superseded by the DVDPLAY. The VCD module is utilised for pursuing VIDEOCD data [12].

Demuxers: When a video is streamed, both audio and video is received in containers format. The Demuxers extract the video and audio content from the video stream and pass them to the decoders. These modules permit the player to pursue various types of configurations as indicated by their type such as AVI, ASF, AAC, OGG, AIFF, AU, WAV, MP4, MKV, FFMPEG, REAL[12].

Video Outputs: The video output modules permit VLC to show feature on the screen. In view of the kind of the framework, VLC picks the most suitable feature module. We can compel any particular module from the order line by utilising "vout modulename". The DirectX module show feature utilising "Microsoft Direct X" libraries and it is suggested for Win32 port. This is a fundamental feature yield module of sort X11. The xvideo is for GNU/Linux working frameworks, what's more, it requires an xvideo-agreeable realistic card [12].

Video Filters: The video filters channel modules permit performing alterations of the rendered picture (de-interlacing, contrast, saturation, adjusting and cropping etc.). They are accessible from the command line control and but not from the GUI. The adjust module permits to alter image contrast, saturation and brightness. The CRT (Cathode Ray Tube)screens are equipped for playing simple feature outlines which is troublesome for present day screens. Consequently a de-interlace channel module was acquainted with proselyte simple casing into computerised one and it is helpful with streams originating from advanced satellite station. The transform module allows rotating the image to 90, 180 and 270, flat flip (hflip) and vertical-flip (vflip). The distort filter channel module makes twisting to the video and the invert channel modules reverses all the colours of the video which is empowered as a matter of course [12].

Audio Outputs: These modules help us to choose the kind of output for audio framework. It picks the most appropriate module or else we can drive to change particular module from command line using "-aout modulename". The OSS module actualises the open sound framework, which is utilised as a part of the Linux environment. The wave output is an audio output module which is used for the windows port and other modules are CORE AUDIO, DIRECT, OSS, ALSA, ESD, and ARTS [12].

Interface Modules: These are either graphical or control interfaces. The dummy interface is used when no interface is required (i.e. show feature with no control keys). The motion modules allow the client to control VLC using mouse gestures. The HTTP interface module allows the client to control VLC through web browser remotely. RC is a remote control interface module with which the client can control VLC using command line controls, for instance play, stop and so forth. The skin modules permits the users to create themes for VLC media player

using XML files. There are operating system support modules and miscellaneous modules for supporting different operating systems and additional modules for random purposes. For Example: The disable function from the command line is used to debilitate the player. The SOUT module is an incidental module which is a propelled element of VLC that allows streaming an MPEG-1, MPEG-2, MPEG-4/DivX record or DVD and HEVC [12].

2.2.5 H.265/HEVC Encoder

High Efficiency Video Coding (HEVC) also known as H.265, is a new video compression standard was developed by the Joint Collaborative Team on Video Coding. HEVC was developed with aim of providing twice the compression efficiency when compared with its previous encoder H.264/AVC. Although the compression efficiency results vary depending on the type of content and the encoder setting used when streaming a particular video. The End users take the advantage of this encoder as it takes up half of the compression efficiency when compared to AVC. When a HEVC video is compressed to the same file size or bitrate as AVC, HEVC delivers significantly better visual quality. Most of power of video compression standards comes from the technique known as motion compensated prediction. Blocks of pixels are encoded by making the reference to one another area in the same frame (intra-prediction) or in another frame (inter-prediction). One of the new features used is Quad Tree Structure. HEVC does not use fixed macro block sizes as the earlier standards. Instead of macro-block, the term Coding Tree Unit (CTU or CU) is used. CTU consists of luma Coding Tree Block (CTB), Chroma CTB's and syntax. For a luma component, the CTB size can be of 16x16, 32x32 or even 64x64. These CTB's can be further divided in smaller blocks called Coding Blocks (CB). These blocks do not have the same size within the CTB, their sizes are determined by the encoder based on the image content. This Encoder also supports VLC Media player which is an Open Source. This encoder can be used to stream the video over the Internet[7][8][9][10][19].

2.2.6 Video Transmission

The streaming protocols are designed to provide data transmission, network addressing and services between server and the client. The transport protocols are used as a communication medium between the streaming servers and clients. TCP and UDP video streaming techniques are the major techniques used at the transport layer. Currently, TCP is the most widely used protocol in the internet. The transmission control protocol (TCP) is a byte stream, connection oriented and reliable delivery transport layer protocol.

TCP is said to reliable due to some of the mechanisms such as checksums, re-transmissions and sequencing. The two main mechanisms of using TCP are

congestion control and flow control in the context of video streaming. TCP has retransmission capabilities due to its undesirable transmission delays. Many video streaming services are usually based on the HTTP over TCP streaming methods[29]. Some of the advantages of HTTP are data integrity, omnipresence and firewall friendliness. RTSP is used for establishing and controlling the media session between end points. WebRTC protocol uses RTP to stream the data.

2.2.7 Transmission Control Protocol (TCP)

TCP is a connection oriented and reliable transport layer protocol that uses a three-way handshake method to establish and it maintains a session between sender and the receiver. The three-way handshake has the following methods:

- The client sends a SYN (synchronisation) packet to the server, requests to server to synchronise its sequence numbers with that of the client. A SYN segment cannot carry the data, but it consumes one sequence number.
- The server responds to the client by sending an acknowledgment (ACK) along with a SYN. A SYN and ACK does not carry the data, but does consumes one sequence number.
- Finally, the client acknowledges the server by sending an acknowledgement packet.

TCP has also congestion control mechanism that adjusts the transmission rate by limiting the TCP connection. The TCP congestion control restricts the packets to transmit at lower rate.

2.2.8 User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It is suitable for a process that requires simple request-response communication with a little concern for flow and error control mechanisms. Multicasting capability is supported for UDP protocol but not in TCP protocol. UDP uses routing protocols like Routing Information Protocol (RIP). It also performs limited error checking [17].

Chapter 3
METHODOLOGY

Chapter 3

Methodology

This chapter explains about the measurement environment. The technical aspects of the research work are discussed as well as the experimental setup, configurations done to collect the measurements and procedure of the research work.

3.1 Experimental Setup

This chapter elucidates the experimental setup, to observe the characteristics of packet loss and delay variation affects on the video quality at end user and a set of experiments have been conducted in different scenarios. The Table 3.1 depicts about the environment setup that has been used for conducting the experiments. The reason for extending the architecture of the testbed for the clients has been explained in Section 3.2.5.

3.2 Setup Design

In this section, the experimental setup of the research will be explained. In this experiment, video streaming is done from server to client over HTTP, RTSP and WebRTC.A single video file is downloaded and saved on the server for

Hardware	**Server**	**Client-1**	**Client-2**	**Client-3**
OS	Ubuntu Server 14.04 LTS	Ubuntu Server 14.04 LTS	Ubuntu Server 14.04 LTS	Ubuntu Server 14.04 LTS
RAM	4GB - 1333 MHz	4GB - 1333 MHz	4GB - 1333 MHz	4GB - 1333 MHz
Processor	Intel Xeon 3.40 Ghz Quad	Intel i3 - 1.7 Ghz Quad	Intel i3 - 1.7 Ghz Quad	Intel i5 - 1.7 Ghz Quad
Hard Drive	500 GB - 5400 rpm	500 GB - 5400 rpm	500 GB - 5400 rpm	500 GB - 5400 rpm
LAN	Gigabit Ethernet	Gigabit Ethernet	Gigabit Ethernet	Gigabit Ethernet
Platforms	HP Proliant DL380 G4	Dell Latitude D620	Dell Latitude D820	Dell Latitude E5540

Table 3.1: Hardware Specifications

streaming the video. Chang et al [30] has described three typical types of videos on YouTube. Based on these statistics, a "super HD which is amazing" video is used in the experiment is considered for streaming. The resolution of the video is 480 pixels, the duration of the video is four minutes thirty seconds. Here, the server and client machines are on Linux Operating Systems. In between server and client, equipment's required are placed to trace out the results for packet loss and delay variations. In this setup two hosts are equipped with traffic shaper in between. The traffic shaper is used to control the traffic in the network going from HOST A (Server) to HOST B (Client). The experimental setup is configured in such a way that, the video will be streamed from HOST A (Video Originator) to HOST B (Video Receiver) at the other end. The Connections are made with Ethernet Cables with the required Static IP address. In this scenario, HOST A sends packets to HOST B via emulator. In the emulator, packet losses and delays are introduced into the network to find out the video quality at the end user while streaming the video using HTTP, RTSP and WebRTC protocols. The setup is mount to Measurement Point (MP) to capture the traffic flowing from sender to the receiver. When the video is received at HOST B, the packets at Application Layer and Network Layer are captured. The experimental setup is shown in the Figure 3.1.

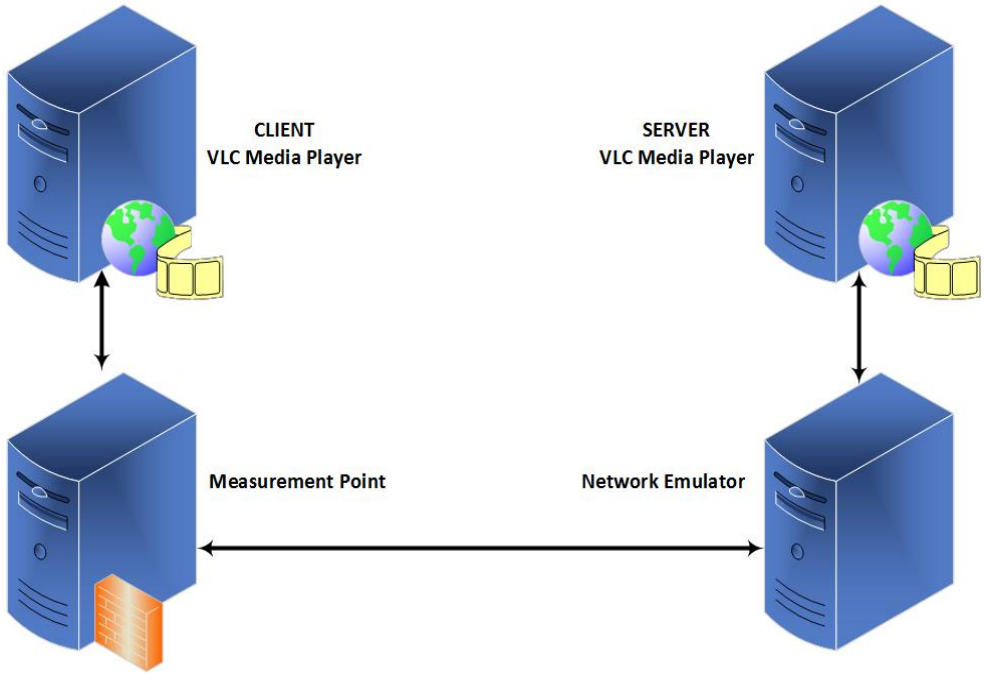

Figure 3.1: Experimental Setup

3.2.1 Measurement Point

Measurement Point (MP) is used for measuring the overall timestamps of the packets received. MP obtains the sender time as well as the receiver time of the packet. With the help of wiretaps, MP collects the data at both sending and the receiving sides. The main role of the wiretaps is to capture the traffic that is flowing in the network. It will duplicate the traffic and passes to MP. The MP consists of two Endace DAG 3.5E cards [26][27]. These DAG cards are placed near the sender and the receiver sides. These cards are synchronised with respect to time and frequency respectively, by using Global Positioning System (GPS) antenna for obtaining the timestamp accuracy of 60ns on network DAG cards.

3.2.2 MArC

MArC (Measurement Area Controller) is a central subsystem in the Measurement Area that manages the MP's. According to applied rules of MArC, the MP will tap the traffic. It recognises the users request information and sends that to MP.

3.2.3 Consumer

Consumer is the system controlled by the user. It is a Linux based environment system. The replicated packets captured by Data Acquisition and Generation (DAG) cards are stored in the consumer. It has libcap-utils-7.0.8 installed on it which is used to convert the binary traces to human readable text format [26][27]. Here, the network traffic which is captured is saved in text files. These stored files are used for further analysis for different protocols.

3.2.4 VLC Media Server

VLC Media server has server software that delivers streaming video and audio content as streaming on a variety of platforms. VLC Media Player can deliver multi bitrate which is live and as well as video on demand. This research has been performed by using Video on Demand (VoD). It is fully interactive server for streaming multimedia content with a full support for H.265 encoder. In this research VLC 2.1.5 Rincewind version has been used on the Linux Platform. The specifications of the Laptop used is HP Server with Intel Xeon 3.40 GHz with Ubuntu 14.04 installed on it.

3.2.5 VLC Media Client

The VLC Media client receives the stream sent from the server. The reason for extending the architecture of the testbed due to the occurred hardware

problems on the initial client setup. Severe electrical fluctuations were imposed on client-1 which resulted in destruction of hardware, forcing to move onto another client with hardware specifications illustrated in the Table 3.1. Client-2 was again replaced with client-3, this was a temporary arrangement due to the lack of sufficient processing power when compared to first client. This was just a theoretical assumption that streaming of video would have an effect on the change of processor (i.e., i3-i5). There hasn't been any change of the values when the setup has be shifted from client-2 to client-3.

3.2.6 NetEm

The Traffic shaping and emulation of network properties is useful in cases like delay, loss, duplication or re-ordering. NetEm or TC shaper belongs to the Traffic Control (TC) bandwidth provisioning package of Linux. In this research, NetEm is placed in between the server and the client to inject traffic onto the network. To capture the video traffic before and behind the shaper, the Distributive Passive Measurement Infrastructure (DPMI) is used. It acts as an interface between them. If traffic such as packet loss, delay are injected on the network, all the packets will be affected with respective configurations [28][29].

3.2.7 Packet Loss

The packet loss determines the number of packets that have been lost at destination when compared with the source. It outlines as the ratio of packets lost at the destination to the source on an Ethernet Packet Switch network. In our research, we have considered 0%, 1% and 2% packet loss.

3.2.8 Delay

The delay in the network is uniform. It adds the delay to all the packets flowing out of the Local Ethernet. In this research, a delay of 0ms, 10ms, and 20ms are injected into the network.

3.2.9 Bitrate

The bitrate is the rate at which bits are transferred from one location to another location. In other words, it is the measure of data that has been transmitted in a given amount of time. Bitrate rates are commonly measured in bits per second (bps), kilobits per second (kbps) or megabits per second (Mbps).

3.2.10 Packet Counts

Packet Counts are the number of packets received, when data/information is transmitted from one location to the other. For example, packets received by the client, when the data is sent by the server.

3.3 Experimetal Procedure

The experimental setup consists of a video streaming server, i.e., VLC at the server, VLC Media Player at the Client, Measurement Point, and Network Emulator. The streaming server namely VLC Media player is used to send the encoded video sequences to the VLC Client. The video originator will send the packets to the client using HTTP, RTSP and WebRTC protocols. The NetEm traffic shaper is used to inject loss and variable delays in the network going from server to client. DPMI is used to capture the traffic at the network layer and VLC stream statistics are captured at the application layer when the video is streamed from server to client. Distributed Passive Measurement Infrastructure (DPMI) is used with an Endance DAG card equipped with MP in order to verify the packet loss and delay variations at the network layer. The streamer and the VLC Media player are installed on the Linux Ubuntu 14.04 LTS platforms for both server and client. The shaper is installed on Linux Ubuntu 12.04 LTS. These are connected using Ethernet cables. The IP addresses are assigned to both server and client. The static IP address for the server is 192.168.186.221 and for the client is 192.168.186.227 with a net mask of 255.255.255.0. The port numbers used for streaming different protocols are HTTP: 8080, RTSP: 5544 and WebRTC: 5004. Initially Wireshark has been used to check the traffic flowing from server to client when a video is being streamed and when the video was streamed from server to client, the video qualities retrieved on the client were checked by some users.

A TCP socket program has been written at both server and client. This socket program creates a socket between server and client at port 7777. When the experiment is carried out, the TCP client program sends the VLC streaming command to the TCP server, then at the server side, the streaming process is carried out and at the client a bash script has been written. This script retrieves the stream that is sent from the server and also automatically a file is created wherein the VLC streaming statistics are appended into that file. A single experiment is carried out for 40 times for HTTP, RTSP and WebRTC. The values obtained at application layer and the network layer are collected and tabulated. The average, minimum, maximum, standard deviation and confidence intervals are calculated and tabulated.

The metrics are collected separately for application layer and network layer. At the application layer, when the video is streamed from the server to

the client, a text file is created automatically at the client side, this text file is updated every second appending the stream statistics obtained while the video is in transmission. The input bitrate, input bytes read, demux bitrate and demux bytes read are collected and are appended in to the text file automatically, using an automation script(shell script). The values are collected till the video stream is completed. Using another automation script (shell script), all the four metrics are filtered out into separate text files, from which the average of all the values are calculated and are appended into a text file. This process is repeated for 40 times thereby obtaining 40 text files with average values separately for all the four metrics. Later on, using another automation script, final average value of all the average values is calculated. Hence, obtaining average values for four metrics. This process is repeated for three of the protocols and for all the scenarios done in the experiment.

At the network layer, using streamstats of libcap utils, the packet counts at the client side are parsed and are appended into a text file. Forty sets of packet count values are appended into the text file for all the scenarios. From the collected packet count values, minimum, maximum, average, standard deviation and the confidence interval values are calculated and tabulated. This process is repeated for three of the protocols used and for all the scenarios done in the experiment.

Chapter 4
RESULTS AND ANALYSIS

Chapter 4

Results and Analysis

This chapter gives a brief description on the results obtained in the experiment scenarios mentioned earlier. The results obtained describes the working of video streaming based on the HTTP, RTSP and WebRTC protocols. Average stream bitrate value at application layer and packet counts at network layer have been found out. The results have been found out for Application layer and Network layer. From the obtained values and packet counts, the minimum value, maximum value, standard deviation, its average value and its confidence intervals for 95% are found out.

The experiment was carried out in the following scenarios separately:

- Delay of 0ms and 0% loss ratio.
- Delay of 0ms and 1% loss ratio.
- Delay of 0ms and 2% loss ratio.
- Delay of 10ms and 0% loss ratio.
- Delay of 20ms and 0% loss ratio.

The results have been carried out in two cases namely, application layer and network layer. In the application layer, input bitrate, input bytes read, demux bitrate and demux bytes read were collected and analyzed, in the network layer, the packet counts were collected and analyzed.

4.1 Mean Value calculations

Average Value

The average value is calculated using the below formula.

$$\overline{X} = \frac{\Sigma X_i}{N}$$

where,

- $\overline{X}$ is mean,
- ΣX_i is sum of all data values,
- N is number of all data values.

Standard deviation

The formula which was used for calculating standard deviation is as follows:

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{\Sigma(X_i - \overline{X})^2}{N}}$$

where,

- σ is standard deviation,
- σ^2 is variance,
- $\overline{X}$ is mean,
- X_i is each data value,
- N is number of all data values.

Confidence Interval

The formula which is used for calculating the confidence interval is as follows:

$$\left[\bar{X} - z_{1-\frac{\alpha}{2}}\frac{s}{\sqrt{n}}, \bar{X} - z_{1+\frac{\alpha}{2}}\frac{s}{\sqrt{n}}\right]$$

where,

- $\bar{X}$ = average; serves as an estimator for μ
- $\frac{s}{\sqrt{n}}$ = estimation of the variation of the mean $\sqrt{Var[\bar{X}]}$
- $z_{1-\frac{\alpha}{2}}$ = percentile of the Normal distribution.
- $\bar{X} - z_{1+\frac{\alpha}{2}}\frac{s}{\sqrt{n}}$ = half-size of the confidence interval

PROTOCOL	**HTTP**			**RTSP**			**WEBRTC**		
Loss Ratio	0%	1%	2%	0%	1%	2%	0%	1%	2%
Minimum (Kb/s)	94.545	94.700	94.545	94.848	95.638	95.637	94.736	94.592	94.728
Maximum (Kb/s)	94.856	94.856	94.856	95.25	96.022	96.022	95.57	95.528	95.512
Average (Kb/s)	94.848	94.836	94.848	95.192	95.939	95.883	95.160	95.12	95.13
Standard Deviation	0.0491	0.0520	0.049	0.087	0.148	0.178	0.196	0.199	0.206
Confidence Interval	0.015	0.016	0.0152	0.027	0.046	0.055	0.060	0.061	0.063

Table 4.1: Demux bitrate values for variable loss ratios

4.2 Application Layer

In the application layer, four metrics were studied and analysed. They are namely, input bitrate, input bytes read, demux bitrate and demux bytes read respectively for each of the three protocols HTTP, RTSP and WebRTC.

When the video is streamed from server to client, the stream statistics are recorded by the VLC media player at the application layer and it shows the four parameters mentioned above. Stream statistics in the sense, the VLC media player records how much network the video is using to successfully reach to the client from the server. Hence, it records the bitrate and the bytes read for demux and input.

In this section, demux bitrate is considered and analysed. Demux bitrate is nothing but the stream bitrate of the video. Stream bitrate refers to the amount of data that is being displayed on the screen at any given point. The reason for considering the demux bitrate is that, when the video is streamed from server to the client, the stream bitrate collected at the VLC describes its usage. The rest parameters such as the input bitrate and bytes read, and demux bytes read analysis are presented in the Appendix B. Input bitrate refers to the data that is being passed into VLC.

Obtained values for respective metrics are visualized and analyzed to derive which protocol is good when compared to the other. Delay of 10ms and 20ms were injected, and a loss ratio of 1% and 2% were included in the analysis part, and derived from the representation on which protocol is performing well when compared to the other. The following graphical representations will describe the conclusions. The Tables are categorised into Demux Bitrate, Demux bytes read, Input Bitrate and Input bytes read.

This Table 4.1 tells us how the average demux bitrate varies with respect to loss ratio for each of the three protocols.

From the Table 4.1, we can derive that, for 0% and 1% loss ratio, using RTSP protocol, yielded more demux bitrate when compared to the other.

PROTOCOL	**HTTP**			**RTSP**			**WEBRTC**		
Delay	0ms	10ms	20ms	0ms	10ms	20ms	0ms	10ms	20ms
Minimum (Kb/s)	94.545	94.545	94.545	94.848	95.637	95.637	94.736	94.824	94.68
Maximum (Kb/s)	94.856	94.856	94.856	95.25	96.022	96.022	95.57	95.608	95.512
Average (Kb/s)	94.848	94.84	94.840	95.192	95.947	95.773	95.160	95.174	95.496
Standard Deviation	0.0491	0.049	0.0685	0.087	0.143	0.179	0.196	0.167	0.1906
Confidence Interval	0.015	0.0152	0.0212	0.027	0.0445	0.055	0.060	0.051	0.059

Table 4.2: Demux bitrate values for variable delays.

Whereas, in the case of 2% of loss ratio, WebRTC protocol yielded more demux bitrate when compared to the other. Hence, HTTP is more reliable when compared to RTSP and WebRTC as it has less bitrate.

The Table 4.2 describes the variation of Demux Bitrate with respect to delay. It shows that for 0ms, 10ms and 20ms delay, RTSP protocol yielded more demux bitrate, whereas HTTP yielded less bitrate. Hence, HTTP is considered to be a good protocol on comparing with other protocols.

The Table 4.1 shows the comparison between the confidence intervals obtained for 0%, 1%, 2% loss ratio for the three protocols used in the experiment. It shows us the variation in the confidence intervals for respective cases. It shows us that, the confidence intervals of WebRTC are relatively higher when compared to the other two protocols. In each of the case of loss ratio, WebRTC confidence interval is higher, whereas in HTTP it is the least among the three and RTSP, is in the middle. Hence, HTTP is more reliable and is accurate when compared to the other three.

The Table 4.2 shows the comparison between the confidence intervals obtained for 0ms, 10ms, 20ms delay for the three protocol used in the experiment. It shows us that, the confidence interval for WebRTC is relatively higher when compared to the other two. HTTP is having the least confidence interval among the three, and hence can be derived that, HTTP is more reliable when compared to the other three.

4.3 Network Layer

In the network layer, the number of sent packets are evaluated and analyzed. HTTP uses TCP to transmit and receive packets whereas in the case of RTSP and WebRTC, it uses UDP to transmit and receive the packets. The comparison here made, is related to the memory usage of the packets and the error rate probability by the packets.

PROTOCOL	**HTTP**			**RTSP**			**WEBRTC**		
Loss Ratio	0%	1%	2%	0%	1%	2%	0%	1%	2%
Minimum	2924	2917	2925	10179	10179	10180	6560	6555	6561
Maximum	2931	2931	2932	10212	10209	10209	6576	6575	6574
Average	2857	2858	2859	9955	9957	9956	6412	6412	6412
Standard deviation	2.753	3.021	1.958	7.87	6.14	7.320	4.453	5.13	3.96
Confidence interval	0.853	0.936	0.606	2.44	1.903	2.268	1.38	1.59	1.22

Table 4.3: Packet count values for variable loss ratios

PROTOCOL	**HTTP**			**RTSP**			**WEBRTC**		
Delay	0ms	10ms	20ms	0ms	10ms	20ms	0ms	10ms	20ms
Minimum	2924	2921	2917	10179	10180	10209	6560	6561	6561
Maximum	2931	2931	2931	10212	10209	10180	6576	6574	6574
Average	2857	2857	2856	9955	9956	9956	6412	6412	6412
Standard Deviation	2.75	3.50	4.18	7.87	7.44	7.44	4.453	4.58	4.59
Confidence interval	0.853	1.087	1.29	2.44	2.30	2.306	1.380	1.422	1.42

Table 4.4: Packet count values for variable delays

Generally, HTTP, RTSP and WebRTC are some of the video streaming protocols which are used to stream a video from the server to the client. The streaming protocols and also the video streamer plays a vital role in the transmission process. Hence, the packet counts are considered and studied to know the performance of each of the protocols used in the Experiment. The average number of packets count and the confidence intervals are calculated using statistical means to get a better analysis of the metrics. The analysis can be understood in two ways. HTTP uses TCP where as RTSP and WebRTC uses UDP to transmit and receive packets.

The Table 4.3, shows us the comparison between average packet counts obtained at the network layer for the three protocols used with 0%, 1%, 2% loss ratios.

The Table 4.3, represents the variation in average packet counts for the network layer. It can be seen that, the number of packets in HTTP is very less when compared to RTSP and WebRTC. The reason is due to the usage of different protocols used to transmit and receive the packets. HTTP uses TCP where as RTSP and WebRTC uses UDP. In this case, RTSP and WebRTC have greater chance to have less error rate due to loss of packets, where as in HTTP, error rate might be more but the network resource usage is very less as it is transmitting less number of packets.

The Table 4.4, shows us the comparison between averages for various delays injected at network layer for the protocols used.

The Table 4.4, represents the variation in average packet counts with variable delay. It can be seen that, HTTP uses less number of packets and RTSP uses more number of packets. Hence, it can be derived that, HTTP has a very less network usage when compared to RTSP and WebRTC and RTSP has less error rate due to loss of packets, when compared to HTTP and WebRTC.

The Table 4.3, shows us the comparison between the confidence intervals obtained for 0%, 1%, 2% loss ratio for the three protocols used in the experiment. It represents the variation in confidence intervals for the number of packet counts. It shows that, the confidence interval for HTTP is less in all the loss cases. Hence, it can be used in cases where in there is a considerable amount of loss ratio.

The Table 4.4, shows us the comparison between the confidence intervals obtained for 0ms, 10ms, 20ms delay for the three protocols used in the experiment. It represents the variation in Confidence Intervals for packet count. It shows that, the confidence interval for HTTP is less in all the cases when compared to the other two protocols. Hence, HTTP can used to transmit and receive packets in the network layer.

Chapter 5
CONCLUSION AND FUTURE WORK

Chapter 5

Conclusion and Future Work

Results of the experiments had shown how the network usage varies in streaming of a video. Primarily, the video is streamed from server to the client, and the network usage is evaluated in terms of demux bitrate in the application layer and the number of packets count in the network layer. Results have been taken out using three protocols namely, HTTP, RTSP and WebRTC. Each protocol has its own way of working and the comparisons shown earlier, gives a brief idea on which protocol is efficient, reliable when compared to the others. In the Application layer, the protocol which has less bitrate, more bytes read is reliable and efficient. In the Network layer, the protocol with a balance in the number of packets counted is reliable and efficient.

In Application Layer, based on the results obtained, HTTP is comparatively more reliable and efficient in some ways and WebRTC is reliable in some ways and RTSP being the protocol which can neither be neglected nor used frequently. Every protocol has got its own pros and cons. The Bitrate usage is less in HTTP which means it is using less network usage, and the Bytes read were more in WebRTC which picks up as many bytes of data as possible. Hence mentioned earlier, each protocol has got its own pros and cons.

Moving on to the network layer, HTTP uses TCP as the protocol to transmit and receive packets, whereas RTSP and WebRTC uses UDP as its protocol to transmit and receive packets. As part of the data transmission process, the packet data is encrypted with headers and trailers attached to it in the device before being transmitted, which is later decrypted at the receiving device to successfully acknowledge the reception of the packets. Hence, HTTP which is based on TCP has less packets when compared to RTSP and WebRTC, which means it makes use of lesser network resources. Nevertheless, RTSP and WebRTC which are based on UDP have more number of packets, which means the data loss while in transmission, is less when compared to that in HTTP. Thereby concluding that, HTTP is slightly preferable and reliable protocol to stream a video when compared to the other protocols, keeping in mind the pros and cons of each protocol.

The technique implemented in this study useful to analyze the performance of the protocols when using H.265 Encoder. As in this research, the per-

formance of the protocols for H.265 Encoder has been calculated on a laptop.This research work can be extended on to comparing on a mobile and laptop as the future work.

LINKING TO RESEARCH QUESTIONS

1. **What is the packet count measured at the network layer when HTTP, RTSP and WebRTC protocols were used to stream video over the network?**

   Ans: The Table 4.3, Table 4.4 contain all the average, minimum, maximum, standard deviation and confidence intervals of the packet counts collected in the experiment. Based on the values obtained and after analyzing them, the performance of the video stream using HTTP protocol was found out to be good on comparing with the other two protocols. Nevertheless, the performance of the video stream using RTSP and WebRTC protocols, was not bad. As, the overall packet size for RTSP is more on comparing with WebRTC, however we are not considering the packet size. We are mainly focused on the performance of these protocols.

2. **What are the streaming statistics measured at the application layer when HTTP, RTSP and WebRTC protocols were used to stream video over the network?**

   Ans: The Table 4.1, Table 4.2 shows the minimum, maximum, average and the confidence interval values of the demux bitrate obtained when the video is streamed. Based on the values and the analysis, it was found out that, the video streamed using HTTP protocol, was found to perform better on comparing with the other two protocols RTSP and WebRTC. On analyzing RTSP and WebRTC protocol statistics, both the protocols perform good but with few drawbacks, as the performance of the protocols depends on the streamer as well as the video streaming protocols.

3. **How does the performance of the above two cases (R1 and R2) vary when subjected to variable delays and packet loss in the network?**

   Ans: In the experiment, a traffic shaper is used to inject variable loss ratios and delays to the video stream, to analyze the performance in the network and application layer. Various experimental iterations were performed, by varying various parameters like loss ratio of 0%, 1% and 2% and a delay of 0ms, 10ms and 20ms. After the experiments were conducted, the same process mentioned in the RQ1 and RQ2 were performed. The obtained values are tabulated and analyzed. The analysis part is mentioned in the RQ1 and RQ2.

REFERENCES

References

[1] A. Biernacki and K. Tutschku, "Performance of HTTP video streaming under different network conditions," Multimed. Tools Appl., vol. 72, no. 2, pp. 1143-1166, Mar. 2013.

[2] H. Zhang, A. Al-Nuaimi, X. Gu, M. Fahrmair, and R. Ishibashi, "Seamless and efficient stream switching of multi-perspective videos," in Packet Video Workshop (PV), 2012 19th International, 2012, pp. 31-36.

[3] J. Lee, J. Kim, S. Kim, C. Lim, and J. Jung, "Enhanced distributed streaming system based on RTP/RTSP in resurgent ability," in Fourth Annual ACIS International Conference on Computer and Information Science, 2005, 2005, pp. 568-572.

[4] F. Fund, C. Wang, Y. Liu, T. Korakis, M. Zink, and S. S. Panwar, "Performance of DASH and WebRTC Video Services for Mobile Users," in Packet Video Workshop (PV), 2013 20th International, 2013, pp. 1-8.

[5] S. Ickin, M. Fiedler, K. Wac, P. Arlos, C. Temiz, and K. Mkocha, "VLQoE: Video QoE instrumentation on the smartphone," Multimed. Tools Appl., vol. 74, no. 2, pp. 381-411, Apr. 2014.

[6] F. Rhinow, P. P. Veloso, C. Puyelo, S. Barrett, and E. O. Nuallain, "P2P live video streaming in WebRTC," in 2014 World Congress on Computer Applications and Information Systems (WCCAIS), 2014, pp. 1-6.

[7] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[8] O. Zach and M. Slanina, "A comparison of H.265/HEVC implementations," in ELMAR (ELMAR), 2014 56th International Symposium, 2014, pp. 1-4.

[9] H. Koumaras, M. Kourtis, and D. Martakos, "Benchmarking the encoding efficiency of H.265/HEVC and H.264/AVC," in Future Network Mobile Summit (FutureNetw), 2012, 2012, pp. 1-7.

[10] P. Seeling and M. Reisslein, "Video Traffic Characteristics of Modern Encoding Standards: H.264/AVC with SVC and MVC Extensions and H.265/HEVC," Sci. World J., vol. 2014, p. e189481, Feb. 2014.

[11] T. Hosfeld, M. Fiedler, and T. Zinner, "The QoE provisioning-delivery-hysteresis and its importance for service provisioning in the Future Internet," in 2011 7th EURO-NGI Conference on Next Generation Internet (NGI), 2011, pp. 1-8.

[12] Anoop Vuppala; Sriram Lakshminarayana, "Measurement of user-related performance problems of live video streaming in the user interface," 33158, pp. 58. COM/School of Computing, 2012.

[13] Georg Westin MSE-2003:25, "Evaluation of Video-on-Demand Streaming Servers," pp. 21. Inst. for programvaruteknik och datavetenskap/Dept. of Software Engineering and Computer Science, 2003.

[14] J. Nightingale, Q. Wang, C. Grecos, and S. Goma, "Modeling QoE for streamed H.265/HEVC content under adverse network conditions," in 5th IET International Conference on Wireless, Mobile and Multimedia Networks (ICWMMN 2013), 2013, pp. 249âĂŞ253.

[15] OYEKANLU Emmanuel Adebomi; JOHN Samson Mwela, "Impact of Packet Losses on the Quality of Video Streaming," MEE10:44, pp. 40. COM/School of Computing, 2010.

[16] Ramesh Goud Guniganti; Srikanth Ankam, "A Comparison of RTMP and HTTP Protocols with respect to Packet Loss and Delay Variation based on QoE," pp. 68. COM/School of Computing, 2013.

[17] Raj Kiran Addu; Vinod Kumar Potuvardanam, "Effect of Codec Performance on Video QoE for videos encoded with Xvid, H.264 and WebM/VP8," pp. 55. Inst. for kommunikationssytem, DIKO/Dept. of Communication Systems, 2014.

[18] Vasanthi Dwaraka Bhamidipati; Swetha Kilari, "Effect of Delay/Delay Variation on QoE in Video Streaming," MEE10:45, pp. 60. COM/School of Computing, 2010..

[19] Sina Tamanna, "Transcoding H.265/HEVC", MSE-2013:04, pp. 71. COM/School of Computing, 2013

[20] C. Jee and K. G. Shin, "A DAVIC Video-on-Demand system based on the RTSP," in 2001 Symposium on Applications and the Internet, 2001. Proceedings, 2001, pp. 231-238.

[21] S. Q. Khan, R. Gaglianello, and M. Luna, "Experiences with blending HTTP, RTSP, and IMS [IP Multimedia Systems (IMS) Infrastructure and Services]," IEEE Commun. Mag., vol. 45, no. 3, pp. 122-128, Mar. 2007.

[22] H. Nam, K. H. Kim, B. H. Kim, D. Calin, and H. Schulzrinne, "Towards dynamic QoS-aware over-the-top video streaming," in World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a, 2014, pp. 1-9.

[23] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming video over the Internet: approaches and directions," IEEE Trans. Circuits Syst. Video Technol., vol. 11, no. 3, pp. 282-300, Mar. 2001.

[24] M. D. Di Benedetto, A. Di Loreto, A. DâĂŹInnocenzo, and T. Ionta, "Modeling of traffic congestion and re-routing in a service provider network," in 2014 IEEE International Conference on Communications Workshops (ICC), 2014, pp. 557-562.

[25] M. Westerlund, C. Perkins, and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP." [Online]. Available: https://tools.ietf.org/html/draft-ietf-rtcweb-rtp-usage-04. [Accessed: 01-Sep-2015].

[26] P. Arlos, M. Fiedler, and A. A. Nilsson, "A Distributed Passive Measurement Infrastructure," in Passive and Active Network Measurement, C. Dovrolis, Ed. Springer Berlin Heidelberg, 2005, pp. 215-227.

[27] P. Arlos, "On the Quality of Computer Network Measurements," Ph.D. dissertation, Blekinge Institute of Technology, 2005.

[28] (2012) NetEm. Linuxfoundation [Online]. Available: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem.

[29] S. Hemminger, "Network emulation with netem," in Linux ConfAu. Citeseer, 2005, pp. 18-23.

[30] H.Chang, H.Liu, Y.-W. Leung, and X. Chu, "Minimum latency server selection for heterogeneous cloud services," in 2014 IEEE Global Communications Conference (GLOBECOM), 2014, pp. 2276âĂŞ2282.

[31] I.E.Richardson, H.264 and MPEG-4 video compression: video coding for next-generation multimedia. John Wiley & Sons,2004.

APPENDICES

Appendix

A APPENDIX A

A.1 Network Configuration

To get the proper network configurations for the device, the network configuration setting are configurations are changed at /etc/network/interfaces at the terminal. The configuration of the laptop is configured as

Auto eth0

iface inet eth0 dhcp

Auto eth1

iface inet eth1 dhcp

After setting up the configuration the then enter the following commands in order to obtain IP address for the device.

#sudo ifdown -a && sudo ifup -a

To take up the above setting restart to the networking service under Linux OS.

#sudo /etc/init.d/networking restart

A.2 Emulator Commands

Packet Loss

#tc qdisc add dev eth2 root netem loss $X\%$

#tc qdisc add dev eth2 root netem loss $X\%$

Delay Variation

#tc qdisc add dev eth2 root netem delay Yms

#tc qdisc add dev eth2 root netem delay Yms

In the above stated commands the terms X stands for the packet loss and Y stands for the delay.

PROTOCOL	**HTTP**			**RTSP**			**WEBRTC**		
Loss Ratio	0%	1%	2%	0%	1%	2%	0%	1%	2%
Minimum (KiB)	1540.6	1540.28	1541.43	1543.77	1552.23	1551.60	1561.8	1561.58	1563.16
Maximum (KiB)	1550.7	1550.03	1549.93	1555.49	1561.94	1558.13	1577.65	1577.64	1575.97
Average (KiB)	1549.26	1548.47	1549.13	1552.23	1555.98	1555.66	1568.99	1570.47	1569.46
Standard Deviation	1.443	2.671	1.274	2.0379	1.871	1.551	3.542	3.592	2.973
Confidence Interval	0.447	0.827	0.395	0.631	0.580	0.480	1.097	1.113	0.921

Table B.1: Demux bytes read values for variable loss ratios

PROTOCOL	**HTTP**			**RTSP**			**WEBRTC**		
Delay	0ms	10ms	20ms	0ms	10ms	20ms	0ms	10ms	20ms
Minimum (KiB)	1540.68	1540.48	1540.48	1543.7	1552.33	1552.55	1561.8	1563.28	1563.13
Maximum (KiB)	1550.72	1550.33	1550.43	1555.49	1561.67	1557.73	1577.65	1576.72	1574.9
Average (KiB)	1549.26	1549.26	1549.10	1552.23	1554.07	1555.39	1568.99	1569.39	1569.43
Standard Deviation	1.443	1.462	1.915	2.037	1.569	1.517	3.542	3.688	2.773
Confidence Interval	0.447	0.453	0.593	0.631	0.486	0.470	1.097	1.143	0.859

Table B.2: Demux bytes read values for variable delays

A.3 Installation of VLC Media Player

#sudo apt-get update

#sudo apt-get upgrade

#sudo apt-get install vlc

Installation of H.265 encoder

sudo apt-add-repository ppa:strukturag/libde265

#sudo apt-get update

#sudo apt-get install vlc-plugin-libde265

B APPENDIX B

The Table B.1 shows us the variation between the demux bytes read, for 0%, 1%, 2% loss ratio for HTTP, RTSP and WebRTC respectively. It shows us that, HTTP protocol has very less demux bytes read and WebRTC has greater demux bytes read. Hence, WebRTC is more reliable as it reads more bytes in comparison with the other two protocols.

The Table B.2 shows the variation between demux bytes read for variable delays. It shows that the average value of WebRTC protocol for variable delays is comparatively greater than the other two protocols. Hence, it is more reliable

PROTOCOL	**HTTP**		
Loss Ratio	0%	1%	2%
Minimum (Kb/s)	97.37	97.37	97.37
Maximum (Kb/s)	98.47	98.772	98.68
Average (Kb/s)	97.59	97.66	97.57
Standard Deviation	0.241	0.372	0.234
Confidence Interval	0.074	0.115	0.072

Table B.3: Input bitrate values for variable loss ratios

when compared to the other two, as it reads more bytes when compared to the other two protocols.

The Table B.1 shows the variation between demux bytes read for variable losses. It shows that, the confidence interval of HTTP is less for the cases of 0% loss and 2% loss, and for 1% loss, RTSP has lesser confidence interval. Hence, HTTP is relatively good when compared to the others in 0% and 2% loss and RTSP is comparatively good in 1% loss cases.

The Table B.2 represents the variation in confidence intervals for demux bytes read for variable delays. It represents the variation in confidence intervals for demux bytes read. It shows that, the confidence interval for HTTP is less in 0ms and 10ms delay whereas in 20ms, RTSP is less. Hence, for 0ms and 10ms delays, HTTP is reliable and for 20ms RTSP is reliable.

The Table B.3 represents the variation in average input bitrate for HTTP for variable losses. It can be seen that, the bitrate is almost all similar in each of the loss case, that is, 0%, 1%, 2% loss. Hence, HTTP does not show any drastic changes to variable losses, and is immune to any losses occurring in the meanwhile.

The Table B.4 represents the variation in Average Input bitrate of HTTP for variable delays. It can be seen that, the bitrate is relatively higher in no delay case, and in 10ms and 20ms delay, it is much less when compared to the no loss case.

The Table B.3 represents the variation in confidence intervals for input bitrate. It can be seen that, the confidence interval is lower in the case of 0% loss and in 2% loss but is higher in 1% loss.

The Table B.4 shows us the comparison between the confidence intervals of HTTP for variable delays. It represents the variation in confidence intervals for input bitrate. It can be seen that, the confidence interval is higher in 20ms delay when compared to the other two cases, and in 10ms delay, the confidence

PROTOCOL	**HTTP**		
Delay	0ms	10ms	20ms
Minimum (Kb/s)	97.37	97.37	97.37
Maximum (Kb/s)	98.47	98.50	98.71
Average (Kb/s)	97.59	97.55	97.54
Standard Deviation	0.241	0.219	0.28
Confidence Interval	0.074	0.068	0.087

Table B.4: Input bitrate values for variable delays

PROTOCOL	**HTTP**		
Loss Ratio	0%	1%	2%
Minimum (KiB)	1574.68	1574.37	1578.97
Maximum (KiB)	1584.76	1584.00	1583.91
Average (KiB)	1583.34	1582.55	1583.26
Standard Deviation	1.450	2.668	0.941
Confidence Interval	0.449	0.8269	0.291

Table B.5: Input bytes read values for variable loss ratios

interval is relatively lower.

The Table B.5 shows us the comparison between the average input bytes read of HTTP obtained for 0%, 1%, 2% loss ratio.

The Table B.5 represents the variation in average input bytes read. It can be seen that, the average values are almost all similar to each other, hence does not yield any drastic changes the system is injected with loss ratio 0%, 1%, 2%.

The Table B.6 represents the variation in average input bytes read of HTTP with variable delays such as 0ms, 10ms, 20ms. It can be seen that, the average bytes read is higher in the no delay case and average is lower in case of 20ms delay. Hence, it is more reliable and accurate in no loss scenario.

The Table B.5 shows us the comparison between the confidence intervals of HTTP obtained for 0%, 1%, 2% loss ratio. It represents the variation in confidence intervals for input bytes read. It can be seen that, the confidence interval value is less in case of 2% loss and is greater in the case of 1% loss case.

The Table B.6 shows us the variation in confidence intervals of input bytes read for HTTP obtained for 0ms, 10ms, 20ms delay. It represents the variation

PROTOCOL	**HTTP**		
Delay	0ms	10ms	20ms
Minimum (KiB)	1574.68	1574.5	1574.5
Maximum (KiB)	1584.76	1584.33	1584.3
Average (KiB)	1583.34	1583.33	1583.17
Standard Deviation	1.450	1.466	1.918
Confidence Interval	0.449	0.454	0.594

Table B.6: Input bytes read values for variable delays

in confidence intervals for input bytes read. It can be seen that, the confidence interval is similar in the case of 0ms and 10ms delay where as in 20ms delay, it is greater. Hence, HTTP is reliable even if there is slight amount of delay.