

QoE Evaluation Across a Range of User Age Groups in Video Applications

By

Mujtaba Roshan

Submitted for the degree of Doctor of Philosophy

School of Electronic Engineering and Computer Science,
Queen Mary University of London,
London, UK

January 2018

Acknowledgements

First and foremost, I would like to thank the God Almighty for giving me the opportunity, strength and determination to complete my PhD.

I would like to express my deepest gratitude to my supervisor Dr. John Schormans for his continuous support throughout the course of my PhD. John facilitated meetings with me on very short notices, and explained several concepts very patiently and resourcefully. Whenever I needed a push in the right direction, John was only a Skype/phone call away whenever he was out of office.

I am also greatly indebted to Steve Uhlig, Professor of Networks and Head of Networks Research Group, to take time out of his busy schedule to meet me several times during my PhD, and for his support with facilitating the research opportunities.

I would also like to thank my secondary supervisors Jonathon Pitts and Vindya Wijeratne, and the independent assessor Tijana Timotijevic for their constructive feedback and encouragement at each assessment stage of my PhD.

I would like to acknowledge that this work was supported by the EPSRC (1589943).

Last but not the least, I am grateful to Intergence, a digital business transformation company, which supported my research.

Abstract

Quality of Service (QoS) measures are the network parameters; delay, jitter, and loss and they do not reflect the actual quality of the service received by the end user. To get an actual view of the performance from a user's perspective, the Quality of the Experience (QoE) measure is now used.

Traditionally, QoS network measurements are carried on actual network components, such as the routers and switches since these are the key network components. In this thesis, however, the experimentation has been done on real video traffic. The experimental setup made use of a very popular network tool, Network Emulator (NetEm) created by the Linux Foundation. NetEm allows network emulation without using the actual network devices such as the routers and traffic generator.

The common NetEm offered features are those that have been used by the researchers in the past. These have the same limitation as a traditional simulator, which is the inability of NetEm delay jitter model to represent realistic network traffic models, such to reflect the behaviour of real world networks. The NetEm default method of inputting delay and jitter adds or subtracts a fixed amount of delay on the outgoing traffic. NetEm also allows the user to add this variation in a correlated fashion. However, using this technique the outputted packet delays are generated in such a way as to be very limited and hence not much like real internet traffic which has a vast range of delays. The standard alternative that NetEm allows is generate the delays from either a Normal (Gaussian) or Pareto distribution. This research, however, has shown that using a Gaussian or Pareto distribution also has very severe limitations, and these are fully discussed and described in Chapter 5 on page 68 of this thesis. This research adopts another approach that is also allowed (with more difficulty) by NetEm: by measuring a very large number of packet delays generated from a double exponential distribution a packet delay profile is created that far better imitates the actual delays seen in Internet traffic.

In this thesis a large set of statistical delay values were gathered and used to create delay distribution tables. Additionally, to overcome another default behaviour of NetEm of re-ordering packets once jitter is implemented, PFIFO queuing discipline has been deployed to retain the original packet order regardless of the highest levels of implemented jitter. Furthermore, this advancement in NetEm's functionality also incorporates the ability to combine delay, jitter, and loss, which is not allowed on NetEm by default. In the literature, no work has been found to have utilised NetEm previously with such an advancement.

Focusing on Video On Demand (VOD) it was discovered that the reported QoE may differ widely for users of different age groups, and that the most demanding age group (the youngest) can require an order of magnitude lower PLP to achieve the same QoE than is required by the most widely studied age group of users. A bottleneck TCP model was then used to evaluate the capacity cost of achieving an order of magnitude decrease in PLP, and found it be (almost always) a 3-fold increase in link capacity that was required. The results are potentially very useful to service providers and network designers to be able to provide a satisfactory service to their customers, and in return, maintaining a prosperous business.

Papers out of this Research

Paper Title	Submitted/ submitting to	Status
Video-on-demand QoE Evaluation Across Different Age-Groups and Its Significance for Network Capacity	QSHINE 2017	Accepted
Emulation-based Evaluation of QoE and QoE-based QoS Measurement Requirements in Video on Demand	IET	In Progress

Table of Contents

Acknowledgements	2
Abstract	3
Papers out of this Research	4
Table of Contents	5
List of Figures	10
List of Tables	15
Glossary	16
Chapter 1 Introduction	17
1.1 Motivation.....	18
1.2 Goal	18
1.3 Skills and Tools Assessment.....	19
1.4 Design of Experiments	19
1.5 Outline of the Thesis	20
Chapter 2 Literature Review.....	21
2.1 Chapter Introduction	21
2.2 Quality of Experience (QoE).....	21
2.3 Mean Opinion Score (MOS)	22
2.4 QoE Research on Network Traffic.....	24
2.5 QoE in the Presence of Congestion (QoS as Metrics)	24
2.5.1 Congestion Control by the Bandwidth Method.....	25
2.6 Network Measurements involving Delay, Jitter, and Loss.....	26
2.7 Network Measurement Types	27
2.8 Network Traffic Patterns.....	28
2.9 Queueing Theory	30
2.10 Queueing Systems.....	31
2.11 The Use of Network Emulation Tools	32
2.12 NetEm (Network Emulator)	33
2.13 Previous QoE Research on NetEm	34
2.14 Previous Work Involving Different Age Groups	35
2.15 Chapter Summary	36
Chapter 3 Relevant Technical Concepts and Terms	37

3.1	Chapter Introduction	37
3.2	Video Properties	37
3.2.1	Bit Rate	37
3.2.2	Video Frame Rate.....	38
3.2.3	Video Aspect Ratio	39
3.2.4	Video Formats/Containers and Codecs	39
3.2.5	Video Resolution	40
3.3	VLC Media Player	42
3.4	Network Technical Terms	44
3.4.1	Transmission Control Protocol (TCP)	44
3.4.2	User Datagram Protocol (UDP)	45
3.4.3	First In First Out (FIFO).....	45
3.4.4	Time First In First Out (TFIFO).....	45
3.4.5	Packet First In First Out (PFIFO)	46
3.5	Chapter Summary	47
Chapter 4	Research Methodology	48
4.1	Chapter Introduction	48
4.2	The Router Lab (Previous Testbed).....	48
4.3	Design Considerations for the New Testbed	49
4.4	NetEm Limitations.....	49
4.5	The Approach Taken to the Use of NetEm	50
4.6	NetEm Testbed: The Initial Design.....	51
4.7	Testbed Improvement – Replacement of the RPi.....	51
4.8	Operating NetEm with PFIFO instead of TFIFO.....	52
4.9	The Use of USB Ethernet Adapter and its Limitations	55
4.10	NetEmBox-1	55
4.11	Custom Delay-Distribution-Table Implementation in NetEm.....	59
4.11.1	Creating a Delay Distribution Table	59
4.12	NetEmBox-1 as a Proxy Server.....	61
4.13	NetEmBox-2: The Extended Testbed Facilities	63
4.14	Testbed Wireless Streaming Functionality for Mobile Phones.....	64
4.15	The Final Upgrade: A More Robust Hotspot Design	65
4.16	Results Collected from NetEm-based Testbed are Realistic.....	66
4.17	Chapter Summary	66

Chapter 5 Experimentation with Artificially Added Loss and Jitter, and Bespoke Delay Distributions

68

5.1	Chapter Introduction	68
5.2	Preliminary Experiments.....	69
5.3	Packet Delay Independent of Packet Loss	69
5.4	Traffic Analysis with Added Packet Loss	74
5.4.1	Case 1: 0.01% Packet Loss.....	74
5.4.2	Case 2: 0.1% Packet Loss.....	78
5.4.3	Case 3: 1% Packet Loss.....	80
5.4.4	Case 4: 5% Packet Loss.....	81
5.4.5	Case 5: 10% Packet Loss.....	82
5.4.6	Error Bar Plot for Given Packet Loss Cases	83
5.5	PLP Testing in Combination with MOS	84
5.6	Investigating NetEm Jitter/Delay	86
5.7	Theoretical Validation of Normal and Pareto Distributions in NetEm	87
5.8	Jitter Implementation with Normal and Pareto Distributions.....	88
5.9	Activating Normal and Pareto distributions on a NetEm Interface	90
5.10	Experimentation with Jitter Using TCP and UDP Protocols	94
5.11	Experimentation with Packet Loss Using TCP and UDP Protocols.....	96
5.12	Custom Delay Distributions for Jitter and Packet Loss	97
5.12.1	Effect of Standalone Jitter and Loss on TCP vs UDP Protocols	101
5.12.2	Effect of Combined Loss and Jitter on Network Traffic QoE.....	101
5.13	Chapter Summary	102
Chapter 6	Evaluating MOS for Varying PLP Over a Range of Different Age Groups	104
6.1	Chapter Introduction	104
6.2	Mean Opinion Score (MOS) to Assess QoE for Video on Demand (VOD).....	104
6.2.1	QoE Evaluation by the Participants from 10-18 Age Group	105
6.2.2	Participants from 19-30 Age Group	109
6.2.3	Participants from 31-45 Age Group	111
6.2.4	Participants from 46-65 age group	114
6.2.5	Participants from over 65 Age Group	116
6.3	Chapter Summary	120
Chapter 7	Evaluating YouTube MOS for Varying PLP and Round-Trip-Time (RTT).....	121
7.1	Chapter Introduction	121

7.2	Experimentation Setup	122
7.2.1	Use of Bash Scripts.....	123
7.2.2	Experimentation Strategy	124
7.2.3	Analytical MOS Calculation	125
7.3	How the YouTube Video Download Works.....	126
7.3.1	From Requesting a YouTube Video to Viewing it	126
7.4	YouTube Content Delivery Infrastructure.....	127
7.5	YouTube QoE Experimentation with Packet Loss	129
7.6	YouTube QoE Evaluation with Automatically Adjusted Download Quality	131
7.6.1	RTT = 1ms.....	131
7.6.2	RTT = 50ms.....	134
7.6.3	RTT = 100ms.....	136
7.6.4	RTT = 150ms.....	138
7.7	YouTube QoE Evaluation with a Resolution of 720p as the Download Quality.....	140
7.7.1	RTT = 1ms.....	141
7.7.2	RTT = 50ms.....	143
7.7.3	RTT = 100ms.....	145
7.7.4	RTT = 150ms.....	147
7.8	Conclusion for YouTube QoE Experimentation	149
7.8.1	YouTube Formats and Codecs.....	149
7.8.2	Effect of Packet Loss on QoE of YouTube Traffic	150
7.8.3	Effect of RTT and Packet Loss as a Combination	151
7.9	Chapter Summary	151
Chapter 8	Significance of VOD QoE Evaluation Across Different Age Groups.....	153
8.1	Chapter Introduction	153
8.2	Experimentation	153
8.3	Analysis of Effect of Age Related MOS Display on Capacity Dimensioning	154
8.4	Results.....	156
8.5	Default Queue Sizing for Router Buffers.....	159
8.6	Chapter Summary	159
Chapter 9	Discussion, Recommendations for Future Work, and Conclusion.....	161
9.1	Discussion.....	161
9.2	Future Work	163
9.2.1	General Recommendations	163

9.2.2	Technical Recommendations	164
9.3	Conclusion.....	164
References	165
Appendix	183
Appendix A	Challenges Faced in Research	183
	Network Router Lab: The Challenges	183
	Issues with the Remote Access to the Testbed	183
	Technical Difficulties Experienced	184
	Networks Lab Overheating	185
	NetEm Testbed: The Challenges	185
	Testbed Design Model	185
	Server and Client Architecture.....	186
	Replacing TFIFO with PFIFO in NetEm	186
	Testbed Advanced Interfacing and Connectivity	187
Appendix B	Initial assessment of research related tools and skillset	188
	Necessary Skills	188
	LabVIEW	188
	Internet Routers.....	188
	CISCO Operating System	188
	Required Tools	188
Appendix C	A simple version of Instructions to operate NetEm with a PFIFO queue	189
Appendix D	Server-side Settings for VOD Streaming	191
	Server-side Network Settings.....	191
	Initiating Video Streaming from the Server	195
Appendix E	Client-side Settings for VOD Streaming	201
	Client-side Network settings.....	201
	Opening a Network Streamed Video on a Client computer	205
Appendix F	Matlab Programs used in the Research	207
	Matlab code to create histogram from the RTT values	207
	Matlab program to convert histograms into the PMF.....	207
	Matlab program for calculating standard deviation of the number of packets lost	208
	Matlab program to plot the error bar for loss values from all experiments	208
Appendix G	Queue Model for Jitter	209
Appendix H	Recording the MOS	212

List of Figures

Figure 2.1: MOS ratings and their description for QoE measurement [21].....	23
Figure 2.2: IP Network Architecture [37]	25
Figure 2.3: Jitter in a packet network [44].....	27
Figure 2.4: A simple PFIFO network traffic model	28
Figure 2.5: Traffic model based on a distribution.....	29
Figure 2.6: Traffic model based on the Normal distribution	29
Figure 2.7: Traffic model based on the Pareto distribution.....	30
Figure 2.8: A Typical Queueing System.....	31
Figure 2.9: NetEm adds a fixed delay to outgoing data packets.	34
Figure 2.10: Packet delay added using a delay distribution in NetEm.	34
Figure 3.1: Visualising frame rate in a video [94].	38
Figure 3.2: Common video aspect ratios on a screen, along with currently popular TV sizes for each aspect ratio [97]	39
Figure 3.3: A media file unable to play without the right codec.	40
Figure 3.4: The concept of image/video resolution illustrated by Vimeo with their logo [105]	41
Figure 3.5: A YouTube video available in different video resolutions [107].....	42
Figure 3.6: VLC built-in protocols for network streaming	43
Figure 3.7: VLC transcoding options for output streaming.....	44
Figure 3.8: A TFIFO queue that resembles a datagram network where packets go out of order in transmission.....	46
Figure 3.9: A PFIFO queue, which resembles a voice circuit network where packets are in order on the receiving end.....	46
Figure 4.1: Testbed model	51
Figure 4.2: Testbed Improvement - replacement of the RPi with another laptop	52
Figure 4.3: Classifying traffic in Linux, traffic filters' hierarchy [119].	54
Figure 4.4: USB to Ethernet adapter.....	55
Figure 4.5: Testbed upgrade, a more powerful server computer	55
Figure 4.6: VOD streaming with 0.01% packet loss. A participant evaluating QoE and recording MOS.....	56
Figure 4.7: VOD streaming with 0.1% packet loss. A participant evaluating QoE and recording MOS.....	57
Figure 4.8: VOD streaming with 1% packet loss. Evaluation of QoE	57
Figure 4.9: VOD streaming with 5% packet loss. A participant evaluating QoE and recording MOS.....	58
Figure 4.10: VOD streaming with 10% packet loss. A participant evaluating QoE and recording MOS.....	58
Figure 4.11: Proxy server arrangement for experimenting with Internet traffic	61
Figure 4.12: The network configuration of NetEmBox with one Network Interface Card (NIC) left, and network configuration with two NICs after the upgrade, right.....	62
Figure 4.13: NetEmBox-1 upgrade – separate LAN ports for a wired Internet connection and local streaming	62
Figure 4.14: NetEmBox-2	63
Figure 4.15: Wi-Fi hotspot for streaming to mobile phones	64
Figure 4.16: Wireless connectivity with NetEmBox-1.....	65

Figure 4.17: The designed testbed with a more robust Wi-Fi connectivity feature.....	66
Figure 5.1: Preliminary experiments.....	69
Figure 5.2: PLP 0.1% - the RTT values extracted from the first run of the experiment	70
Figure 5.3: Histogram of RTT values extracted from 10,000 transmitted packets	71
Figure 5.4: PLP = 0.1%, the PMF plot of packet delays	72
Figure 5.5: The PMF variable created in Matlab.....	72
Figure 5.6: PLP = 1%, the PMF plot of packet delays	73
Figure 5.7: PLP = 5%, the PMF plot of packet delays	73
Figure 5.8: PLP = 10%, the PMF plot of packet delays	74
Figure 5.9: Experimental run 1 - traffic statistics with 0.01% packet loss	75
Figure 5.10: Experimental run 2 - traffic statistics with 0.01% packet loss	75
Figure 5.11: Experimental run 3 - traffic statistics with 0.01% packet loss	75
Figure 5.12: Experimental run 4 - traffic statistics with 0.01% packet loss	76
Figure 5.13: Experimental run 5 - traffic statistics with 0.01% packet loss	76
Figure 5.14: Error bar plot for the PLP values of 0.01%, 0.1%, 1%, 5%, and 10%.....	84
Figure 5.15: Case 1 - only VOD streaming on the link	84
Figure 5.16: Case 2 - VOD streaming and ping probing simultaneously on the same link	85
Figure 5.17: 11% packet loss with only ping traffic on the link	85
Figure 5.18: 9% packet loss with data traffic from both, ping probing and VOD streaming, on the same link.....	86
Figure 5.19: Pareto delay distribution	87
Figure 5.20: One-sided Normal delay distribution	87
Figure 5.21: Pareto distribution with higher transmitted packets, 100,000	88
Figure 5.22: Normal delay distribution with higher number of transmitted packets, 65,000	88
Figure 5.23: A Normal (Gaussian) distribution	89
Figure 5.24: A Pareto distribution.....	89
Figure 5.25: Derivation of Mean from the standard deviation.....	90
Figure 5.26: Graph to analyse Pareto distribution behaviour in the presence of 10ms mean delay.....	93
Figure 5.27: Graph to analyse Pareto distribution behaviour in the presence of a higher, 100ms mean delay.....	93
Figure 5.28: QoE experimentation on UDP and TCP protocols in the presence of NetEm implemented Jitter	96
Figure 5.29: QoE experimentation on UDP and TCP protocols in the presence of NetEm implemented loss	97
Figure 5.30: UDP and TCP MOS comparison graph with loss and jitter applied simultaneously, jitter = 5ms, and loss is variable	99
Figure 5.31: UDP and TCP MOS comparison graph with loss and jitter applied simultaneously, jitter = 10ms, and loss is variable	99
Figure 5.32: UDP and TCP MOS comparison graph with loss and jitter applied simultaneously, jitter = 80ms, and loss is variable	100
Figure 5.33: UDP and TCP MOS comparison graph with loss and jitter applied simultaneously, jitter = 100ms, and loss is variable	100
Figure 6.1: MOS graph plotted against PLP, MOS recorded by participants from 10-18 age group.....	106
Figure 6.2: MOS vs PLP curve fitting for 10-18 age group, with Exponential function and two number of terms	109

Figure 6.3: MOS graph obtained from a 19-30 years' age group participants	110
Figure 6.4: MOS vs PLP curve fitting for 19-30 age group, with the Exponential function and two number of terms.....	111
Figure 6.5: MOS graph against PLP, MOS collected from 31-45 age group participants	112
Figure 6.6: MOS vs PLP curve fitting with 2 terms for 31-45 age group.....	114
Figure 6.7: MOS and PLP graph, MOS collected from 46-65 age group	115
Figure 6.8: MOS vs PLP (%) curve fitting for 46-65 age group, with fitted 2 term Exponential function .	116
Figure 6.9: MOS graph plotted against PLP, MOS recorded by participants from 65+ age group	118
Figure 6.10: MOS vs PLP (%) curve fitting for 65+ age group, with fitted 2 term Exponential function. .	119
Figure 6.11: Combined MOS graph comparing MOS from different age groups	119
Figure 7.1: Minimum Internet connection speed required to watch YouTube videos [141].	123
Figure 7.2: The bash script in action, a user can input parameters and the NetEm commands are applied in the background.....	124
Figure 7.3: Screenshot of MatLab program in action for calculating YouTube QoE.....	126
Figure 7.4: Google datacentres worldwide [148].	128
Figure 7.5: Geographical view of YouTube cache sever locations showing Primary (P), Secondary (S), and Tertiary (T) locations.....	129
Figure 7.6: YouTube QoE experimentation with NetEm implemented loss.....	130
Figure 7.7: YouTube QoE experimentation, analysis of the time taken to start the video and to fully load the video.	131
Figure 7.8: Speed test after the experiment.....	132
Figure 7.9: Speed test prior to the experiment	132
Figure 7.10: Experimental and analytical MOS for automatic YouTube download quality and RTT of 1ms	134
Figure 7.11: Speed test after the experiment.....	134
Figure 7.12: Speed test prior to the experiment	134
Figure 7.13: Experimental and analytical MOS for automatic YouTube download quality and RTT of 50ms	136
Figure 7.14: Speed test after the experiment.....	136
Figure 7.15: Speed test prior to the experiment	136
Figure 7.16: Experimental and analytical MOS for automatic YouTube download quality and RTT of 100ms.....	138
Figure 7.17: Speed test after the experiment.....	138
Figure 7.18: Speed test prior to the experiment	138
Figure 7.19: Experimental and analytical MOS for automatic YouTube download quality and RTT of 150ms.....	140
Figure 7.20: Mozilla Firefox extension for forcing YouTube video quality to a desired resolution on every reload of the page.....	141
Figure 7.21: Speed test after the experiment.....	141
Figure 7.22: Speed test prior to the experiment	141
Figure 7.23: Experimental and analytical MOS for 720p YouTube download quality and RTT of 1ms	143
Figure 7.24: Speed test after the experiment.....	143
Figure 7.25: Speed test prior to the experiment	143
Figure 7.26: Experimental and analytical MOS for 720p YouTube download quality and RTT of 50ms ..	145
Figure 7.27: Speed test after the experiment.....	145

Figure 7.28: Speed test prior to the experiment	145
Figure 7.29: Experimental and analytical MOS for 720p YouTube download quality and RTT of 100ms	147
Figure 7.30: Speed test after the experiment.....	147
Figure 7.31: Speed test prior to the experiment	147
Figure 7.32: Experimental and analytical MOS for 720p YouTube download quality and RTT of 150ms	149
Figure 7.33: Recommended video bitrates for SDR uploads	150
Figure 7.34: Recommended video bitrates for HDR uploads	150
Figure 8.1: Capacity increase factor for C = 1Gbps and Q=1000	156
Figure 8.2: Capacity increase factor for C = 1Gbps and Q=10000	157
Figure 8.3: Capacity increase factor for C = 20Mbps and Q=1000	157
Figure 8.4: Capacity increase factor for C = 512kbps and Q=1000	158
Figure 8.5: Capacity increase factor for C = 512kbps and Q=10000	158
Figure 8.6: Capacity increase factor for C = 512kbps and Q=64	159
Figure A.1: A successful connection to the VPN	184
Figure A.2: Establishing a remote connection with the lab server (RLPortal).	184
Figure A.3: Old VPN server fails to allow VPN connection.	185
Figure C.1: NetEm interface status – PFIFO queue has been activated along with packet delay and packet loss.	190
Figure D.1: Network connections graphical user interface	191
Figure D.2: Creating a new network connection	191
Figure D.3: Naming the new connection	192
Figure D.4: Assigning an Ethernet interface to a new network connection	192
Figure D.5: Choosing the network configuration for this connection	193
Figure D.6: Automatic connectivity option	193
Figure D.7: Accessing all saved network connections	194
Figure D.8: Newly created connection seen in the connections' list.....	194
Figure D.9: Connecting via a network connection	195
Figure D.10: Connection establishment confirmation.....	195
Figure D.11: VLC Player opened on the server computer on the NetEmBox testbed	196
Figure D.12: Streaming option opens 'Open Media' window.....	196
Figure D.13: Streaming output options	197
Figure D.14: Choosing the streaming protocol	197
Figure D.15: Providing network details for the stream	198
Figure D.16: Activating the transcoding.....	198
Figure D.17: Streaming all the added video streams.....	199
Figure D.18: Streaming starts	199
Figure D.19: The option to choose a lowest latency from VLC preferences.....	200
Figure E.1: New connection on the client computer	201
Figure E.2: A meaningful connection name for this connecting to the server	201
Figure E.3: The Ethernet interface that would be used for this connection	202
Figure E.4: Client IP address allocation method	202
Figure E.5: Assigning a manual IP address to client computer for server connection	203
Figure E.6: The option for connecting to the server automatically whenever possible.....	203
Figure E.7: Newly created connection, NetEmBox	204
Figure E.8: To connect to the server	204

Figure E.9: Client successfully connected to the server 204
Figure E.10: Opening a network stream in VLC on the client computer 205
Figure E.11: Providing the steaming protocol and the IP address for VOD 206
Figure E.12: The Client computer successfully opens the network streaming..... 206
Figure G.1: Bespoke Packet Delay Distribution 210
Figure G.2: Delay Distribution..... 210
Figure H.1: Survey sheet used in this research..... 212

List of Tables

Table 2.1: A comparison of Dummynet, NISTNet, and NetEm [81].....	33
Table 5.1: Experimental packet loss from 10 runs with 0.01% packet loss.....	76
Table 5.2: Experimental packet loss from 10 runs with 0.1% packet loss.....	79
Table 5.3: Experimental packet loss from 10 runs with 1% packet loss.....	80
Table 5.4: Experimental packet loss from 10 runs with 5% packet loss.....	81
Table 5.5: Experimental packet loss from 10 runs with 10% packet loss.....	82
Table 5.6: Packet loss statistics along with Confidence Intervals (99% confidence).....	83
Table 5.7: Mean calculated for each standard deviation value.....	91
Table 5.8: Jitter only experimental results with MOS values.....	95
Table 5.9: Experimentation with VOD using UDP and TCP protocols with loss only implementation.....	96
Table 5.10: MOS experimentation with UDP and TCP protocols having implemented both loss and jitter simultaneously.....	98
Table 6.1: Number of participants from each age group.....	105
Table 6.2: MOS recorded by participants from 10-18 age group.....	106
Table 6.3: MOS recorded by participants in 19-30 years' age group.....	109
Table 6.4: MOS collected from participants from 31-45 age group.....	112
Table 6.5: MOS collected from participants in 46-65 age group.....	114
Table 6.6: MOS by participants in 65+ age group.....	117
Table 7.1: YouTube QoE experimentation with loss in the presence of buffering.....	130
Table 7.2: Experiment results when RTT = 1ms.....	133
Table 7.3: Experiment results when RTT = 50ms.....	135
Table 7.4: Experiment results when RTT = 100ms.....	137
Table 7.5: Experiment results when RTT = 150ms.....	139
Table 7.6: Experiment results when downloading at 720p YouTube quality for RTT = 1ms.....	142
Table 7.7: Experiment results when downloading at 720p YouTube quality for RTT = 50ms.....	144
Table 7.8: Experiment results when downloading at 720p YouTube quality for RTT = 100ms.....	146
Table 7.9: Experiment results when downloading at 720p YouTube quality for RTT = 150ms.....	148

Glossary

ACR	Absolute Category Rating
DVD	Digital Video Disk
FCFS	First Come First Served
FIFO	First in First Out
HD	High Definition
HDR	High Definition Resolution
HDTV	High Definition TV
IP	Internet Protocol
LIFO	Last in First Out
MOS	Mean Opinion Score
MPEG	Moving Picture Expert Group
NetEm	Network Emulator
PDF	Probability Distribution Function
PFIFO	Packet First in First Out
PLP	Packet Loss Probability
PMF	Probability Mass Function
QoE	Quality of Experience
QoS	Quality of Service
RTT	Round Trip Time
SD	Standard Definition
SDR	Standard Definition Resolution
SLA	Service Level Agreement
TCP	Transmission Control Protocol
TFIFO	Time First in First Out
UDP	User Datagram Protocol
VCD	Video Compact Disk
VOD	Video on Demand
WMA	Windows Media Audio

Chapter 1 Introduction

In packet networks, the metrics that allow the measurement of network quality are delay, loss, and jitter. These are the objective measures of network performance, which allow network providers to guarantee a level of service to their customers. In telecommunications, these metrics are referred to as the Quality of Service (QoS). QoS is a network-based measurement that outlines the quality of the network that customers are promised from a network provider such as the average up-time in a month, which most of the companies promise to be over 99% (and even 99.9%) [1] [2] [3] [4] [5], or the network coverage allowing the use a particular service.

Quality of Experience (QoE), however, is a subjective measurement of network quality meaning that it corresponds to the network quality measurements taken from a customer's perspective. Although QoS promises a certain level of offered service, the service that reaches a customer on the other end is not always as good as promised by the service provider with their QoS. QoE, then, covers what QoS does not cater for. As customer satisfaction is the key to having a prosperous business, hence, study of QoE can play a significant role in outlining the issues that may require network provider's attention in order to provide a satisfactory service to their customers.

This thesis presents that a) there has been a knowledge gap that has been created by inadequate emulation tools, which do not accurately reproduce real networking conditions, b) a bias in the QoS to QoE mapping results previously reported in the literature and elsewhere, caused by a failure to use a full age range of subjects in QoS-QoE mapping experiments.

A very common way of measuring QoE is through the Mean Opinion Score (MOS). MOS has a scale of 5 to 1 (Excellent to Poor respectively) for scoring video/audio quality of applications¹ for example video or

¹ Although MOS can be used to assess the quality of a wide range of traffic across a network; however, in this research the MOS evaluation is primarily conducted on video traffic unless otherwise specified.

audio network streaming, or real-time applications such as Skype. In the case of a video, the video is streamed over the network and shown to the research participants. They are then asked to give a score from 5 to 1 based on the quality of the video they perceived.

In this research, QoE in packet networks was studied. The analysis was conducted on the QoE for video traffic under a range of scenarios to investigate loss and delays in such a traffic transmission whilst analysing the factors that may compromise the QoE in video applications.

1.1 Motivation

It is now well accepted by many researchers, including [6] [7], that user QoE measures are a far better representation of network performance as experienced by the user than are raw QoS measures (e.g. packet loss probability, mean end-to-end packet delay, delay jitter). However, evaluating QoE first requires accurate measurement of QoS metrics [8] [9] [10].

Prior research has shown that poorly designed measurements can easily lead to results that are very inaccurate [11] [12], e.g. 2 orders of magnitude in error for measured packet loss probability. When translated through QoS to QoE mapping this could be a difference between “good” and “highly unsatisfactory” user quality.

1.2 Goal

The objective of this PhD research was to study QoE by analysing video traffic. This then involved designing the right, innovative toolset to be able to effectively conduct the research. A state-of-the-art testbed was designed to fulfil the research needs. For details refer to Chapter 4 on page 47. The aim was to investigate the video QoE in depth by mimicking common network scenarios. This included analysing video traffic on a local network and also video traffic from across the Internet such as YouTube.

Although previous research has been conducted on video QoE, the objective of this research was to go beyond the limitations experienced by researchers previously. These limitations were mainly not being able to replicate the real-world network patterns within laboratory environment, which consequently gave the researchers an insight into the general behaviour of networks under various constraints but still did not relate closely to the real-world network traffic. Therefore, the main objective of this research was to go beyond the conventional ways of QoE study, such as interpreting the mathematical results that reflect various network scenarios, and experimenting on real network traffic whilst seeing the network artefacts in reality after introducing network imperfections. This would of course involve users' opinion based on the video quality they experience as a result of a network problem added on the network artificially. The experimental setup was such that users were shown a video as if they were sitting in their living room streaming from Netflix or YouTube.

Meaningful results collected from this realistic user-involved research allowed me to highlight the factors that, having worked on and improved, network service providers could provide a better service to their customers which would then lead to a prosperous business.

1.3 Skills and Tools Assessment

In order to successfully complete this research, a certain level of skill was required. Particular software and hardware tools were also required to conduct the research.

It is worth mentioning that required skills/tools assessment was not a one-off task. Along with the progress of the research, further technical skills were acquired. Similarly, more software/hardware tools were added to the required list. Initially, the required skillset and tools were different. For more on initial planning see Appendix B on page 188.

With the arisen need of designing a new testbed, a range of new skills was required equally for the design and the use this new testbed. Some of these skills, which were acquired in order to adapt to the changing circumstances, are listed below:

- Linux administration
 - Operating System installations and maintenance
 - Ability to install and remove required packages
 - Command line familiarity
 - Familiarity with the filesystem and desktop environment
 - Ability to write scripts to simplify the desired tasks
 - Ability to change and add Linux kernel configurations
- Network management knowledge
- Ability to research online, and solve problems

The newly required tools included a laptop to run Ubuntu for the initial testbed design, and a Raspberry Pi.

The initial design of the testbed incorporated a laptop, later, however, a desktop computer (later named NetEmBox-1) was acquired. For more on the testbed design, refer to Chapter 4 on page 47.

1.4 Design of Experiments

When studying the effect on QoE of packet loss and jitter (whether as standalone metrics or a combination), a different level of degradation on the link was added without notifying the experimental subjects what was the amounts implemented. Furthermore, the packet loss/jitter was varied in a random order. For instance, when showing a video to experimental subjects with a range of added packet loss levels: 0.01%, 0.1%, 1%, 5%, and 10%, the packet loss was applied in such a way that it was not increasing or decreasing gradually with each experimental run. This random application of packet loss ensured that, whilst recording their MOS, experimental subjects did not merely follow the trend knowing the packet loss was higher or lower than the previous run of the experiment. This then guaranteed that experimental subjects recorded their MOS based on the actual quality of the video they experienced rather than improving or worsening their MOS rating gradually following the decrease or increase in the packet loss. The same procedure was followed in the experiments to study delay/jitter.

In previous research, the population of experimental subjects was biased, as they all tended to be from academia, either PhD students or PostDoc researchers, and so falling in an average age range of 21-30. As the user population used in this study covers age groups 10-65+, this research is not biased on participants' age. In addition to being from different age groups, users were also of different gender and

ethnicity groups. The results could be potentially biased if viewed on the basis of geographical location of users since all research participants were from London.

During this research, young people from 10-18 years age group attended the QoE evaluation sessions with the permission and in the presence of their parent(s).

1.5 Outline of the Thesis

Chapter 1	This thesis is laid out in such a way that the first Chapter lays the basis of this research work with an introduction.
Chapter 2	The literature is reviewed.
Chapter 3	A description of associated technical concepts and terms is given.
Chapter 4	Research methodology is laid out where the need for an alternative testbed is discussed and the testbed design phases are outlined.
Chapter 5	The newly designed testbed is utilised to proceed with the experimentation – the results are shown therein. In this chapter, experimentation has been done on video traffic with artificially added packet delay and loss. The Pareto distribution in NetEm along with a review of the Normal distribution is also debated. Furthermore, the use of custom delay distributions in NetEm is discussed, and bespoke delay distribution tables is used to study jitter. Additionally, video traffic has been studied over network protocols: UDP and TCP.
Chapter 6	QoE and MOS is discussed and MOS scores collected from various age groups are provided and debated.
Chapter 7	The study conducted to evaluate the QoE of the YouTube traffic is outlined. The progressive-download video is studied in both cases: with automatically adjusted download quality, and with a pre-set 720p download video quality.
Chapter 8	The significance of QoE evaluation on VOD across a range of age groups is discussed. Out of all of the users from different age group, it was outlined which group of participants is most demanding in terms of the service quality of a given service.
Chapter 9	This chapter concludes the findings from this research, along with possible future work. A conclusion is then made towards the commercial importance of this research.

After the references, appendices list the material that was not included in the main part of the thesis to maintain the coherency.

Chapter 2 Literature Review

2.1 Chapter Introduction

To help one understand the importance of this research work this chapter lays the basis for this research with an introduction to Quality of Experience (QoE) of any provided service. QoE is a measure of the actual quality of the promised service experienced by the end user. An outline is then described how the QoE is measured. Mean Opinion Score (MOS), as a way of evaluating QoE, is then explored, and some of the network researchers and companies that take this approach for QoE evaluation are discussed. An overview of QoE research in the area of network is then given. Quality of Service (QoS) as metrics is then discussed, and the effect link congestion on the QoE is explored. This then leads to greater clarity on the necessity of network measurements, involving delay, jitter, and packet loss. Next, types of network measurement are described. Then queueing theory, a basic concept to understand network systems, is discussed along with application of queueing systems.

The use of network emulators in the literature in QoE research is then reviewed, and a discussion of why NetEm was chosen for this research. Major known flaws in NetEm are also highlighted, and a discussion of how these particular flaws have been overcome to take advantage of the valuable core features of NetEm; this is detailed in Chapter 4, on page 48. Then a literature review is provided on the approach taken to record MOS that involved participants of different age groups.

2.2 Quality of Experience (QoE)

QoE is the quality of service that is perceived by the user for many services offered today such as VoIP, IPTV, and even web browsing. With the help of QoE knowledge in such a competitive market, service providers can make useful decisions in providing a satisfactory service whilst being able to monitor the provided service to keep the customer interested [13]. QoE provides a numerical metric that tells us the overall satisfaction that a customer has with their service provider or vender. Traditional Quality of Service (QoS) only refers to the network performance. However, QoE also focuses on the facility or

service quality such as the cost, coverage, usability, and reliability [6] [14] [15]. Various people have defined QoE differently. According to TechTarget.com [16] (direct quote):

“Quality of Experience (QoE or QoX) is a measure of the overall level of customer satisfaction with a vendor. QoE is related to but differs from Quality of Service (QoS), which embodies the notion that hardware and software characteristics can be measured, improved and perhaps guaranteed. In contrast, QoE expresses user satisfaction both objectively and subjectively. The QoE paradigm can be applied to any consumer-related business or service. It is often used in information technology (IT) and consumer electronics.”

As QoE is user-centric, it has a direct impact on the business. AWTG outline this on their website [17] (direct quote):

“In terms of the network operator, the quality of experience is the ultimate measure of how subscribers perceive the performance of the network and its services. A poor user experience will result in dissatisfied customers, leading to a poor market perception and ultimately weakening of the brand. Improving the QoE will improve customer loyalty and enable the network operator to maintain a competitive edge.”

Although different people from different backgrounds have described QoE with slightly different terms, the main concept behind all of these definitions from several researchers is the same – making it a subjective measurement by involving the end user. QoE does not replace the concept of QoS, however, it makes the QoS more tailored and exclusive to what service the user perceives. Hence, QoE still studies the end-to-end communication link, however focussing more on the human interaction rather than the technical aspects of the communication link. This would make sense from the point of view of a human who has to use the services on a daily basis to fulfil their needs. They will not bother much about what goes on in terms of the technical steps in order to process their request, but rather, the end users are more concerned about if they are receiving the expected service to a satisfactory level.

From the history of QoS, the main concern was to understand hence improve the performance of the network to provide a good level of service to the user. Then the improvement of the concept of the QoS included providing a service agreement to the customer, and abiding by the promised metrics of service. Moving on, the new form of QoS included a better service to selected applications and services whilst including the metrics such as reliability, throughput, loss probability, delay and jitter [18]. It was then observed that QoS has a great impact on the user of the service, which led to focussing more on the technical performance parameters instead of the service quality itself. With this approach in focus, it was soon realised that examining these performance metrics alone still does not reflect the actual performance of the network considering the needs of the end user. Therefore, to fill this gap in service monitoring, it was decided the perception by someone was required to assess the service performance accurately. The most commonly used scale in the subjective for voice and video QoE assessment is the ACR. In this type of subjective test, a number of people taking the test denote their experiences on a scale of 1 to 5, where 1 represents the “bad” level and 5 denotes “excellent” level of service [19]. The average of the obtained scores is called the MOS, which is discussed further in the following Section.

2.3 Mean Opinion Score (MOS)

MOS is a means to carry out a subjective testing for evaluating how an end user would receive or experience an offered service. In general, quality of audio or video transmission is expressed by the

qualitative descriptions such as ‘very good’, ‘ok’, or ‘very bad’. However, to express this audio/video quality mathematically, MOS can be used which allows us to describe the quality of such services with numbers corresponding to, for example, an ‘excellent’ or ‘unacceptable’ quality.

The range and type of the experiences the users were put through have a direct impact on the achieved MOS. This results in the MOS values only to be compared in the case of the conditions being the same [20]. The table in the Figure 2.1 gives the most commonly used MOS rating and their descriptions along with their impairment.

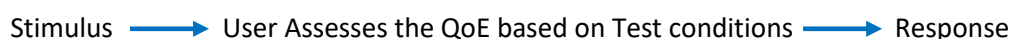
Scores	Quality	Impairment
5	Excellent	Imperceptible
4	Good	Perceptible, but not annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Bad	Very annoying, unwatchable

Figure 2.1: MOS ratings and their description for QoE measurement [21]

This new concept of network assessment led to the decision that a service-user himself / herself would be well suited to carry out the judgement on the service performance and decide the quality of the content delivered. This was then named as Perceived QoS or end-user QoS. With the passage of time, it became clear that the involvement of the end user was vital in ensuring the delivery of a good level of promised service. The competition in the industry for service providers gave more emphasis on the customer, the end user of the service. The opinion of the end user on the experience of their acquired service then became so important that for a successful business, the service vendor reshaped the concept of the perceived QoS whilst centralising the end user in the assessment of the quality of service. This new approach of assessment of the service quality is now called the QoE, making the end-user the subject who experiences the level of the services provided [22].

There has been a great interest into researching the QoE across a multidisciplinary field. The research focuses on assessing the quality of a provided service in a user centric fashion. As the QoE is subjective regarding the complex human quality perception, QoE does not have any precise boundaries or a generally agreed definition. As a result, the QoE measurement has become a challenge.

The above described method of evaluating QoE, also known as the MOS, is referred to as the subjective QoE assessment. The table in Figure 2.1 emphasises the fact that in subjective QoE measurement a user is involved and is provided with predefined ratings to express their opinion on QoE. The provided test conditions are also referred to as Stimulus. The user who experiences the service and records the feedback based on their measures (also known as the response, or the subjective measures). A simple model of Subjective QoE assessment is given below.



Another type of QoE assessment is called objective QoE prediction. Unlike the Subjective QoE assessment, the Objective QoE prediction employs analytical/statistical models and converts the input parameters into estimated QoE. Objective QoE prediction typically measures properties such as task performance, and behaviour. The model for this type of QoE measurement is as follows.

Input \longrightarrow QoE = A formula that utilises input parameters \longrightarrow Output

MOS has been used to evaluate QoE of audio and video by various companies including [23] [19] [24] and independent researchers [25] [26] [27] who are merely a few to mention.

2.4 QoE Research on Network Traffic

QoE has been a major recent focus of research for researchers. Studies have been conducted on wide range of network traffic, in particular: VoIP QoE, real time video streaming QoE, and TCP video QoE. Common factors that have a vital impact on the QoE are the packet loss and packet delay and jitter. Therefore, researchers pay particular attention to studying packet delay and loss probabilities to these metrics for a particular type of traffic in a given network. For instance, [28] provides a packet loss probability analysis in wireless sensor networks, and overviews the causes of the packet loss in such networks.

With a rise in video-based streaming services, many researchers have developed an interest in studying the QoE of video traffic. For instance, QoE quality metrics of video streaming over wireless networks have been studied in [29] with an aim to explore the ways to improve QoE in existing and new streaming services. In [30] the authors have addressed the issue that affects the QoE by the delaying of random packets in the wireless link, which leads to video playback interruptions. Thus, they study the interruption probability in wireless video streaming. Real time estimation of QoE has also been performed by combining subjective and objective assessments by using a no-reference and low-complexity method in [31].

In [32] QoE study on video traffic has been done and researchers have focussed on the HTTP-based video content delivery on a shared wired network. This technology allows the video bit-rate to adapt to the link condition dynamically. The researchers in [32] have proposed a resource management framework, which allows mobile operators to maintain QoE by having control over the resource allocation over a shared network.

2.5 QoE in the Presence of Congestion (QoS as Metrics)

As IP traffic is of bursty nature [33] [34] [35], there may be a very small number of data packets in a queue at a certain interval of time, and the next moment a large number of packets come in simultaneously. This behaviour of the IP traffic has a direct effect on the QoE since it can lead to the buffer being very full, and ultimately a significant number of packets being dropped, delayed, or subject to jitter. This would be due to the arrival rate being greater than the service rate, as well as a finite capacity of the queueing buffer in the router.

As described above, as the load gets close to 100%, the queue server struggles to maintain the service, in other words, QoS degrades. However, having activated delay, jitter, and loss through NetEm on the

router QoE can be degraded well before the load reaches 100%, as has been confirmed in [36]. This is because the traffic congestion results in packet loss and network delays even at lower loads.

2.5.1 Congestion Control by the Bandwidth Method

The congestion in the router or the network can be controlled by increasing the bandwidth of the link or the network allowing more data packets from various applications to be transmitted simultaneously. This will ultimately reduce the queues, causing less delay and loss of information.

The overall structure of modern networks is such that the routers that route the data packets from one end to the other are connected through a physical layer of fibre optics and copper. The whole network architecture is divided into three sections allowing an easy deployment and installation of the network. This division can also be thought of three layers; the first layer consists of the provider network (P) which consists of Core routers. The provider network then connects to the part of the network known as the Provider Edge (PE), which as the name suggests is the edge of the Provider network (P). The PE then connects to the third layer in the network called the Customer Equipment (CE), which is the part of the network setup or installation at customer premises.

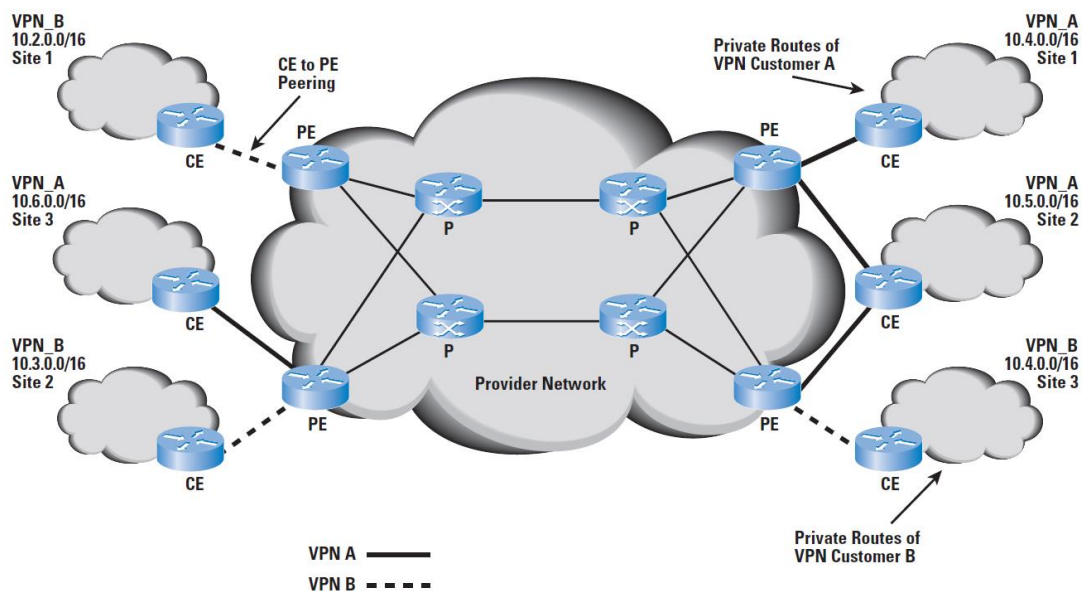


Figure 2.2: IP Network Architecture [37]

As for the capacity of these three divisions of networks, the P as its needs require, has high capacity links. The capacity of these links in developed countries is usually 40Gbit/s to 100 Gbit/s. The network links between the P and PE are typically between 10Gbit/s and 40Gbit/s. Whereas the CE to PE connections have a capacity in the range of 1 Gbit/s to 10Gbit/s.

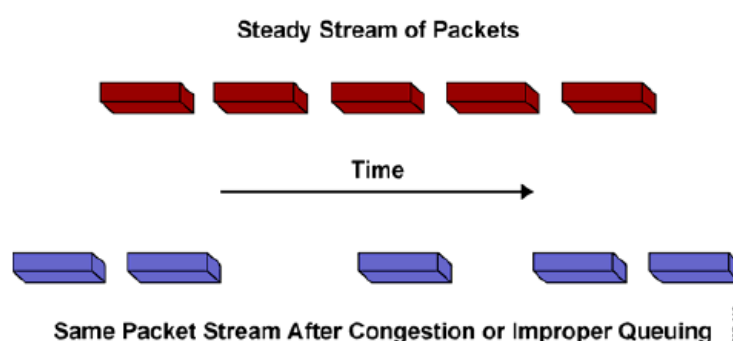
Although increasing network capacity does improve the network performance, it is expensive and can introduce network down times whilst in the upgrade stages, which can lead to poor QoE to the customers. Nevertheless, to cope with the increasing demand of modern applications on networks, it is vital to be able to transmit data from a particular range of applications with a minimum delay utilising the network limited capacity.

2.6 Network Measurements Involving Delay, Jitter, and Loss

In real world packet networks, the common factors related to the network performance are the packet loss and delay. The average packet loss is the ratio of the packets lost with the packets transmitted. Whereas the average length of time for which the buffer is in an overflow state over a certain length of time (whether 24 hours or one busy hour) is called the overflow probability. The mean delay is the average packet delay between two given points in the network. It is worth noting that this calculation is an average, meaning that the overall performance of the network for a given month would be calculated to estimate the mean delay. In a given month, there may be times during a single day when the network performance is poor due to a larger number of service users. However, it is a significant possibility that at night when there will be comparatively fewer users, less delay is experienced. Therefore, on average, the network performance will generally be satisfactory. Hence, for a meaningful test of network performance where the delay² and loss noted give a good representation of the network condition, the testing should be done over a busy hour³ [38]. This busy hour is modelled in my choice of delay distribution and loss probability used in this thesis. Different levels of jitter represent busyness and congestion on different levels in that busy hour.

Delay can be measured using passive and active measurements along with bandwidth and packet loss. Not only does this help measure these the performance of the network, but it can also be very useful in predicting traffic behaviour. In [39], a detailed study is represented for understanding and controlling end-to-end delays on a large-scale IP network. In [40], PDF of delay data is used to interpret the network load status. Additionally, it is also shown that the Pareto distribution can be used to approximate statistical end-to-end delay in a network [41] [42]. In [43], an algorithm, based on the Pareto distribution has been designed to smooth the delay jitter in Cyber-Physical Systems (CPSs). CPSs represent a system which tightly integrates computation, communication and physical processes, and require doing the right thing at the right time [43].

Jitter is the delay variation between the individual packets in a stream of packets. Although the source in a network transmits packets in a steady stream. By the time the packets reach their destination device travelling across the network, all packets do not have the same even spacing between them as it was at the time of initial transmission. Packets get spaced out or clumped together with a random delay among them, which in network terminology is known as jitter. The figure below visualises the concept of jitter.



² In VoIP services the acceptable delay is up to 177 ms, exceeding this threshold will affect the QoE [198].

³ Busy hour refers to a 60-minute window within a 24-hour period when the traffic load on the network is at a maximum [11].

Figure 2.3: Jitter in a packet network [44]

Major causes of jitter are considered to be network congestion, configuration errors, and TFIFO queueing discipline for packet queueing [44]. Although traditionally for the design of the IP networks, typically, the average delay and loss have been focussed on by the network service providers as key metrics. Jitter, if not dealt with effectively, can have a significant impact on real time traffic such as VoIP, and interactive services and video streaming [45] [46] [47] [48].

Therefore, various researchers have taken a range of approaches to calculate and minimise the end-to-end jitter. For instance, in [49] an analytical approximation is shown to calculate the end-to-end delay jitter in periodic traffic with a constant packet size. For calculating jitter in a single queue with Poisson traffic, [50] has proposed fairly straight forward formulas. In [51], delay and jitter for voice traffic have been analysed, and it is suggested that Priority Queueing could significantly improve the voice quality by minimising delay and jitter. More sophisticated approaches have also been employed to preserve the speech quality in VoIP traffic as shown in [52] where the concepts of network fingerprint, data pre-processing, and neural networks are utilised. In [53] and [54], researchers propose a modified version of E-model to predict jitter in a transmission channel.

2.7 Network Measurement Types

An SLA is an agreement between a customer and the service provider, which states the terms and conditions of the provided service and network characteristics such as guaranteed average network up time over a month and latency. If the service provider fails to provide the promised service they compensate the customer. BT promises the service availability of 100% [1]. It is understandable that an SLA is of vital importance to both the customer and the service provider. An SLA, in other words, is to stay within the agreed terms of performance metrics. This emphasises the significance of measuring the performance metrics of the network and keeping a close eye on them.

One of the ways of carrying out network measurements is probing, where measurements are done by taking samples at specified time intervals. Depending on the type of measurement being used, the pattern of the sampling may have an effect on the performance of the network. Once the measurements have been taken, the probes are assessed against the traffic metrics to conclude the packet delay and losses [55] [56].

There are two main types of network probes, active and passive probes. Passive probes do not disturb the flow of the traffic and are done by observing the normal traffic [57]. Cisco Service Agent is an example of passive probing [58]. This type of measurement does not require any dedicated hardware setup, and the Cisco routers can perform the measurements given that traffic is present on the link where the measurement is required.

Active probing on the other hand requires additional packets to be sent out into the network, which clearly requires an additional setup, which could even just be the software to produce and monitor these packets. The results obtained from the probing depend on how the probing packets interact with the traffic in the network. A smaller time gap between probing packets gives better insight into assessing the traffic behaviour. As such, generally, a larger number of probing packets may give more reliable results. However, this is only true to certain point and producing more than a specific number of

probing packets may affect the flow of the data traffic reducing the accuracy of measurement results. This behaviour of active probing has also been outlined in [59]. Therefore, a high frequency of the probing sampling can have a direct impact on the overall performance of the network. To overcome this challenge, various ways have been described in [56] for doing active probing measurements. These techniques of active probing are Simple Random, Stratified Random Sampling, Systematic Sampling, and Batch Poisson Sampling. These different types of probing describe different patterns of probing producing probe packets and sending them out into the network for assessment purpose. In this thesis probing refers to active probing.

2.8 Network Traffic Patterns

Common patterns of network traffic seen in real world networks are shown in Figure 2.4 through to Figure 2.7.

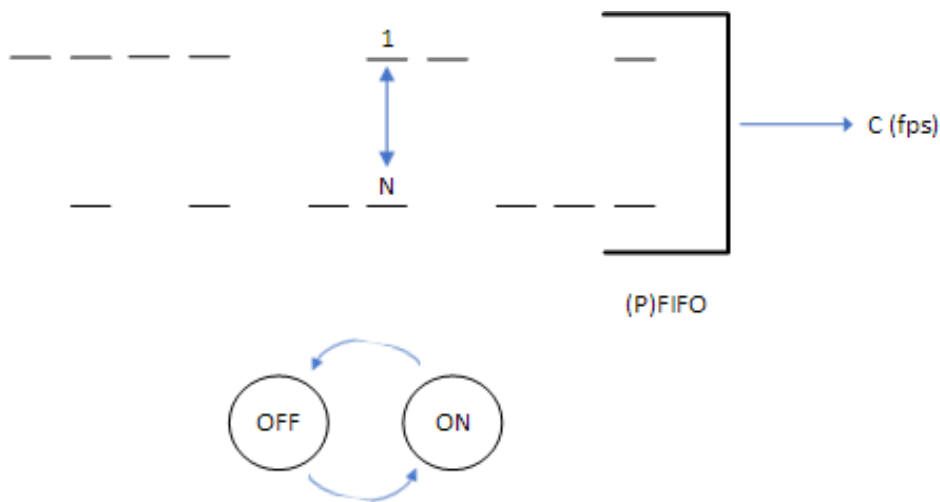


Figure 2.4: A simple PFIFO network traffic model

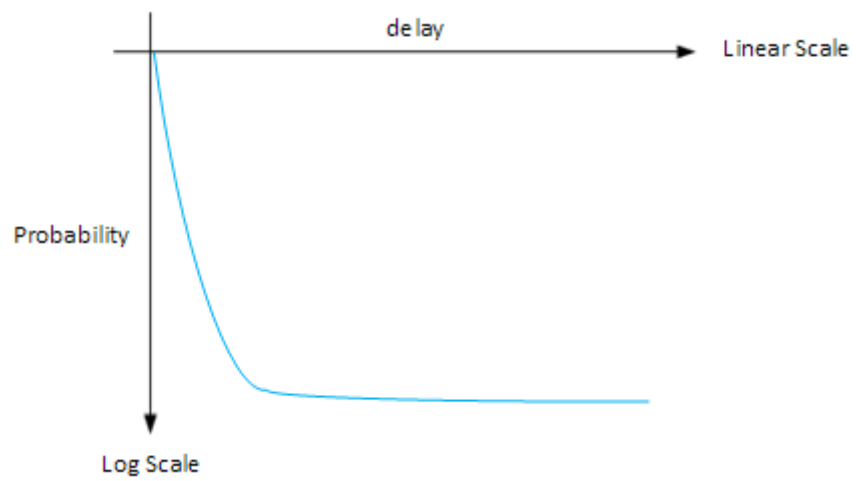


Figure 2.5: Traffic model based on a distribution

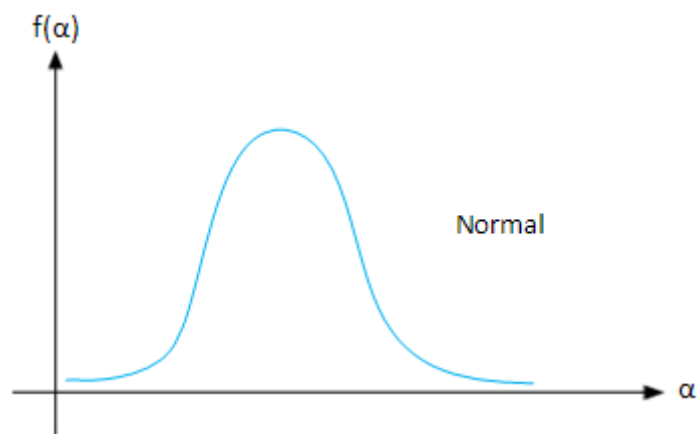


Figure 2.6: Traffic model based on the Normal distribution

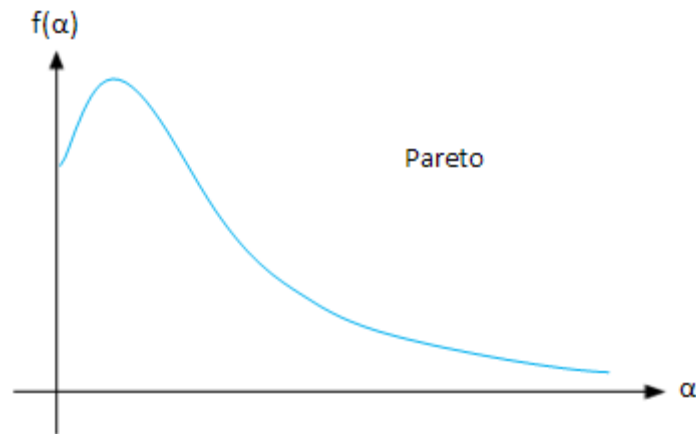


Figure 2.7: Traffic model based on the Pareto distribution

NetEm, a network emulator discussed in detail in Section 2.12 on page 33, allows modelling of these traffic patterns to mimic behaviour of realistic networks.

2.9 Queueing Theory

Queues can be found in various areas and aspects of daily life such as at supermarket checkout points, traffic lights, waiting at passport control and so on. These are the queues we acknowledge encountering every day. However, there are some other queueing mechanisms that cannot necessarily be seen. In fact, we do not even notice them taking place in various computerised structures. An example is in the streaming of a video to a personal device, where the video is delivered in the form of data packets, each going through various routers whilst queueing up for the onward transmission, where they will finally reach the end device.

A similar example is of a web server, which processes a request to open a website, e.g. www.bing.com. The loaded website seen on the computer screen is a result of the content sent by the web server in the form of data packets. These data packets travel through a series of queueing systems across various networks, and then finally reach the computer through the Network Interface Card. Further to this, the web request is not fulfilled by one element of a network only, but it involves various subsystems working to do a specific job to deliver the information requested. At each of these subsystems, there is a queue of requests sent by other users from all over the Internet. Consequently, here, there exists a queueing system too. However, in order to visualize this kind of queueing system the relevant software and hardware tools are required. Nevertheless, there are various services that we receive with the use of queueing mechanisms, and sometimes, without even realizing it [60] [61].

Queueing theory is the mathematical examination of waiting lines and processing units in various systems. Queueing theory helps us to mathematically model the design characteristics of various networks, assessing parameters such as link bandwidth, delay estimation, the number of buffers and their capacities, blocking and dropping probability. A typical network may have customers (in this case packets) arriving, waiting in the queue then being served. In our case of VoIP, modelling and analysing the results allows us to evaluate the performance of the system under design consideration.

2.10 Queueing Systems

In queueing systems, the arriving calls, messages or tasks are denoted as ‘customers’, whereas the specific system mechanisms that are in place to provide service to these customers are known as ‘servers’. Queueing systems have an important role in modelling and performance analysis, particularly for telecommunication and computer systems. These systems can range from a very simple to very complex incorporating a network of queues, such as those in communication and operating systems. The number of servers in a queueing system can be one or more depending on the nature of the system and the desired functionality. If the arriving customers find the server busy they have to queue up to receive the service. For example, in the telecommunications field, the ‘servers’ could be routers, computer processors, as well as the web servers with back-end processes. Hence the ‘customers’ in such queueing systems could be users, packets, and web requests [62] [63].

A queueing system can be described in terms of the arrival pattern, the nature of the service provided, the number of the servers in the system that can provide a service simultaneously, and the capacity of the system. The buffer length in real systems is always finite, although in the analysis it is often assumed to be infinite to study the queueing behaviour. The arriving packets at the input of the system are measured by the metric known as the mean arrival rate, λ , which is the average number of packets arriving per unit time.

A queueing system typically has the following structure;

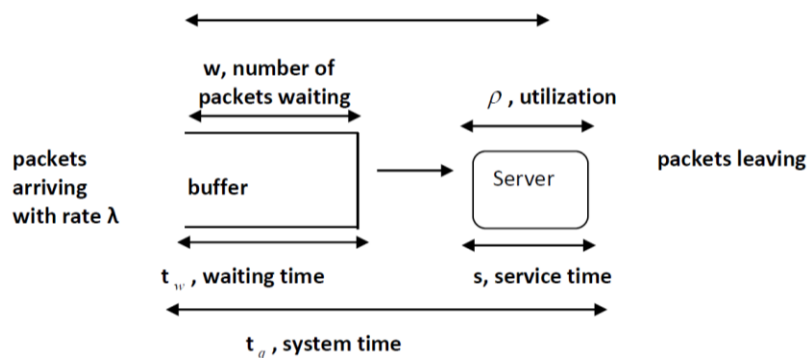


Figure 2.8: A Typical Queueing System

Queueing systems can be characterized by different measurement parameters in order to model specific systems. For example, the arrival process can be based on either inter-arrival times or group arrivals depending on the type of the system under examination. Similarly, the number of queues and the queue servers also vary. Some other features also vary depending on a particular requirement, such as the service disciplines (FIFO), or priority scheduling [62].

The total traffic arrived and fed into the system is known as the ‘load’, and is denoted with ρ . This term has great importance in the field of queueing theory. To determine the load, the mean arrival rate is multiplied with the average service time:

$$\text{Load} = \text{Mean Arrival Rate} \times \text{Average Service Time} \quad (2.1)$$

Since the average service time is defined as $1/\mu$, hence the load becomes:

$$\rho = \frac{\lambda}{\mu} \quad (2.2)$$

Where:

ρ is the load;

μ is the service rate;

λ the mean arrival rate.

It is worth noting that ρ is dimensionless. For the system to be stable ρ should always be less than 1. Conversely, if ρ is equal to or greater than 1, the server will not be able to maintain the service and the queue length will grow to overflow.

Furthermore, it should be noted that according to the above formula, if the arrival rate λ and the service rate μ are equal then the load will be 1, indicating that the system will become unstable. This emphasises the fact that the service rate, μ , should always be greater than the mean arrival rate, λ , for a stable system [64].

This makes sense if one considers the analogy of a supermarket. If, at a till at a supermarket, the cashier is serving at a higher rate than the rate of customer arrivals in the queue then it will allow quick transactions. The opposite is true also: if the cashier is too slow to process the customers as they arrive, transactions will be slow and the queue will grow.

The study of various queue models is useful in visualising the queueing systems in order to get more insight into their characteristics. In queueing theory, these different queue models can be used to model systems that have certain behaviours or arrangements. Some widely used queues for the design and analysis of server-based queueing systems are the $G/G/k^4$, $M/G/k$, $M/D/k$, $M/M/k$, where in each case, G expresses General distribution, M refers to Markovian, and 'k' denotes the number of queue servers.

2.11 The Use of Network Emulation Tools

Many emulation tools have been used to study several types of networks by a number of researchers. In [65] [66] [67] [68] researchers discussed network emulators that are suitable for different network sizes and natures to meet specific needs. In [69], the three most common network emulators: Dummynet, NISTNet, and NetEm have been discussed along with their pros and cons; and a feature comparison among these three production-quality network emulation tools is carried out.

⁴ The term follows the Kendall's notation, which has the following form:
Packet Inter-arrival times distribution / Service time distribution / The number of queue servers.

Table 2.1:A comparison of DummyNet, NISTNet, and NetEm [69]

	DummyNet	NISTNet	NetEm
<i>Availability</i>	Included in FreeBSD	Available for Linux 2.4 and 2.6 (<2.6.14), patch available for more recent versions	Included in Linux 2.6
<i>Time resolution</i>	System clock (up to 10 KHz)	Real time clock	System clock (up to 1 KHz) or high resolution timers
<i>Interception point</i>	Input and output	Input only	Output only
<i>Latency</i>	Yes, constant value	Yes, with optionally correlated jitter following uniform, normal, Pareto distributions	Yes, with optionally correlated jitter following uniform, normal, Pareto distributions
<i>BW limitation</i>	Yes, delay to add to packets is computed when they enter DummyNet	Yes, delay to add to packets is computed when they enter NISTNet	Yes, using the Token Bucket Filter from TC
<i>Packet drop</i>	Yes, but without correlation	Yes, optionally correlated	Yes, optionally correlated
<i>Packet reordering</i>	No	Yes, optionally correlated	Yes, optionally correlated
<i>Packet duplication</i>	No	Yes, optionally correlated	Yes, optionally correlated
<i>Packet corruption</i>	No	Yes, optionally correlated	Yes, optionally correlated

Although there were many network simulation and emulation tools to choose from including the ones shown in [69], it was discovered that the use of NetEm in this testbed would offer higher flexibility, in particular, with the research plans to go beyond the traditional experimental arrangements covered by many researchers in the past. Therefore, it was decided to use NetEm as the network emulation tool in the to-be-designed testbed.

2.12 NetEm (Network Emulator)

NetEm, from Linux Foundation [70], has been very popular among network researchers over the past few years. Many researchers [71] [72] [73] [74] have been utilising the capabilities of NetEm to carry out experimentation over a range of scenarios to study network traffic.

Network Emulator, as the name suggests, is a Linux based tool that has the capability to emulate a vast range of network scenarios. It allows network engineers to emulate wide area network properties to test protocols. With the help of NetEm, one can emulate various networks whilst having a control over delay, loss, duplication and re-ordering of network data packets.

“NetEm provides Network Emulation functionality for testing protocols by emulating the properties of wide area networks. The current version emulates variable delay, loss, duplication and re-ordering”, direct quote [70].

NetEm allows two types of delay emulation. The first type of NetEm delay, a fixed delay, is depicted in Figure 2.9.

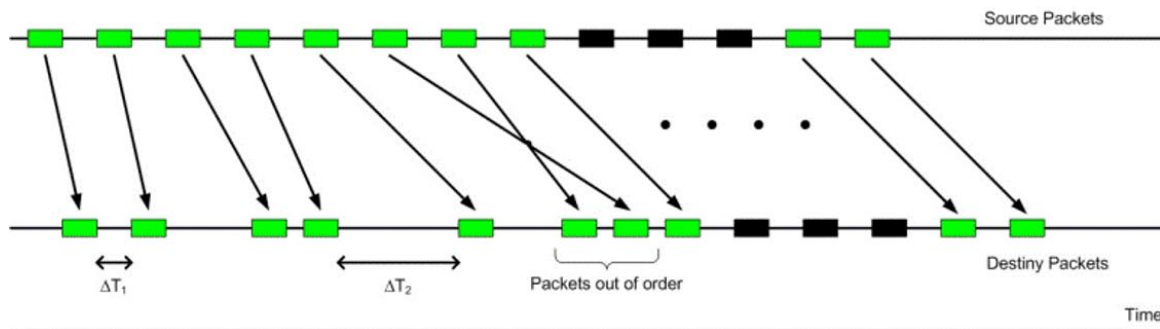


Figure 2.9: NetEm adds a fixed delay to outgoing data packets.

The second option available in NetEm to emulate a delay is with the use of a distribution. NetEm has built-in delay distributions called Normal and Pareto. A delay can be added to the outgoing traffic based on one of these built-in distributions. This is illustrated in Figure 2.10.

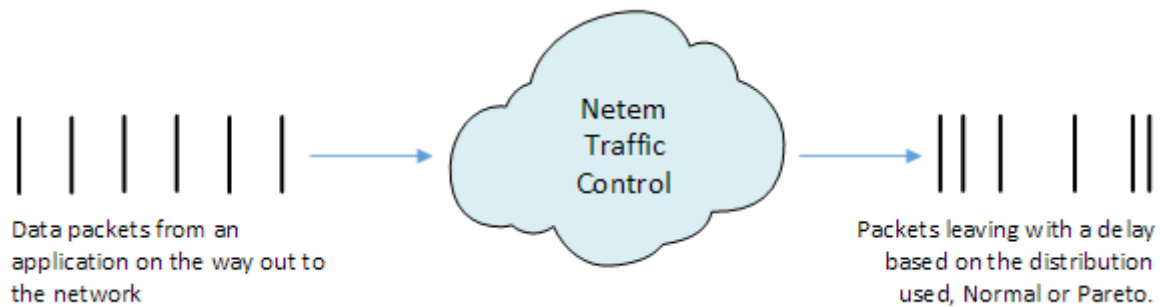


Figure 2.10: Packet delay added using a delay distribution in NetEm.

NetEm is a very common tool used widely by many researchers. It allows the network researchers to emulate network environments for the purpose of evaluating new applications and protocols. These environments offer the testing and design of network related applications in controllable conditions where the experiments can be repeated infinitely for concluding a desired outcome or a concrete finding. Among many other companies, Actual Experience (www.actual-experience.com) and Teragence (www.teragence.com) use NetEm for network research.

2.13 Previous QoE Research on NetEm

The experiments in this thesis were carried out on a testbed that utilises the core functionalities of NetEm. NetEm has previously been used by many researchers including [31] [71] [72] for studying networks' behaviour. In [31], NetEm was used to analyse VoIP traffic, in order to study the behaviour of jitter and delay in the traffic. The researchers also noted that the effect of PLP on the QoE was not as severe, and the MOS dropped below 2 only after the PLP reached as high as 20% [31].

However, the approach taken in [31] to evaluate jitter and delay has followed the method introduced by the Linux Foundation, the people behind NetEm. This way of modelling jitter simply involves adding +/- 't' milliseconds to the arrival time of each packet. This, however, does not represent real life networks since the actual networks do not show a uniform delay jitter [44] [75]. As such, modelling a realistic jitter

model in a controlled lab environment, which reflects real world telecommunication networks, has been a challenge for network researcher for many years.

Despite being a very popular network emulator tool, NetEm with its default emulation settings has a major flaw – although network jitter can be modelled with minimal effort with out-of-the-box configuration, the jitter model under the default settings does not represent patterns found in real world networks. Real world networks do not follow a predefined delay/jitter pattern, which is what NetEm allows as input. For instance, at one particular part in the network there could be delays that are negligible for many applications, however, at another point in time there could be massive delays at the same network node due to congestion or a link failure. NetEm’s official developers are aware of this flaw in NetEm, and therefore describe the application of delay/jitter as an “approximation”. This approximation, however, is acceptable for low level network emulation, but not reliable for a detailed analysis of a network behaviour. The main limitation that NetEm poses, i.e. being unable to produce a realistic jitter model, is also highlighted by [82].

Furthermore, by default NetEm does not retain the original packet order in the presence of non-trivial jitter. NetEm uses a queueing discipline called TFIFO (see Section 4.8 on page 52 for details for operational details of TFIFO). TFIFO in NetEm has the same working principle as a datagram network, meaning that the packets could be going through different nodes across the network and arrive at the destination in any order regardless of the original sequence they were initially transmitted in (see Section 3.4.4 for a review of theoretical behaviour of this queueing discipline). This behaviour of NetEm would not affect the results severely in general network emulations where the focus is on packet loss. However, in a QoE investigation losing packet order can be a major drawback with a direct impact on the QoE degrading it significantly [76] [77] [78]. Again, this was an issue that required attention to gain convincing results.

The research represented in this thesis attempts to overcome these challenges by 1) modelling a delay jitter from a self-made distribution table, and 2) implementing a traffic control with PFIFO queueing discipline. The newly developed delay distribution tables were inputted in NetEm to create jitter values from many hundreds of randomly produced variants. This brought the resulting delay jitter very close to that seen in real world networks; refer to Appendix G on page 209 for details.

The distribution that was chosen to be built in this thesis has been outlined in [79] and [80]. This delay jitter distribution was created by utilising NetEm-offered functionality that allows the user to build an arbitrary distribution [71].

2.14 Previous Work Involving Different Age Groups

Earlier QoE research has not been done across a wide range of age groups. In schools, some published work on QoE has studied QoE across age groups [81]. Otherwise a report to the UK Government’s Office of the communications regulator in the UK (OfCom) concluded that “...*the scores are comparable across the different age groups, although the oldest participants had a tendency to score higher and the youngest to score lower...*” [82]. Also, a survey of global broadband networking by consultants Ovum (2016) concluded that “...*Younger people tend to be heavier and more demanding users, meaning that their usage is higher and expectations greater*” [83].

The literature, e.g. [84], [85] and [86] suggests that the age group 19-30 is much better represented in earlier work mapping QoE/MOS to QoS metrics. This is the age group that contains most PhD students and post-docs, so most published QoE/MOS to QoS mapping the results will have tended to be biased towards this group, and will have necessarily excluded the group of slightly younger users who make up a considerable proportion of heavy users of broadband networks (as noted earlier). These younger users are potentially of considerable interest to network and service providers. The experimental results in this research suggest that youngest people in the 10-18 age group are the least patient out of all other age groups, and hence require a higher level of service quality. It was concluded that, as a comparison with the most often studied 19-to-30-year-olds, if the same level of MOS score is to be achieved by 10-18-year-olds an order of magnitude improvement in the network PLP will be required. Experiments were done to determine the capacity increase for targeting the aforementioned drop in the PLP. This capacity increase was studied through a bottleneck link that has multiplexed TCP traffic queueing at the link. It was seen with the results achieved, that the bottleneck link capacity needs to be increased approximately 3-times (unless RTTs on the network are very low) to achieve the above-mentioned order of magnitude improvement in the PLP.

Videos are not only seen just by people from a certain age group, hence any study of video on demand (VOD) QoE will be more viable if a range of user groups were involved in the research to give their opinion: from young to elderly people. As reported by the UCL researchers, the brain of an adult person functions differently to a young person or a child [87]. It has also been researched that elderly people have more patience than children and adults when it comes to experiencing visuals, and that "...elderly people are more susceptible to a negative experience" [88], and "...older people tend to be significantly happier and better at appreciating what they have..." [89]. Furthermore, in Chapter 1 of the book *Brain Aging: Models, Methods, and Mechanisms* [90] it is stated that "The basic cognitive functions most affected by age are attention and memory. Neither of these are unitary functions, however, and evidence suggests that some aspects of attention and memory hold up well with age while others show significant declines. Perception (although considered by many to be a precognitive function) also shows significant age-related declines attributable mainly to declining sensory capacities".

It is believed that earlier work in the literature has not focused enough on age varying QoE. For example, a recent (2016) PhD thesis states of its experimental subjects in a study of QoE "*In this study, a total of 43 subjects participated in subjective tests. The mean age of participants was 24.5 years, the maximum age was 32 years and minimum age was 21 years.*" [91]. Hence, the aim of this research is to outline the experimental approach of studying QoE on users from a range of age groups. Having determined the most critical of these users in terms of service demand, we evaluated (as previously noted) the capacity cost of meeting the QoS parameters demanded by this group of people from the age group in question. Both the report [82], and Ovum survey [83] support the basis of this part of experimentation to examine QoE across age groups, and its effect on the underlying networks.

2.15 Chapter Summary

Before proceeding with the experimentation in this research, it is important to understand what led to the approach taken, and the reason for conducting this particular research work. This chapter reviewed the research work previously carried out in QoE in the field of data networks. It was also described how the research work in this thesis builds on and improves the QoE work previously engaged in across the area of networks.

Chapter 3 Relevant Technical Concepts and Terms

3.1 Chapter Introduction

In this chapter, technical concepts and terms relevant to this research have been discussed. These technical concepts may serve as background knowledge for a better understanding of the work done in this research. Firstly, properties of a video file were outlined that govern the quality and size of a video file. Then some light was shed on the media player, VLC player, used for network streaming in this research. In the final section of this chapter, transport protocols along with some other network-specific terms are discussed.

3.2 Video Properties

The word video is no mystery to us these days as we see videos every day on a range of platforms including the web, TV, digital video disc (DVD) player, and our own videos recorded with our mobile phone cameras. The aspect, however, that we usually do not pay much attention to is the properties of these videos that differentiate them from one another in various features.

In this Section, some video-related technical terms have been described that are of significant importance equally to the video cameras manufacturers and video makers, and in some cases, to the viewers of the videos too.

3.2.1 Bit Rate

Bit rate (also written as Bitrate) is the number of bits processed per unit of time in a video or in other file types. Bit rate is generally expressed in bits per second (bps) but also occasionally denoted as kilobits per second (kbps) and megabits per second (Mbps).

As a rule of thumb, the higher the bitrate the higher the video quality. For instance, the videos seen on web are generally 1-2 Mbps. Bitrate of a DVD video is between 4-8 Mbps which is about four times higher quality than a web video. A Blue-ray HD video is in a range of 20-25 Mbps [92]. However, the higher quality as a result of higher bit rate also comes with a cost of higher file size which can be a strict limitation in some cases. A very popular video codec used these days in video, H.264, has various bit rates targeting different platforms. [93] has listed them as follows.

LD 240p 3G Mobile @ H.264 baseline profile 350 kbps (3 MB/minute)
LD 360p 4G Mobile @ H.264 main profile 700 kbps (6 MB/minute)
SD 480p Wi-Fi @ H.264 main profile 1200 kbps (10 MB/minute)
HD 720p @ H.264 high profile 2500 kbps (20 MB/minute)
HD 1080p @ H.264 high profile 5000 kbps (35 MB/minute)

It can be noted from the above listing which covers a variety of video qualities that higher bit rate gives a higher quality video, but in return the produced video file has a bigger size corresponding to the greater bit rate of the video.

3.2.2 Video Frame Rate

In the world of video, a frame is a single image that is viewable by the video viewer. What we see as a video, in reality, is a series of still images. As these images are played in front of us very quickly, hence our brain gets tricked into believing that sequence of images to be a smooth motion video. Frame rate, thus, is the number of frames, i.e. images, that are played in every second in a video. Frame rate is expressed in frames per second (fps). As long as the frame rate of a video is a minimum of 16 fps, our brain will see it as a video playback [94]. The diagram below illustrates this concept.

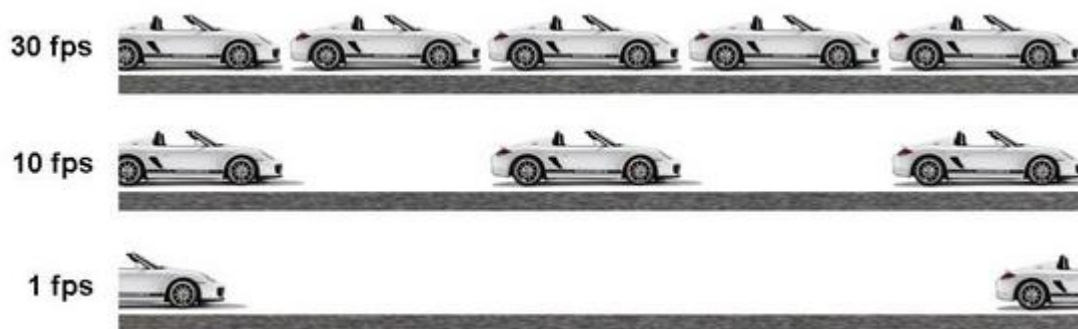


Figure 3.1: Visualising frame rate in a video [94].

Since the 20th century, for film videos the standard frame rate has been 24 fps, whereas for TV broadcast it has been 30 fps in some countries including North America and Japan, whereas 25 fps in other countries including Europe [94].

Although the frame rate has been traditionally 24 fps in the world of cinema for decades, recently, this traditional barrier has been crossed by the director of *The Hobbit: An Unexpected Journey* movie, where the director doubled the frame rate for a film from 24 fps to 48 fps for the first time in history. According to Warner Bros., the concept behind this higher frame rate in a film was bringing the visuals even closer to reality [95]. In general, the higher the frame rate the smoother the video.

3.2.3 Video Aspect Ratio

Aspect ratio refers to the ratio between the width and height of an image or video frame. Aspect ratio is commonly expressed in a “width x height” or “width:height” form. For example, a 16:9 video aspect ratio means that the video is 16 units wide and 9 units high. It should be noted that aspect ratio is not same as the size of the video file, but it relates to only the relationship between width and height of the video [96] [97].

The popular aspect ratios used widely are 4:3, 16:9, and 21:9. 4:3 was the previously used standard for TV screens from the mid of 20th century till the new wide screen TVs came out recently [98]. Wide screen TVs, along with DVD and other high-definition videos, use 16:9 aspect ratio. 21:9 aspect ratio, also known as Cinemascope, has mainly been used in motion picture films in the past. Recently computer monitors, also referred to as UltraWide monitors [99] and some TVs are also coming in 21:9 aspect ratio.

The Figure 3.2 below visualises how different aspect ratios look on a screen.

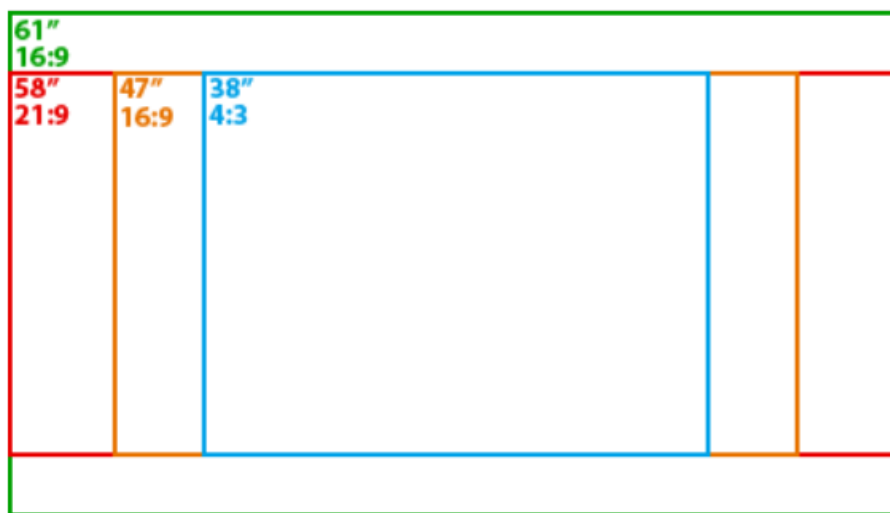


Figure 3.2: Common video aspect ratios on a screen, along with currently popular TV sizes for each aspect ratio [97]

Old TVs and standard definition broadcast channels in the past used 4:3 aspect ratio. New TVs and flat screen computer monitors have an aspect ratio of 16:9, which is also the aspect ratio used by high definition (HD) TV broadcast channels. The movies are generally filmed with an aspect ratio of 21:9, which is why the projector screens in cinemas have a 21:9 aspect ratio.

3.2.4 Video Formats/Containers and Codecs

Videos these days come in various formats, or technically termed, containers. A container contains video and audio data, and information about the video such as codec to be used to play the video, required audio codec, and possibly metadata for example video title and subtitle information. The file extension of a video file generally refers to the container used for that video. Some popular containers are Audio Video Interleave (.avi) from Microsoft, QuickTime (.mov or .qt) developed by Apple, Advanced Video Coding High Definition (AVCHD) developed in collaboration by Sony and Panasonic, and Flash Video (.flv, .swf) designed by Macromedia, a company that later merged with Adobe [100] [101].

Video codecs have the job to represent analogue data in a digital form. A codec is a shorter word for compression/decompression. The design of codec requires various considerations to be taken into account, for instance, the codec should be able to offer a good quality video with a minimum size of the video file whilst utilising a lowest possible bit rate but keeping the data loss at a minimum [102].

A codec can be a hardware device or software. Without the right codec a media player, whether a device or software, will not be able to play a media file. A screenshot of an error thrown by Windows Media Player in the absence of a required codec is shown in Figure 3.3.

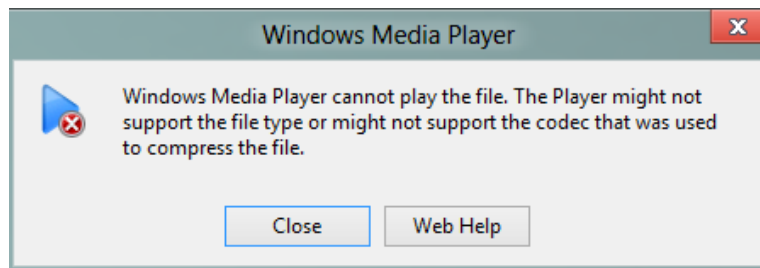


Figure 3.3: A media file unable to play without the right codec.

Some of the popular codecs are [100] [103] [104]:

MPEG-1: from Moving Picture Expert Group, mostly used in VCD production.

MPEG-2: used in DVD production, and sometimes in HDTV broadcast.

MPEG-4: the latest compression algorithm of its family, very popular across a variety of formats ranging from full HD to the lowest sized mp4 videos.

H.264 (also known as MPEG-4 AVC): very popular among people who work with high definition digital video, used in digital video cameras and camcorders, can compress good quality videos for the web and equally for HD TVs.

WMA: mostly used in video/audio streaming, supports 720 and 1080 high resolutions.

DivX (DivX-encoded Movie): offers video compression with a minimal loss in quality, supports high definition resolutions up to 1080. May not be supported by some VCD players.

Xvid: an open source equivalent of DivX, with a Xvid decoder, it is possible to play DivX media files.

All codecs are designed having a particular platform in mind. The choice of a codec for producing videos is always a compromise between the file size and video quality. Depending on whether a video file is being prepared for a DVD or a web usage, or anything in between or beyond, we would choose a specific codec to meet our particular needs.

3.2.5 Video Resolution

Resolution is a measure of pixels, defining the number of pixels present in a certain image. Pixels are small square shaped dots in different colour that make up the whole picture that we see [105]. In simple terms, video resolution defines how clear the video will be once played back on a video player (whether

hardware or software). It is important to know that a video monitor or TV should also be HD for viewing a HD video.

Video resolution is expressed in “length (in pixels) x height (in pixels)” of the video and denoted in pixels, written ‘p’ usually. If there are more pixels in a picture, the image will be clearer and very detailed. On the other hand, if there are less pixels present in a picture then the image will not look as sharp. The picture given in Figure 3.4 illustrates this further, where the resolution starts with 16X16p and goes up to 512X512p.



Figure 3.4: The concept of image/video resolution illustrated by Vimeo with their logo [105]

The above picture from Vimeo visualises that a higher resolution picture (having more pixels) will look much clearer and sharper.

Most common video resolutions currently are SD with a resolution of 640x480, HD with a resolution of 1280x720p, full HD with a resolution of 1920x1080, and 4K Videos that have resolution in the range of 4000x2160p [105] [106].

Some video streaming platforms, such YouTube, allow us to watch videos in various resolutions in order to keep the playback smooth considering our specific Internet connection speed. In the screenshot given in Figure 3.5, the actual resolution of this video is 4000x2160p (4K) but user can switch between many other available resolutions.



Figure 3.5: A YouTube video available in different video resolutions [107]

In the resolution option of YouTube, however, only the height parameter of the video resolution is given starting with 240 pixels going all the way up to the maximum available resolution for this particular video. The highest available resolution available on YouTube player would be the actual video resolution at which that video has been shot on originally.

3.3 VLC Media Player

VLC, an open source media player from VideoLan [108], can be used on Windows, Linux, and MacOS computers. In addition to working as a media player to play a wide range of media files, VLC has very powerful streaming capabilities that allow local or network streaming with an extensive control over the protocols and codecs to meet specific streaming requirements.

The VOD streaming can be done on VLC using a range of protocols including HTTP, UDP, and RTP. A full list has been given in Figure 3.6, followed by a description for each protocol as documented on the VLC website [109]:

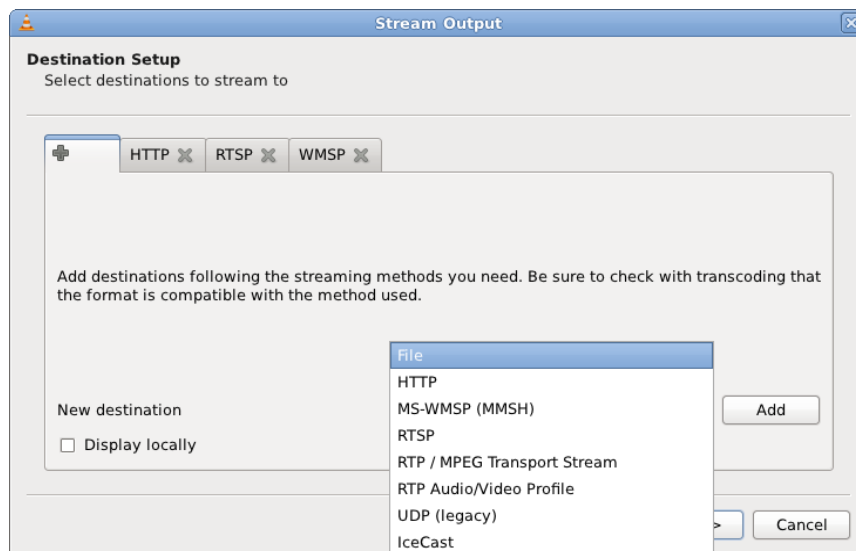


Figure 3.6: VLC built-in protocols for network streaming

- *Display locally*: display the stream on your screen. This allows one to locally display the actual video being streamed. Effects of transcoding, rescaling, etc. can be monitored locally using this function.
- *File*: Save the stream to a file.
- *HTTP*: Use the HTTP streaming method. Specify the TCP port number on which to listen.
- *MS-WMSP (MMSH)*: This access method allows one to stream to Microsoft Windows Media Player. Specify the IP address and TCP port number on which to listen. Note: This will only work with the ASF encapsulation method.
- *UDP*: Stream in unicast by providing an address in the 0.0.0.0 - 223.255.255.255 range or in multicast by providing an address in the 224.0.0.0 - 239.255.255.255 range. It is also possible to stream to IPv6 addresses. Note: This will only work with the TS encapsulation method.
- *RTSP*: Use the Real-Time Transfer Protocol. Like UDP, it can use both unicast and multicast addresses.
- *IceCast*: Stream to an IceCast server. Specify the address, port, mount point and authentication of the IceCast server to stream to.

Transcoding can be used if the video is to be streamed with a specific format regardless of the original format of the video. The required transcoding option can also be chosen from available profiles in VLC player.

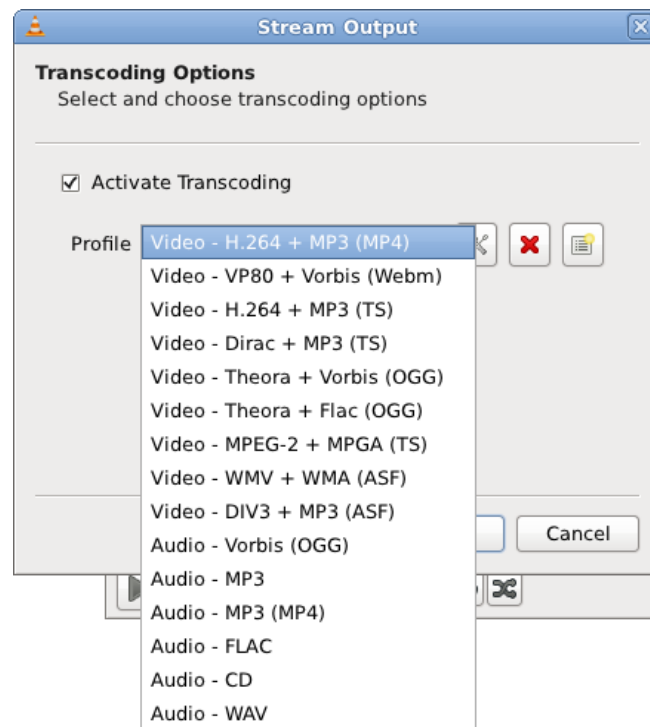


Figure 3.7: VLC transcoding options for output streaming

The transcoding options shown in Figure 3.7 can be customized too with control buttons given on the right side of the dropdown menu for profiles. VLC also gives the option to create new profiles for transcoding.

3.4 Network Technical Terms

3.4.1 Transmission Control Protocol (TCP)

In telecommunications, various protocols are used for governing the behaviour of transmission that would take place. TCP is perhaps the most widely used protocols on the Internet. TCP has been designed to be used in a system where the reliability and performance has priority over the transmission speed. It offers a guarantee for delivering every packet from the source to the destination.

From a technical aspect, TCP keeps the source and destination hosts in communication through a synchronised connection. The bulk of the transmission data (or file) is divided into small chunks (called segments) and transmitted separately [110]. Each transmitted packet is numbered. When a recipient receives a packet with a unique packet number it sends an acknowledgment packet back to the sender to confirm that a particular packet has been delivered. If an acknowledgement is not received in a certain duration of time, TCP assumes that the packet was lost on the way to the receiver, hence the sender is required to transmit the same packet again to address the packet loss. This way, although, the data transmission is slower because of the waiting time among individual packet transmissions, all information is transmitted without any loss or corruption in data.

TCP requires every host to have a unique Internet Protocol (IP) address. Then, with the help of ports, various connections to request/deliver various services can be established with the same device having

a unique IP address. A host with one IP address is able to communicate over 65535 unique ports at the same time [111].

With sophisticated mechanisms such as TCP, there is certainly a complex set-up if its specifications are seen, and every complex system poses many challenges. Likewise, TCP due to its dynamic traffic control gives a slow data transmission. TCP can be a choice of protocol when the transmitted data is desired to be received in full without any portions of it missing. For example, in the case of a file download, if the download is delayed by a few seconds it will not frustrate the user as much as having the file a few seconds earlier but not being able to open this corrupted file due to some bits of data missing in the downloaded file.

3.4.2 User Datagram Protocol (UDP)

UDP has been designed to transmit data as quick as possible, as a result, UDP does not keep track of whether a data packet was successfully delivered or not. Hence having the error checking not being enabled by default makes it a faster protocol as compared to TCP. Unlike TCP, UDP sends all the packets without waiting for the acknowledgement from the recipient on whether or not any particular packets were received or not. If any packets get lost during transmission they cannot be retrieved or resent, the sender will rather carry on sending the remainder of packets.

Technically, not having the synchronisation and error-correction present in UDP reduces the overhead, which in return gives a faster communication between hosts. Therefore, UDP is ideal when loss of a few packets will not affect the overall experience of a user [110], such as in streaming a video or Skype calling.

3.4.3 First In First Out (FIFO)

Queueing systems used in various systems use buffers to store data packets temporarily before transmitting them on into the network or taking from the network in the case of incoming packets. There are some rules, commonly referred to as Queue Disciplines, that govern the order of the packets being taken out of the queue.

The queueing disciplines used in buffers to dequeue the packets can differ from application to application. There are many queue disciplines some of which are FIFO, LIFO, FCFS, and random [112].

FIFO buffers take the packets out of the queue based on their order of arrival in the buffer, hence first in first out. So, the first packet that arrived in the queue will be served first. FIFO has the advantage of offering a simple implementation. FIFO also maintains the packet order, hence is suitable for packet networks where packet order needs to be preserved.

3.4.4 Time First In First Out (TFIFO)

TFIFO is a NetEm queueing discipline based on FIFO. However, this queueing discipline gives precedence to arriving packets based on time instead of the original packet order formed at the sender host. With that as a working principle, TFIFO sends out packets that arrived in a buffer before the ones reaching the buffer later in time, even if the later arriving packets may have a higher priority according to the packet order number set by the sender.

Normally in packet transmission, after packets are transmitted from source they are free to take any path along the network to avoid congestion. At the receiving end, packets coming from all different routes are then put back in order before sending to the application they have been intended for. However, TFIFO works like a datagram network where packets take various paths and may reach in a different order at their destination. Figure 3.8 depicts this scenario.

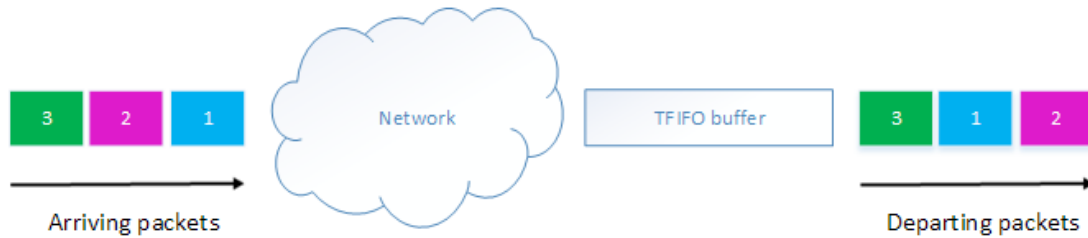


Figure 3.8: A TFIFO queue that resembles a datagram network where packets go out of order in transmission

In a TFIFO queue, if some data packets are delayed due to congestion in the network this will significantly affect the QoE as the packets will be prioritised based on their arrival time at the buffer in question.

TFIFO is the primary queueing discipline used in NetEm. Hence to stay aligned with real world network traffic behaviours, TFIFO had to be replaced with a suitable queueing discipline that does not re-order the packets based on time in order to preserve the original packet order set by the server computer.

3.4.5 Packet First In First Out (PFIFO)

PFIFO queueing discipline, based on FIFO, is the second available queueing discipline in NetEm. Unlike TFIFO that gives priority to packets based on their arrival time in the queue, PFIFO transmits packets based on the actual order of packets in which they were initially sent from the server. Figure 3.9 illustrates the functionality of a PFIFO queueing discipline.



Figure 3.9: A PFIFO queue, which resembles a voice circuit network where packets are in order on the receiving end

As seen in Figure 3.9, even though some packets may experience delays across the network and reach the buffer out of order, still they are put back in order before transmitting them onto the egress port. The packet order is strictly preserved even if this requires delaying all other packets just to maintain the original packet order. A PFIFO queue resembles a virtual circuit, where a line is dedicated to one connection and data packets only from this established connection utilise this given link to ensure packets follow the same path and stay in order. Delays are additive in PFIFO but not in TFIFO, therefore, PFIFO models real-world networks more credibly.

3.5 Chapter Summary

This chapter discussed some key concepts and terms that serve as the prerequisite for this research work. As this research includes streaming of a video and then quality assessment of this video, hence, here the video properties were discussed that are considered in the making of a video as well to maintain the quality of a video. Then several options of VLC media player were conversed that allow the local playback and streaming of a video with a desired video format and video quality. Some network phenomena and protocols were also debated.

Chapter 4 Research Methodology

4.1 Chapter Introduction

This chapter overviews the whole process of the research methodology: from starting the research work on the existing testbed and the need of an alternative testbed to the design of new hybrid testbed.

For the design of the testbed, some technical drawbacks experienced during the design improvements have also been outlined. The chapter then concludes with the most recent design of the testbed, which is now being used by other students in Networks Research Group.

4.2 The Router Lab (Previous Testbed)

Initially a traditional testbed encompassing network routers and traffic generators was used and some of the experimental work on this existing set-up was conducted in the router lab. Initially, about twelve months were spent operating this testbed to develop the theoretical understanding of network simulations/emulations and the driving principles of the testbed in the router lab.

However, the existing lab environment posed some challenges that required a serious consideration whether or not the testbed would align with the objectives of this research. It was unimaginable that natural disasters could have an impact on the pace of work until early summer 2015, when the router lab was seriously affected by storms allowing the rain water to penetrate the lab building. This rain water caused the lab air-conditioning unit to fail, which resulted in the lab temperature to rise to 45°C. This massive increase in temperature caused serious damage to the lab equipment. Although the kit was repaired, the incident raised concerns over the consistent use of the router lab in the long run.

Another reason to seek an alternative experimental setup was the limitations and unnecessary challenges posed by the router lab that incorporated the actual routers and traffic generators. As the network components in question were the actual hardware network devices, configuring the testbed for

experimentation over large network scenarios would be very complex due to the innate lack of flexibility. Thus, studying advanced and state-of-the-art traffic patterns such as Video on Demand (VOD) would require a lot of work, and even then, this would not guarantee full control over the experimental setup. This router lab showed these limitations since it was originally designed to enable experimentation on static traffic only. This static traffic would be stochastically modulated by a Stochastic Modulation Tool (SMT), a software tool that worked in conjunction with the Xena traffic generator and was designed for VoIP traffic only.

To address the above issues and meet the specific research objectives, a brand-new testbed was designed with an aim to conduct experiments more effectively. This newly designed testbed not only overcame the above outlined issues, but also offered more operational capability for carrying out a diverse range of experiments on a variety of traffic types to evaluate VoIP QoE, real time video QoE, TCP/UDP video QoE, and more.

4.3 Design Considerations for the New Testbed

Although it was now decided to design a more robust testbed, the actual specifications of the testbed were still to be explored to assess what proportions of hardware and software were to be implemented in the new testbed. Considering the performance and functionality that was required for the testbed, the distribution of network tasks among hardware and software also required thorough planning. For details on the challenges that were faced during the testbed design refer to Appendix A on page 183.

As for the design structure of the new testbed, it was desired not to have something too close to traditional network simulations such as modelling network scenarios in Matlab. At the same time, reflecting on the past incident with the router lab, a major part of the testbed specifically being hardware-dependant was not required either. Having considered many design approaches over a significant period whilst in the design phase, the design specifications were finalised on the basis that the network features that network simulation/emulation tools offer should be taken advantage of. It was also anticipated that the testbed would facilitate the interfacing of actual consumer devices such as a computer, laptop, a tablet device, and a mobile phone. This would allow the study of a diverse range of network scenarios with a possibility to evaluate the service experienced by the end user on a particular device in a specific infrastructure, whether wired or wireless. After a comparison of popular network emulators, NetEm was chosen as the underlying system for the new testbed. For details, refer to Section 2.11 on page 32.

4.4 NetEm Limitations

While network simulation and emulation tools offer a way to mimic various network scenarios, saving us the effort and cost to put together actual network components, they unfortunately possess some restrictions that become of particular importance if a good fidelity is required in analysing a network.

NetEm, like many other simulation tools, has some limitations too. One which raises a significant concern is the fact that NetEm's delay jitter model does not represent the "patterns" witnessed in real world packet networks. These real-world network patterns show variability and do not follow a predefined trend of delay and jitter on a certain link. For example, there could be higher delays at a certain time over a network link, and other times (even in the matter of minutes), data packets could be propagating very swiftly with a negligible delay or jitter on that link. NetEm, however, only allows

implementation of jitter within a range specified by the input command. As outlined in the NetEm official documentation, delay can be implemented on an interface with the following command:

```
# tc qdisc change dev eth0 root NetEm delay 100ms 10ms 25%
```

According to NetEm developers, “...this causes the added delay to be $100\text{ms} \pm 10\text{ms}$ with the next random element depending 25% on the last one. This isn't true statistical correlation, but an approximation.” [70].

Notably this approach has been described just as an ‘approximation’ by the NetEm developers themselves.

In this research however, this issue was resolved, the details of which are to follow.

4.5 The Approach Taken to the Use of NetEm

Researchers have previously been utilising the default settings of NetEm to model packet loss, delay, and jitter as shown in many papers including [71] [72] that outline the fundamental operational facilities of NetEm. [72] also highlights one of the main limitations that NetEm poses – its inability to produce a realistic jitter model.

In this research, however, the limitations shown by NetEm were bypassed whilst only making use of the best features that NetEm has to offer. As the NetEm delay jitter model fails to represent the patterns observed in real packet networks, in this research, bespoke delay distribution tables were implemented into NetEm. It was then possible to experiment with the VOD streams in a controlled lab environment with more accuracy. This way, the behaviour of network traffic was analysed by aligning it with real world network scenarios, which has not been possible previously in a lab environment. This functionality was achieved by producing jitter tables with a very large set of statistical values. These jitter tables were then changed into distribution tables to make them operational in NetEm. When delay was applied from NetEm with these distribution tables, the resulting traffic stream showed variability in delay based on this rich collection of statistical values present in the distribution table, hence bringing delay patterns very close to that of those seen in real-world Internet traffic.

Another functionality that was added to this NetEm testbed was the ability to retain the original packet order. By default, NetEm puts data packets out of order when jitter is added. This issue was resolved by the use of a PFIFO queueing discipline.

Thirdly, the testbed features have been advanced to apply delay, jitter, and loss simultaneously on network traffic. This combination of network parameter application is not possible with NetEm out-of-the-box capabilities.

With the above-mentioned added features in the testbed, one can add delay, jitter, and loss on the video being streamed and observe the effect on the perceived quality of the VOD i.e. the QoE at the user end.

As such, the testbed was designed to not only use the core capabilities of NetEm but also give full control to model traffic behaviour for a wide range of network scenarios. This way, studying video traffic through local streaming, as well as sourcing from over the Internet such as YouTube, was made possible.

4.6 NetEm Testbed: The Initial Design

The newly designed testbed consists of two Linux computers: a server, and a client. The role of the server was to house the NetEm facilities and take input from the user to stream and manipulate the traffic by altering its requested network parameters. The role of the client on the other hand, was to take the outputted traffic from the server and play it back for MOS evaluation or the testing.

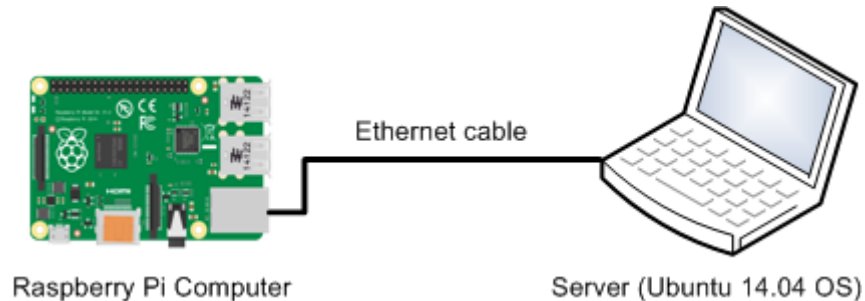


Figure 4.1: Testbed model

As the client in this testbed was only responsible for playing back the video, or responding to whatever is sent to it, there were no limitations on the operating system running on the client computer with the option to have Windows, Macintosh, or Linux. Any Linux distribution would work, but to keep things simple and to utilise a wide range of utilities available in Ubuntu, it was decided that the Linux-based operating system of choice would be Ubuntu (www.ubuntu.com).

A laptop was used as the server and a Raspberry Pi (RPi) as the client. Both, the server and the client were interlinked via a CAT5 Ethernet cable. Both computers were assigned a unique IP address so they could communicate with each other on the network. For instructions on assigning IP addresses to the server and the client, refer to Appendix D on page 191 and Appendix E on page 201.

As NetEm was running on the server⁵ any outgoing traffic from the egress could be manipulated utilising NetEm traffic management capabilities. Figure 4.1 shows the design of the initial testbed where a Raspberry Pi was used as a client and my laptop had the role of a server to control the traffic.

4.7 Testbed Improvement – Replacement of the RPi

Having tested various network scenarios with the use of NetEm on the designed testbed, it was noted that the testbed was operational. The precision, however, was not as required and the RPi was failing to keep packet scheduling in precise places.

It was suspected that due to the limited processing power of RPi, it introduced lags in its response time. To verify that, another alternative set-up was trialled where the RPi was replaced with another laptop and the collected results were analysed again. On running the ping probing with the same network parameters, it was seen that the RTT in milliseconds was shorter in the case of a laptop being used as a client as compared to the RPi being the client. This showed that the RPi introduced additional delay of a few milliseconds when responding to the ping packets sourced from the server. This would certainly be a major concern when the delay would be required to be very precise in the later experimentations.

⁵ The instructions on getting the latest build of ipRoute2 for NetEm are provided in Section 4.11.1 on page 47.

Therefore, this limitation of processing power of the RPi seemed a drawback in the designed testbed, which needed to be addressed.

The RPi was then removed from the testbed and replaced with a laptop to function as a client. Now in this version of the testbed, there were two laptops connected to each other via an Ethernet cable working as a client and a server.

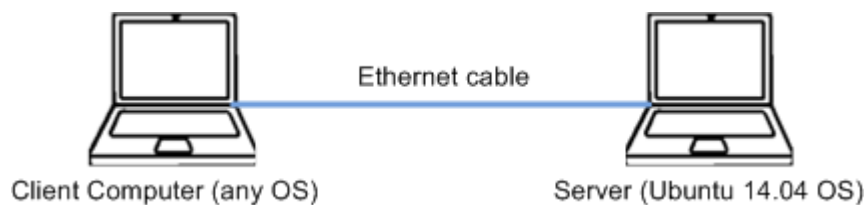


Figure 4.2: Testbed Improvement - replacement of the RPi with another laptop

Now, moving on from the basic type of experimentation i.e. the ping probing to experiment with the packet delay and loss, experimentation to model realistic network scenarios was desired. As previously packet loss and delays have been studied under QoE research on VoIP traffic [113] [114] [115], it was decided that the analysis of video traffic would be a state-of-the-art arrangement.

The possible options for the design of a server-client model for traffic flow between the two computers was studied next. During the design evaluation and online research on the topic, a range of ways for accomplishing the task were considered. However, it was later realised that a too complicated client-server model was not required if the main objective was to only analyse one type of traffic: the video traffic. As the networking between the client and the server had already been set up, to merely transport video traffic on the network, the network streaming capabilities of the VLC media player (www.videolan.org) could have been utilised. VLC is an open source cross-platform multimedia player hence free to use.

The testbed was then prepared for VOD streaming which would allow analysis of the video traffic⁶. Upon completion of this iteration of the design of the testbed, the video successfully played on the server and streamed over the network. The video streaming was evaluated and analysed for assessing the playback quality on the client computer. This way, the effect of network performance metrics on the video traffic was studied, by not only emulating this, but actually being able to see the affected QoE on the actual video. It was then possible to stream the video from the server, and by adding any network delay or packet loss from NetEm, study the output video across the network on my client computer. With this advance testbed, a range of protocols could be used to transport the video traffic over the network, including UDP, HTTP, and RTP / MPEG Transport Stream.

4.8 Operating NetEm with PFIFO instead of TFIFO

By default, the queueing discipline that NetEm uses is TFIFO (see Section 3.4.4 on page 45 for a review of theoretical behaviour of this queueing discipline), which may not be an issue for simple network

⁶ For detailed instructions on how to prepare the server and client to talk to each other for analysing the video traffic and capturing MOS, refer to Appendix D on page 186 and Appendix E on page 196.

emulations encompassing packet loss. However, when jitter is added to the network traffic through NetEm, the packets start getting re-ordered due to the behaviour of TFIFO queueing systems. TFIFO makes NetEm act like a datagram network where the packet order would not matter. However, packet order has a significant importance linked directly to QoE. Packets that are out of order at the destination node cause the QoE to degrade significantly [76] [77] [78]. Therefore, retaining packet order in packet networks will massively improve the QoE. Not being able to keep the packets in order in NetEm is a major flaw in this emulator if one plans to carry out advanced level network emulation to cover a wide range of network scenarios to mimic real life packet networks.

Many people have desired to address this issue for disabling this automatic packet re-ordering in NetEm [116] and [117] are two of those places where this question was asked. The solution of course is replacing the TFIFO with a PFIFO, as instructed by Linux Foundation [118], the people behind NetEm:

“Starting with version 1.1 (in 2.6.15), NetEm will reorder packets if the delay value has lots of jitter. If you don't want this behaviour, then replace the internal queue discipline TFIFO with a pure packet FIFO PFIFO.”

However, these official instructions are not very precise and the NetEm manual does not specifically go into detail of how this can be done in practice. Having attempted various ways to achieve the desired functionality, initially there was not much progress since it was a widely recognised complex problem awaiting a solution. Further progress was made by exploring the Linux network routing principles. This fundamental knowledge of Linux traffic control was believed to be vital for comprehending the traffic structure and the level these mechanisms could be configured on.

By exploring the Linux traffic control hierarchy, it was discovered that every interface consists of one egress root qdisc, and a handle is assigned to each qdisc (and class) which allows one to target a certain qdisc for configuration purposes. This qdisc-assigned handle has two parts in the form of <major>:<minor>. The <minor> number to a root qdisc has always a value of 0, whereas for the root qdisc, the <major> number is traditionally chosen to be 1 [119]. It was also noted that if there are classes applied on a qdisc they must have the same <major> number as their parent, which should be unique within an egress/ingress setup. However, the Linux kernel only communicates with the root qdisc rather than cascading through the whole hierarchy tree. The diagram in Figure 4.3 depicts this concept.

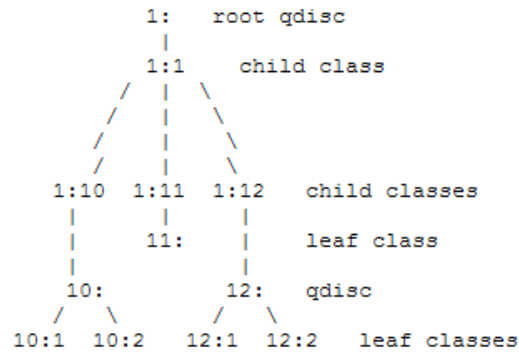


Figure 4.3: Classifying traffic in Linux, traffic filters' hierarchy [119].

I then advanced my understanding on the traffic queueing behaviour within the hardware buffer. It revealed to me that when a packet needs to be released from the queue and sent to an interface for onward transmission, the kernel sends a dequeue request to the root qdisc (at the top of the tree given in Figure 4.3). Nested classes do not communicate with an interface but only to their parent qdisc; the kernel only dequeues the root qdisc.

Following this, it was noted that it will not be possible to change TFIFO to PFIFO simply by replacing the word TFIFO with PFIFO in my NetEm instructions. But instead, different levels of traffic control hierarchy shown in Figure 4.3 were to be addressed. To be able to that, the handles provided in Linux traffic control were to be used. This way, the default NetEm TFIFO functionality could have been replaced with a PFIFO queueing discipline.

Targeting various levels of the traffic control hierarchy, of course, consisted of a set of instructions that were to be executed in NetEm for activating the PFIFO functionality on the egress port. The desired functionality was achieved with the use of filters on the same interface, which was facilitated by the handles. Ultimately, regardless of how excessive the added jitter was, NetEm was able to keep the packets in order during the transmission. Additionally, this improvement to the testbed made it possible to combine jitter, delay, and packet loss and still operate the queue as PFIFO. This way, the industry-known NetEm issue of automatic packet reordering on jitter application was resolved. To see a summary of hands on practical steps for operating NetEm with a PFIFO queue, refer to Appendix C on page 189.

Progressing to complex experimental scenarios where delay, jitter, and loss were to be added simultaneously, it could be quite tedious to type various lines of instructions in NetEm command prompt at the same time. Hence to save time, when executing a large number of instructions, bash scripts were later developed that simplified the process.

This state-of-the-art setup was then showcased to other researchers who found it very valuable for studying the QoE. In their discussion, other researchers informed that although the QoE has been a focus of research for a long time, the expected outcomes so far were mainly in the form of simulations and the set of numbers that were further interpreted to approximate particular behaviours of video traffic over a network. But on this newly designed testbed, being able to see the added network imperfections on an actual network-streamed video was indeed a step forward in QoE research. Furthermore, the researchers agreed that this controllable environment to study QoE on a specific type of traffic such as video was unarguably a very effective and scalable way of conducting QoE research.

4.9 The Use of USB Ethernet Adapter and its Limitations



Figure 4.4: USB to Ethernet adapter

Replacing the RPi with a laptop in the testbed solved the problem that the RPi posed – not being able to provide sufficient processing power. However, as the new client laptop in the testbed did not have an Ethernet port, a USB-NIC adapter to do the network interfacing had to be used.

This adapter served the purpose but introduced extra delay at the network interface of the client computer. This would not be ideal when the precision was required in comparing the added delays against the analytical delay model. Therefore, eliminating this adapter was thought to be a necessity. This then led to reviewing the design of the testbed.

4.10 NetEmBox-1

The USB-NIC introduced additional delay which was seen in the playback of the video streamed across the network. This delay experienced in the video was uncontrollable as it was not explicitly added from NetEm. This unexpected and irrepressible delay was clearly a drawback potentially giving me inaccurate results in the later stages when the NetEm added delay was to be controlled precisely. Hence the design of the testbed was to be improved.

In this upgraded version of the testbed, the main server setup was moved to a desktop computer (previously a laptop). This improved design of the testbed is shown in Figure 4.5. This new testbed design had many advantages over using a laptop as a server. Firstly, a desktop computer had much higher processing power when compared to a laptop. Secondly, this design gave the flexibility for upgrading or modifying the hardware in order to achieve the flexibility to carry out experimentation over a variety of network states.

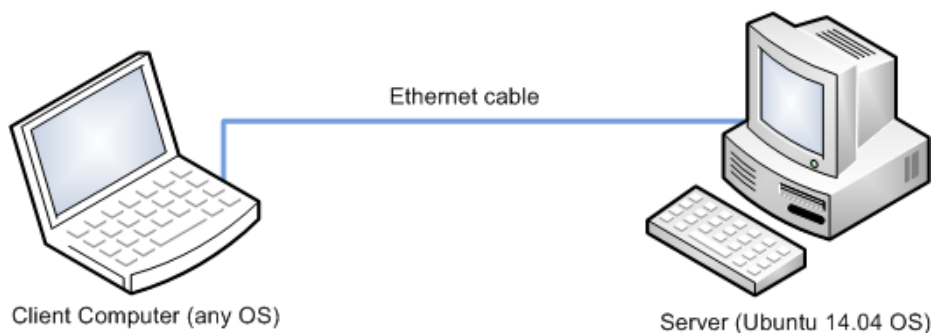


Figure 4.5: Testbed upgrade, a more powerful server computer

This fully functional and optimised testbed was called NetEmBox-1.

The server on NetEmBox-1 was running Ubuntu 14.04, which was a stable version of Ubuntu operating system with long term support. The client computer would work despite the type of operating system installed on it: Linux, Mackintosh, or Windows.

The design of the testbed at this stage had met all predetermined requirements, hence the experimentation was put under focus until further hardware changes were required in the testbed to meet any specific research needs.

So, proceeding with the experimentation, the effects of various added network deficiencies through NetEm were studied and the resulting video output across the network was viewed on the client computer. Figure 4.6 through to Figure 4.10 show a volunteer participating in recording the MOS.



Figure 4.6: VOD streaming with 0.01% packet loss. A participant evaluating QoE and recording MOS

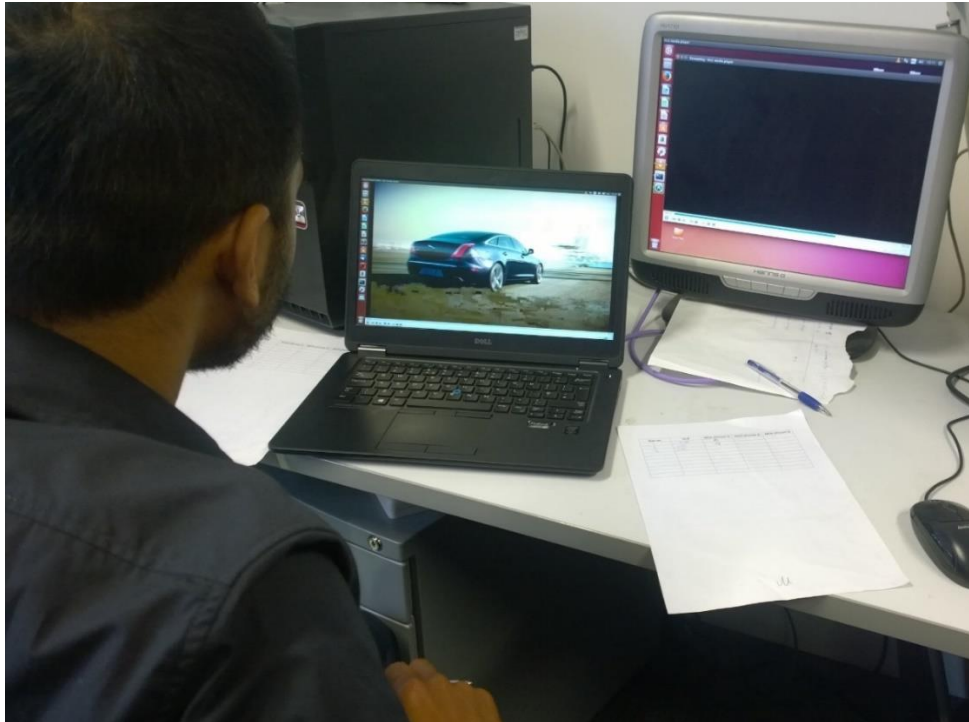


Figure 4.7: VOD streaming with 0.1% packet loss. A participant evaluating QoE and recording MOS

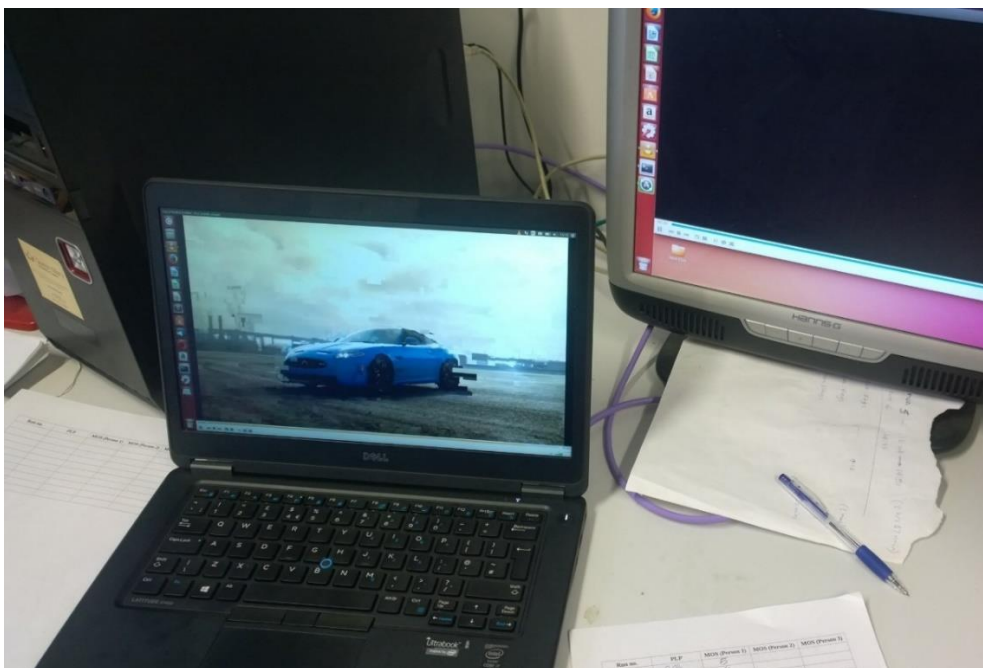


Figure 4.8: VOD streaming with 1% packet loss. Evaluation of QoE

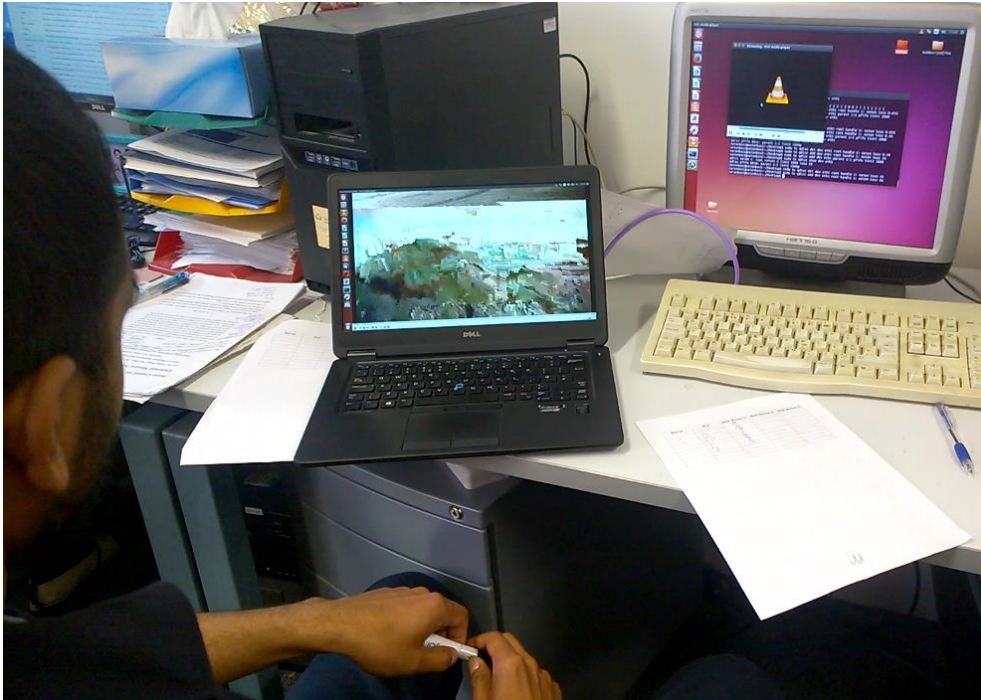


Figure 4.9: VOD streaming with 5% packet loss. A participant evaluating QoE and recording MOS

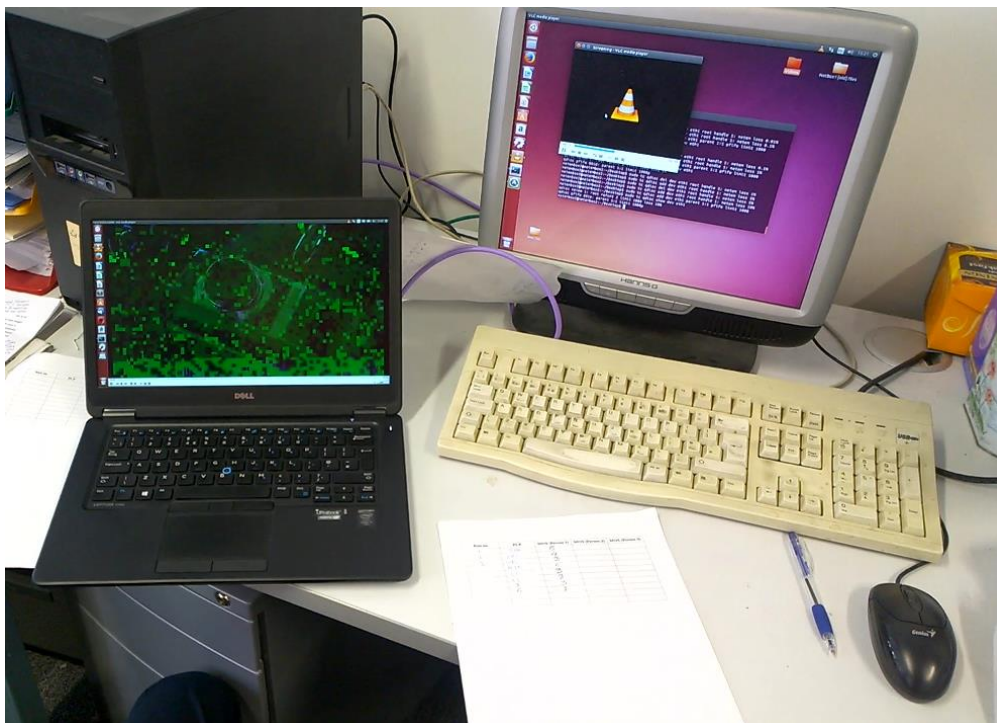


Figure 4.10: VOD streaming with 10% packet loss. A participant evaluating QoE and recording MOS

As the Figure 4.6 to Figure 4.10 depict, the video was shown to the participants with different levels of added packet loss through NetEm. It can be seen that as the applied packet loss increased, the video gradually showed more imperfections. To see the experimental results, refer to Section 5.4 on page 74.

Having experimented with video traffic under various default NetEm settings, it was now time to move on to study more realistic patterns of network traffic. As stated in Section 4.5, this would require the creation of custom jitter-distribution-tables, details of which are covered in the next section.

4.11 Custom Delay-Distribution-Table Implementation in NetEm

Real network traffic does not follow a specific pattern, hence delay in such traffic is unpredictable. To reflect the behaviour of the actual network traffic and in addition to offer the standard traffic control options, the distributions that NetEm has available to be used are the Normal and the Pareto distributions. According to NetEm developers “Typically, the delay in a network is not uniform. It is more common to use a something like a normal distribution to describe the variation in delay [70].” However, NetEm allows users to make custom distribution tables (based on experimental data) in NetEm for applying the delay on the traffic in question. Delay distribution tables were produced to model real-world-like jitter pattern in the network traffic. Refer to Appendix G on page 209 for details about the algorithm that was used. Distribution tables created this way gave a more convincing delay distribution based on the traffic patterns described in Section 2.8.

Although NetEm gives the option to add custom distribution delays, NetEm official documentation does not show any detailed instructions or any examples on how to do so. Additionally, any similar authentic work previously done by any researcher was not found. Therefore, the nature of the task not being very straightforward, various ways of implementing the distribution tables were tested and validation experiments were carried out to verify its operation.

One typical way of creating a delay distribution table for a certain network is by using ping statistics for that particular channel or network. For instance, if one wants to analyse the behaviour of (or emulate) a network between two computers connected on a network, they ping between them to get large number of delay values for the network connecting both computers. The RTT values are then extracted out of the ping statistics and used to create a distribution table. The mean and the standard deviation are obtained from these RTT values, which are then used in the NetEm command when activating the distribution table produced. The larger the number of pings the better the resulting delay model.

To create a delay distribution table in this research however a different approach was used. With the help of an algorithm, a set of statistical values was created for a particular standard deviation. This spreadsheet of values, which was based on a given standard deviation, were processed further for creating a delay distribution that could be used use in NetEm. Refer to Appendix H on page 209 for details.

For describing the procedure of creating these distribution tables, the major steps have been highlighted in the following section.

4.11.1 Creating a Delay Distribution Table

As the operating system used on the testbed was Ubuntu, the instructions below may only work in most recent versions of Ubuntu.

To be able to create custom distribution tables, two utilities `./maketable` and `./stats` are required. Then one would use them in order to design the delay distribution tables from a given set of experimental data.

Firstly, the source code for iproute2 should be downloaded:

```
git clone \
git://git.kernel.org/pub/scm/linux/kernel/git/shemminger/iproute2.git
```

Then having navigated to the NetEm directory, the maketable and stats utilities were prepared. They were compiled with the following commands:

```
cd iproute2/NetEm
gcc -o maketable maketable.c -lm
gcc -o stats stats.c -lm
```

The statistical data produced was in an Excel spreadsheet, but having tested that for distribution design, it was noted that it would simplify the process if this data was firstly moved to a text file. Hence, the Excel spreadsheet data was moved to a txt file before proceeding further.

The steps below should be followed in the given sequence to successfully create a delay distribution table that can be used in NetEm. Step 1 only applies if a user has the statistical delay values initially in an Excel spreadsheet.

- 1 Copy data column from Excel sheet, make a new txt file, and paste the copied data into this and save it, giving it an exact name as the Excel file for your convenience.
- 2 Open a Terminal from the menu or with Ctrl+Alt+T, and navigate to the directory where you saved the txt file from step 1 (e.g. Desktop), copy this txt file to iproute2/NetEm with the following command:

```
cp filename.txt /home/PC_NAME/iproute2/NetEm
```

- 3 Compile the distribution table. This would require converting from .txt to a .dist file. For this, you firstly need to navigate to the NetEm folder with the command similar to this:

```
cd iproute2/NetEm/
```

- 4 Now once in NetEm folder, make use of the ./maketable utility to create a distribution table from the text file:

```
./maketable filename.txt > distribution-name.dist
```

- 5 Copy the distribution file to the tc folder:

```
cp distribution-name.dist /usr/lib/tc
```

- 6 Get the stats of your data (the txt file) for obtaining your mean and standard deviation.

```
./stats filename.txt
```

Save these stats somewhere since you will need them later on. This step was not necessary for the setup in this research as the mean and standard deviation had already been calculated.

- 7 The delays are now ready to be activated from the distribution table. Go to the tc folder:

```
cd /usr/lib/tc
```

- 8 Add the delays from the newly created distribution table:

```
sudo tc qdisc add dev eth0 root NetEm delay 8ms 0.8ms distribution
distribution-name
```

Where 8ms is the mean and 0.8ms is the standard deviation of number of delay.

- 9 To clear the delay replace the keyword add with del:

```
sudo tc qdisc del dev eth0 root NetEm delay 8ms 0.8ms distribution
Distribution-name
```

Section 5.12 on page 97 includes experimental work based on custom-delay-distribution tables.

4.12 NetEmBox-1 as a Proxy Server

Having conducted the QoE study on local network traffic, the research was advanced to study the network traffic originating from the Internet, such as YouTube or Skype traffic. It is worth outlining that the connection to the Internet had specifically been chosen to be a wired connection. This is because a wireless connection (due to common wireless connection limitations) would lead to introducing additional uncontrollable transmission delays. On the other hand, a wired Internet connection would give much faster connection. Additionally, the connection between the Internet and NetEmBox-1 was not just a standard wired connection, but rather a fast Ethernet link with 1GB/s transmission rate.

Figure 4.11 illustrates this experimental setup, referred to as proxy server setup.

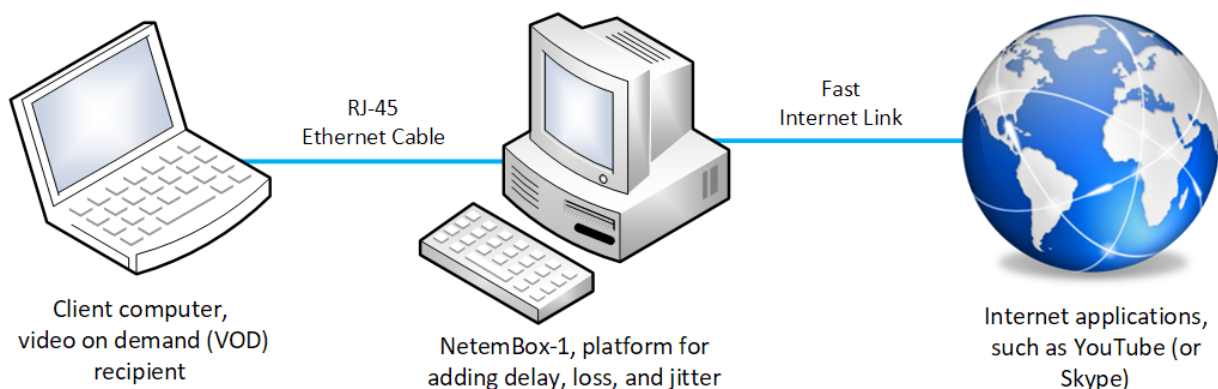


Figure 4.11: Proxy server arrangement for experimenting with Internet traffic

To be able to study the Internet traffic, the testbed needed hardware advancement. Hence, proceeding with testbed design improvement, a second Network Adapter was installed on NetEmBox-1. This would allow the testbed to be connected to the Internet for experimenting on the network traffic from across the Internet, e.g. YouTube traffic.



Figure 4.12: The network configuration of NetEmBox with one Network Interface Card (NIC) left, and network configuration with two NICs after the upgrade, right.

Figure 4.13 depicts the upgraded hardware capabilities of NetEmBox-1 to be able to work as a proxy server. An additional Local Area Network (LAN) port allowed a wired connection to the Internet via a fast link Ethernet port. The second Ethernet port would serve the purpose of facilitating the local streaming, which was the primary function of the previous versions of this testbed.



Figure 4.13: NetEmBox-1 upgrade – separate LAN ports for a wired Internet connection and local streaming

Configuring two LAN ports was somewhat challenging as the computer would get confused as to which Ethernet was being used for what purpose. This was not straightforward as the IP addresses were of different nature for both configurations. Various ways were trialed for accomplishing the task and

eventually both LAN ports were successfully assigned the desired functionality. This involved configuring network parameters on both the server computer and the laptop.

This upgraded design then allowed experimentation on the traffic coming from the Internet, which was in addition to analysing the traffic locally. This way, the study could be conducted on the QoE of real life applications whilst artificially modifying the network metrics such as delay, jitter and packet loss.

4.13 NetEmBox-2: The Extended Testbed Facilities

Netebox-1 was desired to always be in working condition for demonstrating research work to other researchers and to collect the MOS. Therefore, any newly desired functions that was to be included in the testbed, was intentionally not tested on NetEmBox-1. This was because a newly added function might have not worked as expected in the first instance, which could leave some of the existing features of the testbed to malfunction too.

Hence, to explore new possibilities, an additional testbed was prepared and was called NetEmBox-2. Whenever a new feature was required in the main testbed that was firstly tested on NetEmBox-2. So, in the design phase, the strategy was to design a feature, test it to verify the expected outcome, then optimise the design by addressing any issues outlined by the testing. This way, even if one particular new feature in question did not work as expected on NetEmBox-2 leaving it temporarily unusable, it would not introduce any setbacks as NetEmBox-1 would still be operational with the most recent stable version of the testbed design.

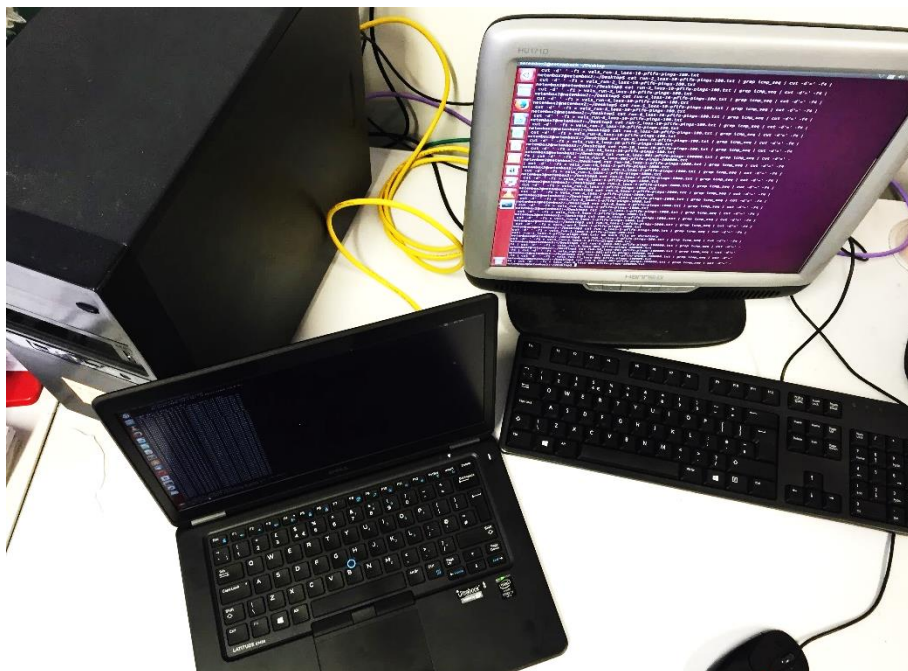


Figure 4.14: NetEmBox-2

For example, the implementation of the Wi-Fi hotspot feature was firstly implemented and tested on NetEmBox-2. Various packages were to be installed (then some removed) and various ways were tested to achieve the best working solution. This occasionally caused local wired streaming to become affected and not work. Therefore, working on this desired Wi-Fi hotspot feature on NetEmBox-2 was very convenient as the NetEmBox-1 was still in a stable working state. Once the desired wireless hotspot

functionality was achieved after attempting various approaches, the very last approach that finally worked was accepted and implemented on the main testbed, NetEmBox-1.

4.14 Testbed Wireless Streaming Functionality for Mobile Phones

The most recent functionality that was added to the testbed was the ability to carry out MOS evaluation on mobile phones. This allowed volunteers to watch the video streaming on their phones and tablets, and record their MOS responses.



Figure 4.15: Wi-Fi hotspot for streaming to mobile phones

Adding the Wi-Fi interfacing was quite challenging as the Linux kernel, Ubuntu, in particular does not support plug-and-play hotspot functionality. As a result, various packages were installed in the test stage to determine the most suitable settings to achieve the desired functionality before a working solution was reached.

Figure 4.15 shows this state-of-the-art testbed offering all types of streaming options: Internet traffic streaming, local wired streaming, and the streaming of Internet and local-network videos wirelessly to hand-held devices such as mobile phones and tablets.

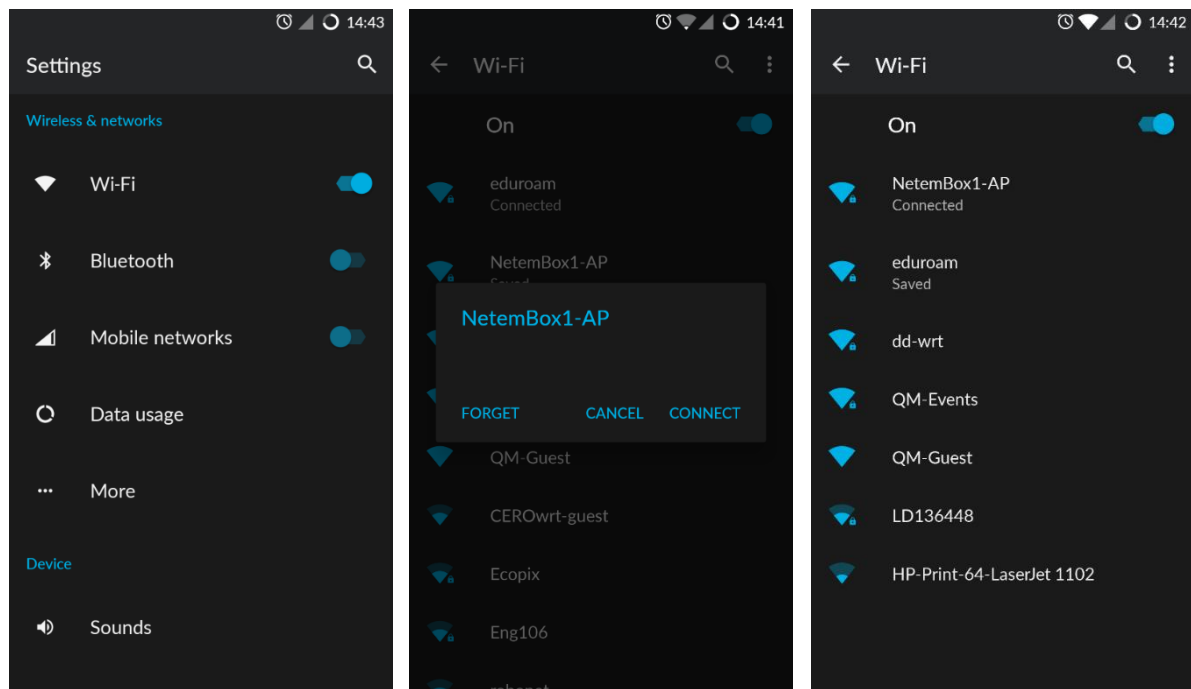


Figure 4.16: Wireless connectivity with NetEmBox-1

The screenshots above show the successful connection of a mobile phone with the NetEmBox-1 hotspot.

4.15 The Final Upgrade: A More Robust Hotspot Design

Although the NetEmBox-1 hotspot design that utilised a USB wireless modem served the purpose of basic mobile connectivity, it did not offer high throughput. Additionally, it was discovered that iOS devices such as iPhones and iPads were not able to connect to this hotspot. It was also observed that when more than one device was connected with this hotspot, some devices would experience intermittent Wi-Fi connectivity.

This required an improved, more versatile design of the hotspot Wi-Fi infrastructure for evaluating MOS. To meet this design objective, further research was conducted in which it was found that using a network router instead of a USB wireless modem would give much higher throughput and signal strength. This is because a robust network router is a dedicated network device that has been designed with performance in mind. A higher specification hardware present in a network router would also allow multiple connections simultaneously without compromising on signal strength or quality.

Therefore, the hardware design was revised and a Cisco router was incorporated in the testbed. This advanced hotspot facility is illustrated in Figure 4.17.

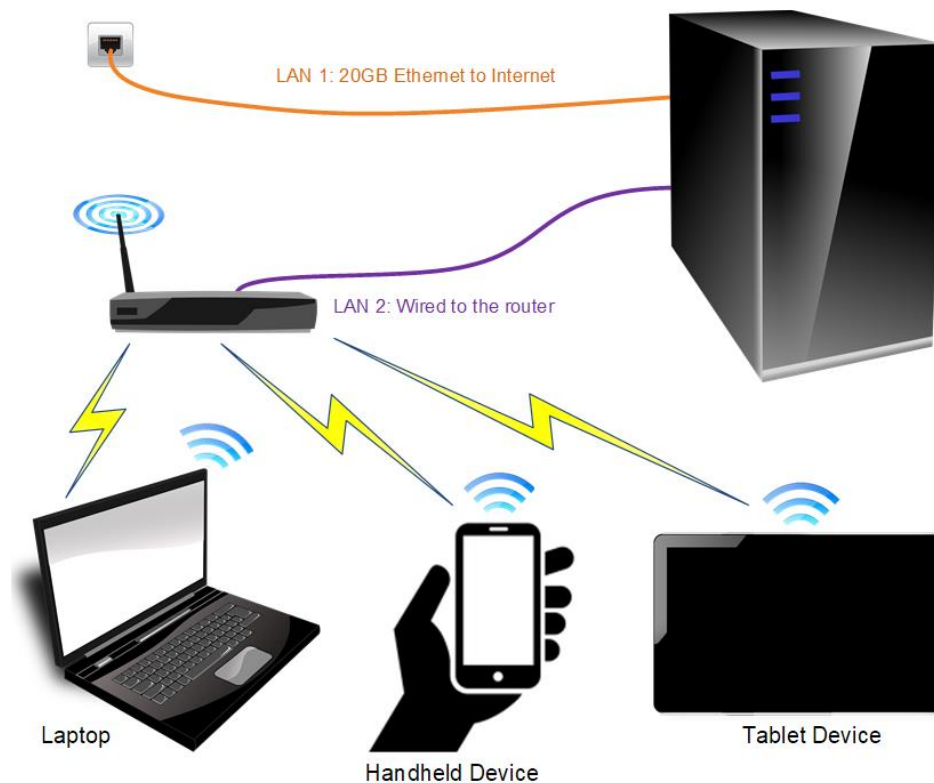


Figure 4.17: The designed testbed with a more robust Wi-Fi connectivity feature

With this upgraded wireless infrastructure of the testbed, any device (regardless of whether its operation required a 2.4GHz or 5GHz channel) can be connected with the NetEmBox-1 hotspot. This full-bodied Wi-Fi infrastructure does not compromise on signal quality and throughput regardless of the number of devices connected to the hotspot simultaneously.

4.16 Results Collected from NetEm-based Testbed are Realistic

Simulators use mathematical models as input and, based on a chosen experimental scenario, give a set of numbers as the output which are then interpreted to draw conclusions that relate these results to real world traffic. With the modifications made to NetEm on the designed testbed, however, a real video source as the traffic generation was used, so it is naturally realistic and closer to reality when compared to mathematical models in simulators and emulators. It was not the traffic generation that NetEm modified, but the model of packet queuing and therefore packet delay and jitter. The configured loss, jitter and delay that were implemented are based on the model given in [79], for details refer to Appendix G on page 209.

4.17 Chapter Summary

This chapter gave an insight into the experimental methodology used in this research. Various stages of testbed development have been described with a gradual improvement in the experimental lab used in this research. The testbed designed in this research has been discussed in depth. The underlying Linux-based network emulator, NetEm, has been discussed and the reason for choosing NetEm over other network emulators and simulators has been explained. Many of the well-known flaws of default NetEm configurations have been discussed. Then, these NetEm limitations were addressed and resolved.

Having laid out all design stages of the designed testbed, it is seen why this hybrid testbed is a state-of-the-art testbed for network experimentation.

Chapter 5 Experimentation with Artificially Added Loss and Jitter, and Bespoke Delay Distributions

5.1 Chapter Introduction

This chapter lays out the experimental work conducted on the NetEm testbed. The experimentation in this chapter has been divided in three parts. In the first section, experiments were done to verify the interfacing between all components of the testbed and it was investigated whether activating more than one network parameter, i.e. packet loss and delay/jitter, would affect the performance of the testbed, potentially leading to inaccurate experimental results. NetEm artificially added packet loss on to a video stream which was then implemented and studied.

Then in the second set of the experiments, experimentation was done to study the Normal and Pareto distributions in NetEm. The default NetEm procedure of adding jitter does not implement realistic patterns of jitter (refer to Section 4.4 on page 49 for details). NetEm facilitates the implementation of delay distributions: Normal and Pareto, which enables NetEm to output delay patterns better than merely inputting a standalone end-to-end delay value. This was explored further in the second part of this chapter.

In the third part of the experimentation in this chapter, QoE was evaluated by streaming VOD over both UDP and TCP protocols with artificially added loss and jitter. Firstly, a QoE evaluation was performed by experimenting individually with jitter and packet loss. Then both of these network metrics: jitter and packet loss were combined, and experimentation with self-made delay distribution tables was used to evaluate the QoE of the video. Using self-made delay distributions represent delay jitter in a far better manner than just adding/subtracting 't', a fixed interval to/from the packet arrival time. In the

subsequent sections the experimentation results are given. The experimental setup was the same as the one given in Figure 4.11 on page 61.

5.2 Preliminary Experiments

Firstly, to verify the successful interfacing between the server and the client, ping probing was conducted – the client sending ping packets out to the server. A response from the server confirmed that the connection between the client and the server was successfully established.



Figure 5.1: Preliminary experiments

Then, packet loss was set from within NetEm and the ping probing was carried out again to verify that NetEm was working and some packets were being dropped as expected.

5.3 Packet Delay Independent of Packet Loss

The hypothesis: packet delay in a controlled packet network does not affect packet loss in that network, was studied and verified with the results that were achieved through experimentation. This part of experimentation was divided in the following stages:

1. Required number of pings were collected in each case;
2. Histograms from collected packet statistics were drawn;
3. The histograms were changed into the PMF;
4. All the points from the stem graph of PMF were added, they were expected to give a sum of 1 if the results were accurate from the PMF;
5. The ping values were analysed to verify the added packet loss in each case.

The sketches of PMF obtained in step 3 above showed convergence to a Gaussian. Here a delay plot was expected where all given values would condense very close to each other giving a bell shape corresponding to the central limit theorem. In the middle of this shape on the x-axis, the mean delay value was expected to be around a millisecond or half a millisecond due to the particular processing

power of the processor driving the Linux computer used. This seen delay would be a very low and introduced by machine processing and due to hardware limitations.

The main objective of this experiment was that, although a packet loss at various levels was introduced, the delay would be approximately the same for all levels of the added packet loss. This would then verify that the packet loss does not affect the packet delay in a controlled packet network environment, which conforms to expected NetEm operation.

To proceed with this experiment, ping probing was carried out and ping packets were sent from the server to the client across the network. To visualise the data conveniently, a target of 10 lost packets in each run of the experiment was set for individual packet loss scenarios. As a range of packet loss values was covered in his set of experiments, with a higher applied packet loss, a smaller number of packets would be required to meet the target of 10 lost packets and vice versa.

In the first experimental run, 10,000 data packets were transmitted. As the aim was to lose 10 packets out of these 10000 transmitted packets, the packet loss was set to be:

$$plp = \frac{10}{10000} = \frac{1}{1000} \quad (5.1)$$

$$= 0.001 \text{ or } 0.001 \times 100 = 0.1\%$$

According to Equation (5.1), in every 1000 transmitted packets one packet is likely to be lost. For calculating the packet delay, the RTT was to be extracted from the ping probing output statistics. RTT is the time taken for a ping packet to go out of the egress port of the server out in the network and to the ingress port of the client computer, and then come back to the sever. Figure 5.2 shows the RTT values that were extracted from the statistics of the first run of the experiment with 0.1% PLP.

99982	0.670
99983	0.626
99984	0.644
99985	0.647
99986	0.623
99987	0.656
99988	0.637
99989	0.627

Figure 5.2: PLP 0.1% - the RTT values extracted from the first run of the experiment

These RTT values were then imported into Matlab and a histogram was drawn to visualise the data pattern. To achieve a histogram of the packet delay values, a MatLab program was created. Refer to Appendix F on page 207 to view the code for this program. With this designed program, the histogram that was obtained of the RTT data is shown in Figure 5.3.

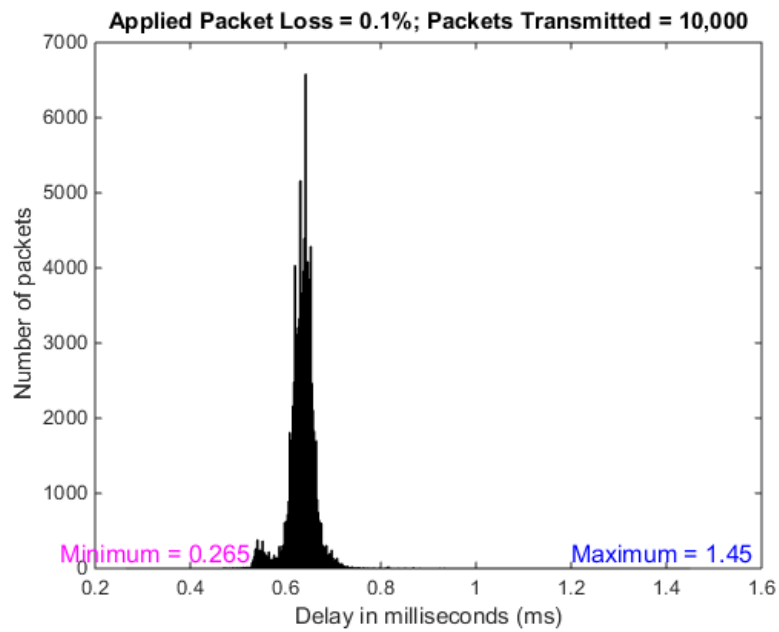


Figure 5.3: Histogram of RTT values extracted from 10,000 transmitted packets.

This histogram showed that most of the packets delayed had the RTT value between 0.5 and 0.7 millisecond. The minimum and maximum delay values in the histogram have also labelled.

The above histogram was then converted into PMF for a meaningful visualisation of the collected data, refer to Appendix F on page 207 to see the Matlab program that was developed to achieve this.

The PMF plot was plotted in stem since this would clearly show the data points, making it easy to analyse the data behaviour. Again, the minimum and maximum data points in the PMF were labelled on the plot.

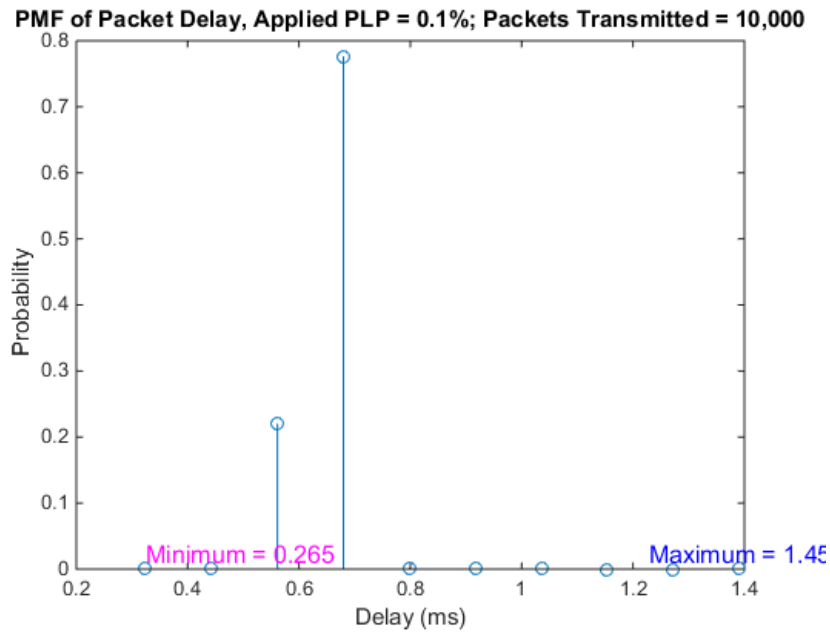


Figure 5.4: PLP = 0.1%, the PMF plot of packet delays

For verification, the 10 data points in the PMF were taken and their sum was computed. It was expected that if the sum totals to a 1 then it would prove to be a realistic PMF of the data being analysed.

pmf										
1x10 double										
	1	2	3	4	5	6	7	8	9	10
1	6.0007e-05	7.5008e-04	0.2202	0.7765	0.0020	4.7005e-04	2.0002e-05	0	0	1.0001e-05
2										

Figure 5.5: The PMF variable created in Matlab

In a similar way, the delay was studied for the rest of the loss scenarios under discussion, 1%, 5%, and 10%. The PMF plots for these have been shown in Figure 5.6 to Figure 5.8.

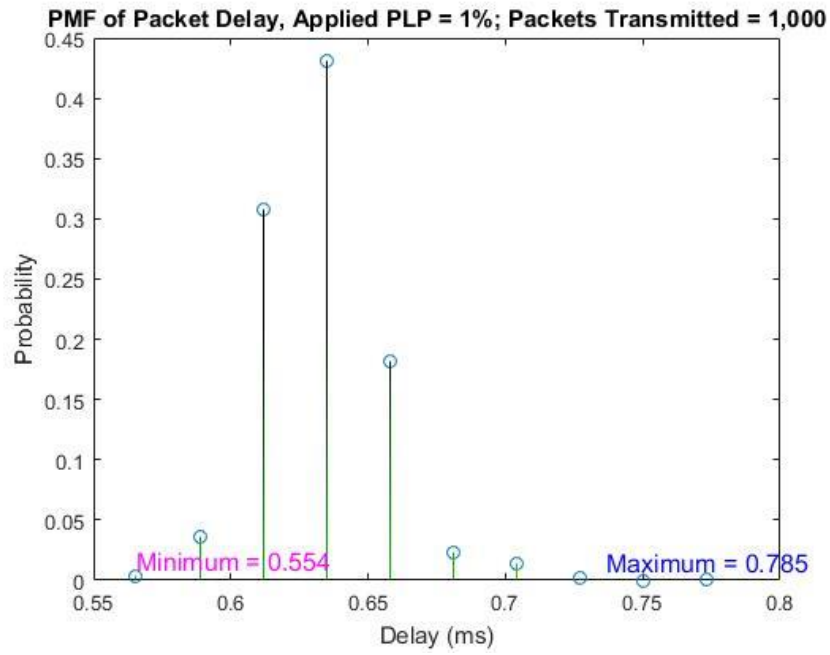


Figure 5.6: PLP = 1%, the PMF plot of packet delays

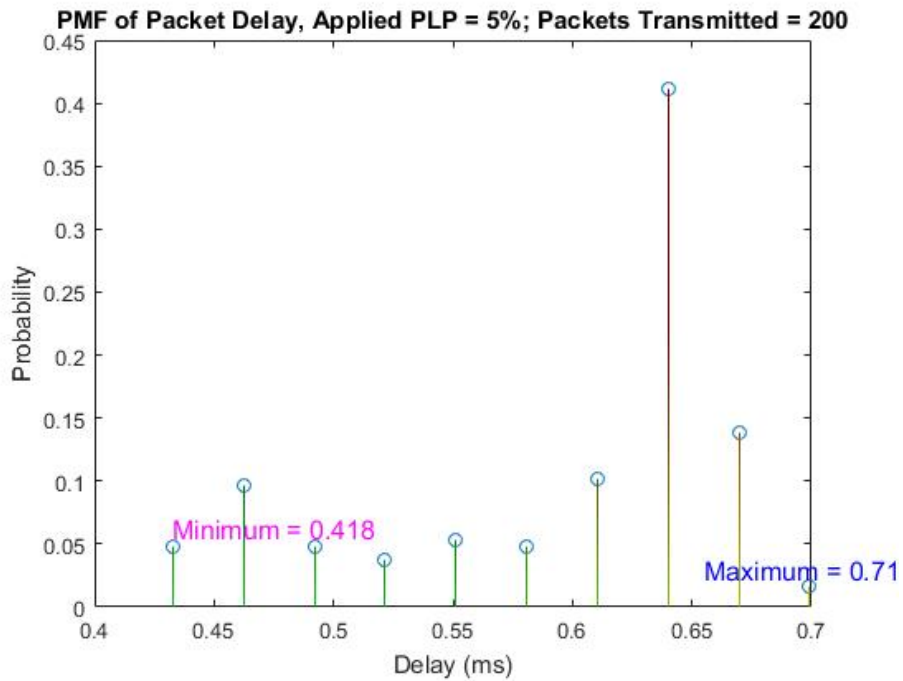


Figure 5.7: PLP = 5%, the PMF plot of packet delays

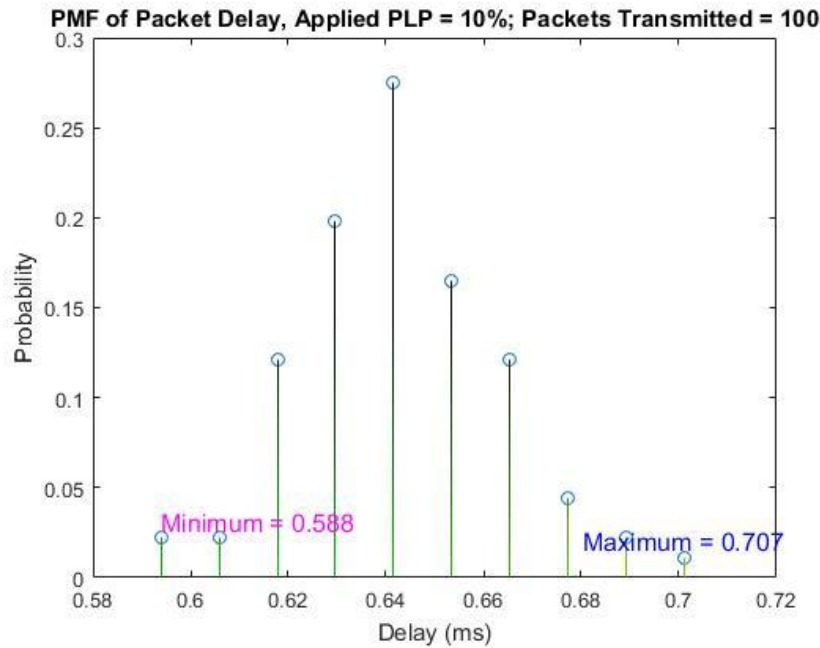


Figure 5.8: PLP = 10%, the PMF plot of packet delays

The PMF graphs shown in Figure 5.6 to Figure 5.8 for all five PLP values depict that although the minimum and maximum delay values (highlighted in cyan and blue in the graphs) were different in each case, the peak of the PMF graph was always seen between 0.6ms and 0.7ms.

This proved that in a controlled NetEm environment, the packet loss does not affect the packet delay. In this part of experimentation having witnessed that the delay for any loss values is almost the same, it was concluded that such packet loss is independent of packet delay, which verifies the expected NetEm behaviour.

5.4 Traffic Analysis with Added Packet Loss

Proceeding further with the experimentation, packet loss in network traffic was studied. This packet loss was added from NetEm and covered a range of values: 0.01%, 0.1%, 1%, 5%, and 10%.

The reason for choosing the above packet losses specifically for experimenting with VOD was the fact that going any lower than 0.01% packet loss is not detected by many simulators/emulators. Whereas, having too high packet loss would result in the video being not viewable. Consequently, many researchers, including [120] [121] and [122] have used packet losses in a similar range for their experimentation.

5.4.1 Case 1: 0.01% Packet Loss

Ping probing allows network researchers to determine packet loss as well as packet delay. The packet loss can be calculated simply by looking at the statistics and subtracting the received packets out of the total transmitted packets.

One can determine the expected packet loss, i.e. the number of packets that would drop in a particular scenario, with the help of the following formula:

$$\frac{plp}{100} \times \text{packets transmitted} \quad (5.2)$$

For example, if the packet loss of 0.01% is activated, and the total number of packets to be transmitted are set to be 100,000, the predicted packet loss using equation (5.2) would be:

$$\frac{0.01}{100} \times 100000 = 10 \text{ packets}$$

In this particular case, it would be expected that 10 packets would drop in this transmission. However, due to randomness, it is very unlikely that running the experiment would give a loss of 10 packets exactly. Even if it did happen in one experiment, running the experiment more than once would show a significant variation. Because of this variation, the packet loss will be less than or greater than 10 in every run of the experiment. This statistical randomness was investigated by analysing the experimental results.

A packet loss of 0.01% was added from NetEm and 100,000 packets were transmitted to the client across the network. This experiment was repeated 10 times and all of these runs did not give a packet loss of exactly 10 packets. Instead, the packet loss varied and wobbled around 10, which can be seen in the results in Figure 5.9 through to Figure 5.13. From these figures it can be noted that for a low packet loss, e.g. 10 packets, the packet loss description in the ping summary may have worded it as “0% packet loss”, however, packet statistics for “received” field did show around 10 packets being lost in each run.

```
99988 64 bytes from 10.42.0.2: icmp_seq=34462 ttl=64 time=0.656 ms
99989 64 bytes from 10.42.0.2: icmp_seq=34463 ttl=64 time=0.637 ms
99990 64 bytes from 10.42.0.2: icmp_seq=34464 ttl=64 time=0.627 ms
99991
99992 --- 10.42.0.2 ping statistics ---
99993 100000 packets transmitted, 99989 received, 0% packet loss, time 99999037ms
99994 rtt min/avg/max/mdev = 0.265/0.634/1.459/0.039 ms
99995
```

Figure 5.9: Experimental run 1 - traffic statistics with 0.01% packet loss

```
99992 64 bytes from 10.42.0.2: icmp_seq=34462 ttl=64 time=0.629 ms
99993 64 bytes from 10.42.0.2: icmp_seq=34463 ttl=64 time=0.636 ms
99994 64 bytes from 10.42.0.2: icmp_seq=34464 ttl=64 time=0.632 ms
99995
99996 --- 10.42.0.2 ping statistics ---
99997 100000 packets transmitted, 99993 received, 0% packet loss, time 99998998ms
99998 rtt min/avg/max/mdev = 0.466/0.638/1.847/0.040 ms
99999
```

Figure 5.10: Experimental run 2 - traffic statistics with 0.01% packet loss

```
99989 64 bytes from 10.42.0.2: icmp_seq=34462 ttl=64 time=0.644 ms
99990 64 bytes from 10.42.0.2: icmp_seq=34463 ttl=64 time=0.672 ms
99991 64 bytes from 10.42.0.2: icmp_seq=34464 ttl=64 time=0.639 ms
99992
99993 --- 10.42.0.2 ping statistics ---
99994 100000 packets transmitted, 99990 received, 0% packet loss, time 99999000ms
99995 rtt min/avg/max/mdev = 0.415/0.638/1.748/0.028 ms
99996
```

Figure 5.11: Experimental run 3 - traffic statistics with 0.01% packet loss

```

99989 64 bytes from 10.42.0.2: icmp_seq=34462 ttl=64 time=0.605 ms
99990 64 bytes from 10.42.0.2: icmp_seq=34463 ttl=64 time=0.634 ms
99991 64 bytes from 10.42.0.2: icmp_seq=34464 ttl=64 time=0.661 ms
99992
99993 --- 10.42.0.2 ping statistics ---
99994 100000 packets transmitted, 99990 received, 0% packet loss, time 99999019ms
99995 rtt min/avg/max/mdev = 0.371/0.637/1.712/0.037 ms
99996
    
```

Figure 5.12: Experimental run 4 - traffic statistics with 0.01% packet loss

```

99989 64 bytes from 10.42.0.2: icmp_seq=34462 ttl=64 time=0.645 ms
99990 64 bytes from 10.42.0.2: icmp_seq=34463 ttl=64 time=0.648 ms
99991 64 bytes from 10.42.0.2: icmp_seq=34464 ttl=64 time=0.640 ms
99992
99993 --- 10.42.0.2 ping statistics ---
99994 100000 packets transmitted, 99990 received, 0% packet loss, time 99999001ms
99995 rtt min/avg/max/mdev = 0.510/0.639/1.718/0.040 ms
99996
    
```

Figure 5.13: Experimental run 5 - traffic statistics with 0.01% packet loss

The above experimental output screenshots shown in Figure 5.9 to Figure 5.13 are only for 5 runs, however, the packet loss values from all 10 experimental runs have been included in Table 5.1. It can be witnessed that the packet loss was not exactly 10 packets for any experimental run out of these 10 runs. But, it was varying above and below 10 packets in every run. This proved that the expected randomness does exist in the NetEm results.

However, following this statistical randomness in results, if one needs to agree to a particular value for a network metric of interest (e.g. the packet loss in this case) for using it in further calculations, such a value would be achieved by taking the Confidence Interval (CI) of the values collected over a range of experiments. This gives an average packet loss, which can be referred to as a realistic packet loss value.

This leads to calculating the CI for this experimental scenario. The packet loss from all 10 runs is utilised for the calculation of the CI. As 0.01% was a very low packet loss, to target 10 packets drop, the number of packets to be transmitted was set to 100,000.

The overall mean PLP was then calculated along with standard deviation and the CI's.

Table 5.1: Experimental packet loss from 10 runs with 0.01% packet loss

Experimental Run(s)	Packet Loss = Packets Transmitted – Packets Received
1	100000 – 99989 = 11
2	100000 – 99993 = 7
3	100000 – 99990 = 10
4	100000 – 99990 = 10
5	100000 – 99990 = 10
6	100000 – 99989 = 11
7	100000 – 99991 = 9
8	100000 – 99992 = 8
9	100000 – 99990 = 10
10	100000 – 99992 = 8

The average packet loss \bar{x} , was estimated with the help of equation (5.3).

$$\bar{x} = \frac{\sum \text{packet loss}}{\text{number of experimental runs}} \quad (5.3)$$

$$\bar{x} = \frac{94}{10} = 9.4$$

The absolute error in the number of packets lost was:

$$\delta = \frac{t_{N-1, 1-\frac{\alpha}{2}} \times \text{standard deviation } (\sigma)}{\sqrt{\text{number of experimental runs } (N)}} \quad (5.4)$$

Where t is determined from a Table-T, and is 2.576 which was determined by 99% confidence interval. Note that t was used, not z, because the standard deviation of the distribution was unknown.

$$\sigma_{PLP} = \sqrt{pq} \quad (5.5)$$

where:

$$p = plp \quad (5.6)$$

And

$$q = 1 - plp \quad (5.7)$$

For the PLP:

$$plp = \frac{\text{lost packets}}{\text{total transmitted packet}} \quad (5.8)$$

$$= \frac{9.4}{100000} = 9.4 \times 10^{-5}$$

The q using equation (5.7) was then:

$$q = 9.4 \times 10^{-5} = 0.9999$$

The standard deviation of the PLP then, from equation (5.5) was:

$$\sigma_{PLP} = \sqrt{9.4 \times 10^{-5} \times 0.9999} = 0.00999 = 9.69 \times 10^{-3}$$

In addition, the standard deviation of number of packets lost⁷ was:

$$\sigma_{NPL} = \sqrt{\frac{\sum_{i=1}^N (x_i - \bar{x})^2}{N - 1}} \quad (5.9)$$

Where:

$x_i = \text{each value of data}$

⁷ There are two types of standard deviations (σ): one for the PLP, and the second one for the number of packets lost (NPL).

\bar{x} = mean value of x_i

N = number of data points.

Now substituting the values in equation (5.9):

$$\sigma_{NPL} = \sqrt{\frac{(11 - 9.4)^2 + (7 - 9.4)^2 + (10 - 9.4)^2 + (10 - 9.4)^2 \dots}{10 - 1}}$$

The above calculation was carried out in MatLab, the program that was developed for this calculation is given in Appendix F on page 208.

$$\sigma_{NPL} = 1.3499 \cong 1.35$$

Using equation (5.4), the absolute error in the number of packets lost for the 10 runs of the experiment was:

$$\delta = \frac{2.576 \times 1.35}{\sqrt{10}} = 1.0997 \cong 1.10$$

To calculate the lower and upper confidence intervals:

$$UCI = \bar{x} + \delta \tag{5.10}$$

$$LCI = \bar{x} - \delta \tag{5.11}$$

Where:

UCI = Upper Confidence Interval

LCI = Lower Confidence Interval

\bar{x} = average packet loss

δ = absolute error in the number of packets lost.

So, from equations (5.10) and (5.11), the CI's with (99% confidence and) packet loss of 0.01%:

$$UCI = 9.4 + 1.1 = 10.5$$

$$LCI = 9.4 - 1.1 = 8.3$$

5.4.2 Case 2: 0.1% Packet Loss

In this experiment, the packet loss was changed to 0.1% and data from 10 runs was collected. The packet loss from each run has been provided in the Table 5.2. For an easy to follow analysis, it was decided that a scenario with the same target packet loss of 10 dropped packets would be focused on. However, as the PLP was now higher than before, to target a packet loss of 10 packets the total number of transmitted packets had to be reduced from 100,000 to 10,000. The overall mean PLP was then calculated the same way.

Table 5.2: Experimental packet loss from 10 runs with 0.1% packet loss

Experimental Run(s)	Packet Loss = Packets Transmitted – Packets Received
1	10000 – 9989 = 11
2	10000 – 9989 = 11
3	10000 – 9989 = 11
4	10000 – 9988 = 12
5	10000 – 9990 = 10
6	10000 – 9986 = 14
7	10000 – 9989 = 11
8	10000 – 9991 = 9
9	10000 – 9990 = 10
10	10000 – 9990 = 10

The average packet loss \bar{x} , was calculated as follows using equation (5.3):

$$\bar{x} = \frac{109}{10} = 10.9$$

The PLP, using equation (5.8), was:

$$plp = \frac{10.9}{10000} = 1.09 \times 10^{-3}$$

The values for p and q using equations (5.6) and (5.7) respectively were:

$$p = 1.09 \times 10^{-3}$$

$$q = 1 - 1.09 \times 10^{-3} = 0.99891 \cong 0.999$$

The standard deviation of the PLP then, from equation (5.5) was:

$$\sigma_{PLP} = \sqrt{(1.09 \times 10^{-3}) \times 0.999} \cong 0.03$$

In addition, the standard deviation of number of packets lost, using equation (5.9):

$$\sigma_{NPL} = 2.0923 \cong 2.09$$

Using equation (5.4), the absolute error in the number of packets lost for the 10 runs of the experiment was:

$$\delta = \frac{2.576 \times 2.09}{\sqrt{10}} = 1.70$$

Therefore, from equations (5.10) and (5.11) the CI's with a packet loss of 0.1% were:

$$UCI = 10.9 + 1.7 = 12.6$$

$$LCI = 10.9 - 1.7 = 9.2$$

The upper confidence interval is larger because the experiment was run only 10 times. If packet loss statistics were collected from a large range of experiments, then the confidence intervals would be closer to the mean packet loss.

5.4.3 Case 3: 1% Packet Loss

Due to a rise in the PLP in part of the experiment, i.e. 1%, number of transmitted packets were reduced to 1000 to stay consistent with the target packet loss of 10 packets. The mean PLP was then calculated for this scenario.

Table 5.3: Experimental packet loss from 10 runs with 1% packet loss

Experimental Run(s)	Packet Loss = Packets Transmitted – Packets Received
1	1000 – 991 = 9
2	1000 – 992 = 8
3	1000 – 990 = 10
4	1000 – 993 = 7
5	1000 – 989 = 11
6	1000 – 990 = 10
7	1000 – 987 = 13
8	1000 – 988 = 12
9	1000 – 992 = 8
10	1000 – 991 = 9

The average packet loss \bar{x} , was calculated with equation (5.3):

$$\bar{x} = \frac{98}{10} = 9.8$$

The PLP then, using equation (5.8), was:

$$plp = \frac{9.8}{1000} = 9.8 \times 10^{-3}$$

The values for p and q using equations (5.6) and (5.7) respectively were:

$$p = 9.8 \times 10^{-3}$$

$$q = 1 - 9.8 \times 10^{-3} = 0.99$$

The standard deviation of the PLP then, from equation (5.5) was:

$$\sigma_{PLP} = \sqrt{(9.8 \times 10^{-3}) \times 0.99} = 0.098$$

In addition, the standard deviation of number of packets lost, using equation (5.9):

$$\sigma_{NPL} = 1.8619 \cong 1.86$$

Using equation (5.4), the absolute error in the number of packets lost for the 10 runs of the experiment was:

$$\delta = \frac{2.576 \times 1.86}{\sqrt{10}} = 1.50$$

Therefore, from equations (5.10) and (5.11) the CI's with a packet loss of 1% were:

$$UCI = 9.8 + 1.5 = 11.3$$

$$LCI = 9.8 - 1.5 = 8.3$$

5.4.4 Case 4: 5% Packet Loss

For the PLP being 5%, the required number of packets to be transmitted was 200 in this case for meeting the packet loss target of 10 packets Packet loss statistics have been given in the Table 5.4.

Table 5.4: Experimental packet loss from 10 runs with 5% packet loss

Experimental Run(s)	Packet Loss = Packets Transmitted – Packets Received
1	200 – 187 = 13
2	200 – 190 = 10
3	200 – 183 = 17
4	200 – 190 = 10
5	200 – 187 = 13
6	200 – 188 = 12
7	200 – 188 = 12
8	200 – 191 = 9
9	200 – 195 = 5
10	200 – 186 = 14

The average packet loss \bar{x} was calculated with equation (5.3):

$$\bar{x} = \frac{115}{10} = 11.5$$

The PLP then, using equation (5.8), was:

$$plp = \frac{11.5}{200} = 0.0575 \cong 0.06$$

The values for p and q using equations (5.6) and (5.7) respectively were:

$$p = 0.06$$

$$q = 1 - 0.06 = 0.94$$

The standard deviation of the PLP then, from equation (5.5) was:

$$\sigma_{PLP} = \sqrt{0.06 \times 0.94} = 0.24$$

In addition, the standard deviation of number of packets lost, using equation (5.9):

$$\sigma_{NPL} = 3.9243 \cong 3.92$$

Using equation (5.4), the absolute error in the number of packets lost for the 10 runs of the experiment was:

$$\delta = \frac{2.576 \times 3.92}{\sqrt{10}} = 3.19$$

Therefore, from equations (5.10) and (5.11) the CI's with a packet loss of 5% were:

$$UCI = 11.5 + 3.19 = 14.69$$

$$LCI = 11.5 - 3.19 = 8.31$$

5.4.5 Case 5: 10% Packet Loss

To achieve a loss of 10 packets under a 10% PLP, 100 packets were to be transmitted. The packet loss from the 10 runs is given in Table 5.5.

Table 5.5: Experimental packet loss from 10 runs with 10% packet loss

Experimental Run(s)	Packet Loss = Packets Transmitted – Packets Received
1	100 – 91 = 9
2	100 – 90 = 10
3	100 – 90 = 10
4	100 – 93 = 7
5	100 – 90 = 10
6	100 – 90 = 10
7	100 – 85 = 15
8	100 – 88 = 12
9	100 – 95 = 5
10	100 – 92 = 8

The average packet loss \bar{x} was calculated with equation (5.3):

$$\bar{x} = \frac{96}{10} = 9.6$$

The PLP then, using equation (5.8), was

$$plp = \frac{9.6}{100} = 0.096 \cong 0.1$$

The values for p and q using equations (5.6) and (5.7) respectively were:

$$p = 0.1$$

$$q = 1 - 0.1 = 0.9$$

The standard deviation of the PLP then, from equation (5.5) was:

$$\sigma_{PLP} = \sqrt{0.1 \times 0.9} = 0.3$$

In addition, the standard deviation of number of packets lost, using equation (5.9):

$$\sigma_{NPL} = 2.7244 \cong 2.72$$

Using equation (5.4), the absolute error in the number of packets lost for the 10 runs of the experiment was:

$$\delta = \frac{2.576 \times 2.72}{\sqrt{10}} = 2.2157 \cong 2.22$$

Therefore, from equations (5.10) and (5.11) the CI's with a packet loss of 10% were:

$$UCI = 9.6 + 2.22 = 11.82$$

$$LCI = 9.6 - 2.22 = 7.38$$

5.4.6 Error Bar Plot for Given Packet Loss Cases

Having calculated the mean packet loss, standard deviation, and confidence intervals for all five cases of PLP, the absolute error, δ was plotted having the packet loss values on x-axis and the corresponding mean PLP values on the y-axis.

Table 5.6: Packet loss statistics along with Confidence Intervals (99% confidence).

Applied PLP (%)	Mean PLP (\bar{x})	Standard Deviation (δ)	Lower Confidence Interval (LCI)	Upper Confidence Interval (UCI)
0.01	9.4	1.10	8.3	10.5
0.1	10.9	1.70	9.2	12.6
1	9.8	1.50	8.3	11.3
5	11.5	3.19	8.31	14.69
10	9.6	2.22	7.38	11.82

A Matlab program was created to plot error bars, this program is given in Appendix F on page 207. The following error bar graph was obtained.

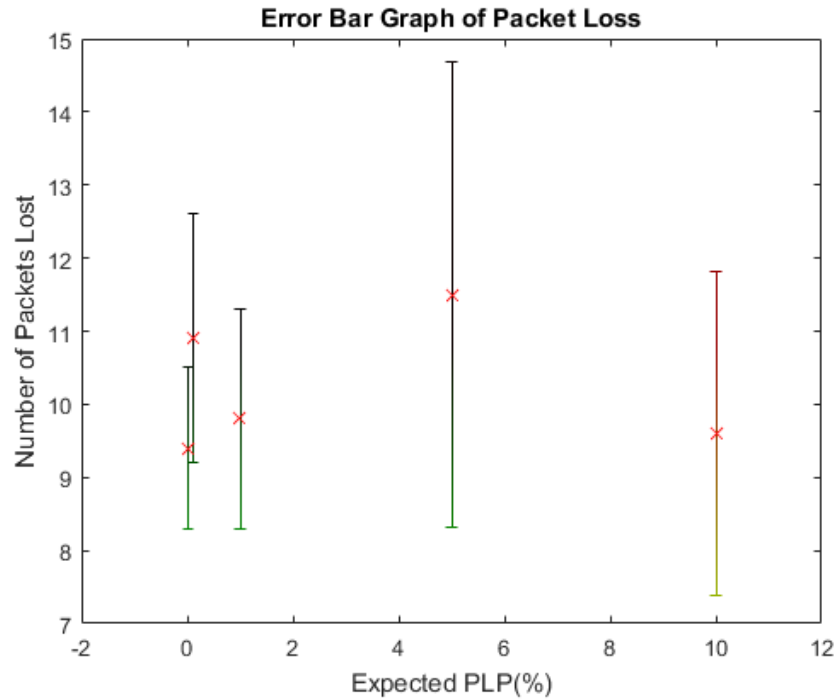


Figure 5.14: Error bar plot for the PLP values of 0.01%, 0.1%, 1%, 5%, and 10%

As the collected data was from 5 experiments only, i.e. 5 cases of the PLP, the error bars shown in the graph are widely spaced. More samples of data would bring the error bars very close to each other giving a well-defined pattern on the graph.

5.5 PLP Testing in Combination with MOS

When MOS was evaluated, only for VOD traffic was transmitted across the network. This was done as a precaution assuming more than one type of traffic transmitting across the network at the same time may lead to packet collisions, which could result in packet loss that was not planned from within NetEm. Consequently, the resulting packet loss could become higher than anticipated. Or, in the case of packet delays, it was suspected that a major discrepancy might be seen between the artificially added delay and the actual delay seen in the transmission. The experimental setup that was used for MOS evaluation is shown in Figure 5.15.

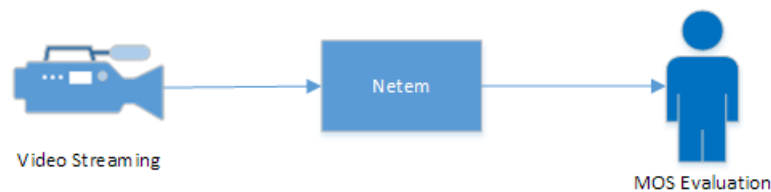


Figure 5.15: Case 1 - only VOD streaming on the link

Later on, it was decided that it would be beneficial to know if some other type of traffic on the same interface would further degrade the QoE. To explore this, packet probing was conducted at the same

time as the VOD streaming. The ping duration was kept the same as the duration of the video in the streaming. This experimental scenario has been illustrated in Figure 5.16.

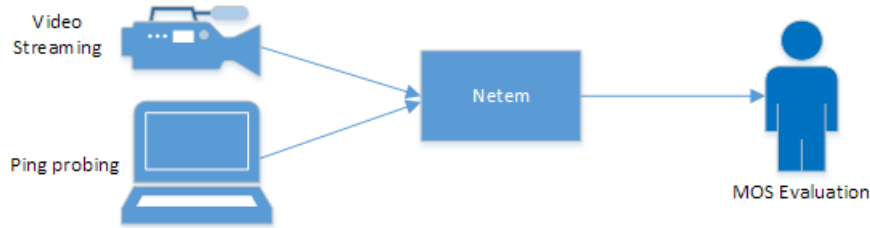


Figure 5.16: Case 2 - VOD streaming and ping probing simultaneously on the same link

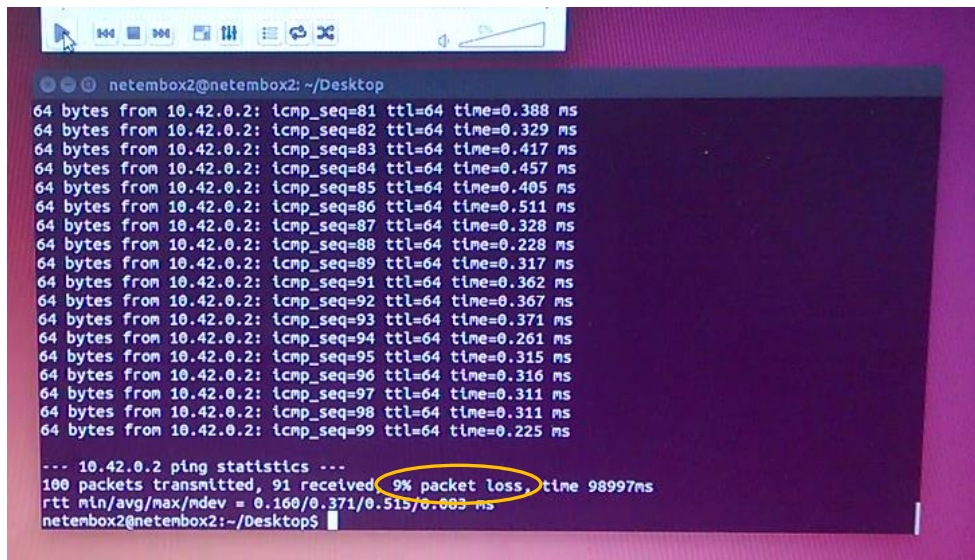
It was noted that having two types of traffic on the same network travelling through the same interface on server and client did not affect the QoE. For verification, the ping statistics collected through the ping probing in this experiment were analysed. From NetEm, a packet loss of 10% was implemented on the link used for this experiment. Figure 5.17 and Figure 5.18 show the ping statistics in both cases.

```

netembox2@netembox2: ~/Desktop
64 bytes from 10.42.0.2: icmp_seq=81 ttl=64 time=0.310 ms
64 bytes from 10.42.0.2: icmp_seq=82 ttl=64 time=0.316 ms
64 bytes from 10.42.0.2: icmp_seq=84 ttl=64 time=0.356 ms
64 bytes from 10.42.0.2: icmp_seq=85 ttl=64 time=0.306 ms
64 bytes from 10.42.0.2: icmp_seq=86 ttl=64 time=0.276 ms
64 bytes from 10.42.0.2: icmp_seq=87 ttl=64 time=0.367 ms
64 bytes from 10.42.0.2: icmp_seq=88 ttl=64 time=0.284 ms
64 bytes from 10.42.0.2: icmp_seq=89 ttl=64 time=0.343 ms
64 bytes from 10.42.0.2: icmp_seq=90 ttl=64 time=0.254 ms
64 bytes from 10.42.0.2: icmp_seq=92 ttl=64 time=0.288 ms
64 bytes from 10.42.0.2: icmp_seq=93 ttl=64 time=0.284 ms
64 bytes from 10.42.0.2: icmp_seq=94 ttl=64 time=0.323 ms
64 bytes from 10.42.0.2: icmp_seq=95 ttl=64 time=0.277 ms
64 bytes from 10.42.0.2: icmp_seq=96 ttl=64 time=0.213 ms
64 bytes from 10.42.0.2: icmp_seq=97 ttl=64 time=0.266 ms
64 bytes from 10.42.0.2: icmp_seq=98 ttl=64 time=0.298 ms
64 bytes from 10.42.0.2: icmp_seq=99 ttl=64 time=0.336 ms
64 bytes from 10.42.0.2: icmp_seq=100 ttl=64 time=0.334 ms

--- 10.42.0.2 ping statistics ---
100 packets transmitted, 89 received, 11% packet loss, time 98998ms
rtt min/avg/max/ndev = 0.213/0.306/0.400/0.043 ms
netembox2@netembox2:~/Desktop
    
```

Figure 5.17: 11% packet loss with only ping traffic on the link



```

netembox2@netembox2: ~/Desktop
64 bytes from 10.42.0.2: icmp_seq=81 ttl=64 time=0.388 ms
64 bytes from 10.42.0.2: icmp_seq=82 ttl=64 time=0.329 ms
64 bytes from 10.42.0.2: icmp_seq=83 ttl=64 time=0.417 ms
64 bytes from 10.42.0.2: icmp_seq=84 ttl=64 time=0.457 ms
64 bytes from 10.42.0.2: icmp_seq=85 ttl=64 time=0.405 ms
64 bytes from 10.42.0.2: icmp_seq=86 ttl=64 time=0.511 ms
64 bytes from 10.42.0.2: icmp_seq=87 ttl=64 time=0.328 ms
64 bytes from 10.42.0.2: icmp_seq=88 ttl=64 time=0.228 ms
64 bytes from 10.42.0.2: icmp_seq=89 ttl=64 time=0.317 ms
64 bytes from 10.42.0.2: icmp_seq=91 ttl=64 time=0.362 ms
64 bytes from 10.42.0.2: icmp_seq=92 ttl=64 time=0.367 ms
64 bytes from 10.42.0.2: icmp_seq=93 ttl=64 time=0.371 ms
64 bytes from 10.42.0.2: icmp_seq=94 ttl=64 time=0.261 ms
64 bytes from 10.42.0.2: icmp_seq=95 ttl=64 time=0.315 ms
64 bytes from 10.42.0.2: icmp_seq=96 ttl=64 time=0.316 ms
64 bytes from 10.42.0.2: icmp_seq=97 ttl=64 time=0.311 ms
64 bytes from 10.42.0.2: icmp_seq=98 ttl=64 time=0.311 ms
64 bytes from 10.42.0.2: icmp_seq=99 ttl=64 time=0.225 ms

--- 10.42.0.2 ping statistics ---
100 packets transmitted, 91 received, 9% packet loss, time 98997ms
rtt min/avg/max/mdev = 0.160/0.371/0.517/0.003 ms
netembox2@netembox2:~/Desktop$

```

Figure 5.18: 9% packet loss with data traffic from both, ping probing and VOD streaming, on the same link

The packet loss in each run was expected to be random around the actual activated packet loss of 10% from NetEm (as previously seen). However, if the packet loss had been affected with both types of traffic flowing simultaneously on the same link, it would show a significant increase in the packet loss in the latter case. But as seen in Figure 5.17 and Figure 5.18, the fact that the packet loss is lower in the case of data traffic from both applications on the same link shows that QoE was not affected when data packets from another application were travelling on the same link. This would be true for even higher number of applications transmitting data at the same time on the same network link as long as sufficient bandwidth is available on that link.

Likewise, having monitored the received VOD steaming across the network, it was noted that in both cases, there was not a significant change in packet delay values either, which if existed, could have potentially been because of possible congestion in the network buffer⁸ when two types of data traffic were present on the same link.

5.6 Investigating NetEm Jitter/Delay

In NetEm, delay can be added on an interface simply by specifying a value under the delay parameter as shown below:

```
# tc qdisc add dev eth0 root netem delay 100ms
```

This method of activating NetEm delay is the simplest and adds a fixed delay of 100ms on the specified interface. However, it adds a fixed delay on the interface NetEm is running on. This could be acceptable for basic network testing but certainly is not adequate for analysing network traffic against conventional network behaviours.

NetEm also facilitates the option to add a variation to a specified delay. As real-world networks show delay with variability, hence taking this approach to model a network delay could give a better view of

⁸ Congestion in the network buffer causes delay and differential delay leads to jitter.

real networks. This functionality can be achieved by adding delay and jitter with either of the two NetEm built-in delay distributions: Normal (Gaussian) or Pareto.

As this research involved a detailed study of QoE involving MOS evaluation, simply adding a fixed delay to the video stream would not reflect realistic network behaviours. More flexibility and control over the jitter and delay implementation was required to be able to mimic realistic modern networks. Modelling delay through a delay distribution gives a more precise control on the delay pattern. With that objective, it was decided to explore the use of Normal and Pareto distributions for modelling delay jitter as seen in real world network traffic patterns. To study the behaviour of these delay distributions, and to verify whether NetEm outputs the results for each of these as expected, ping probing was conducted for each of these distributions and results were put through some validations to verify their accuracy.

5.7 Theoretical Validation of Normal and Pareto Distributions in NetEm

When activating delay and jitter from NetEm with Normal and Pareto distributions separately, it was expected that on the achieved plots for each of these distribution:

- Having x-axis and y-axis both in log scale, if the plotted graph shows a straight line that indicates Power Law, e.g. Pareto;
- Having y-axis in log and x-axis in a linear scale, if the plotted line is straight it indicates an Exponential tail.

Graphs produced from the ping results verified this expected behaviour for both distributions Normal and Pareto. Figure 5.19 and Figure 5.20 show the trend of the stem graphs shaping up to a straight line in the case of both Normal and Pareto distributions. As the experiment was run on a lower number of packets in transmission, the graph trend is not a well-shaped straight line. The number of packets were increased, and experiments were done again. The graphs were then plotted to investigate the effect of probing over many transmitted packets.

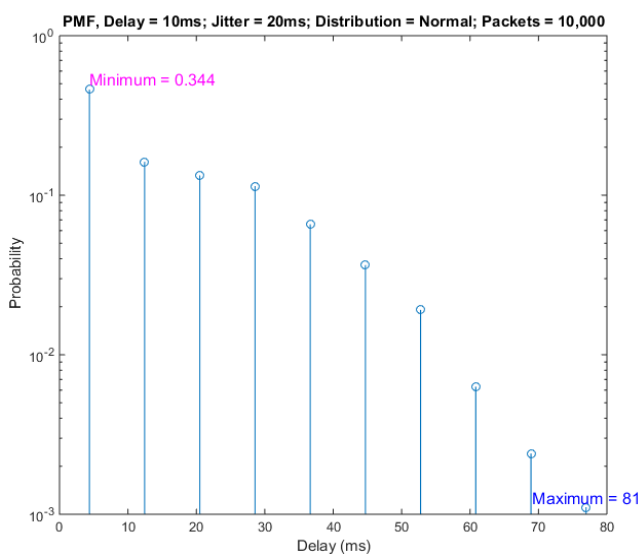


Figure 5.20: One-sided Normal delay distribution

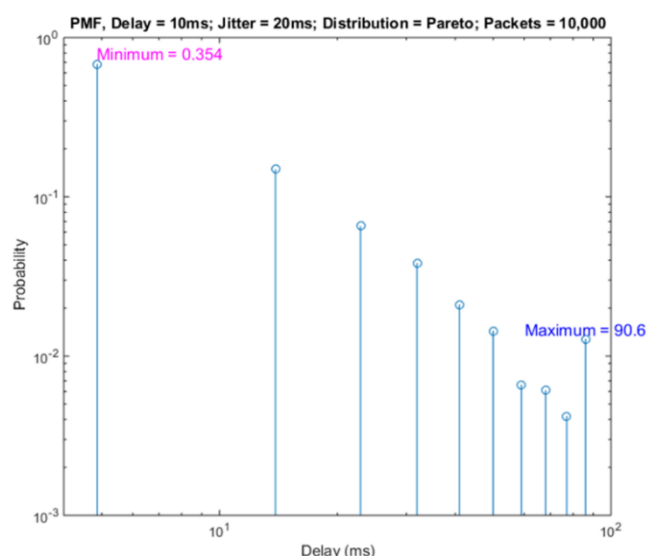


Figure 5.19: Pareto delay distribution

Figure 5.21 and Figure 5.22 show that data points denoted in the stem graph shape up to an improved plot which conforms to the shape of the Normal and the Pareto as outlined above in the hypothesis bullet points: the exponential tail has the x-axis as a linear scale and the y-axis with a log scale, and the Pareto has both the x-axis and the y-axis on log scales. As the delay values have been broken down into segments of 10ms, the graphs show discrete points instead of a continuous probability density function.

It is also noted, in the case of the Pareto distribution for the chosen datasets plotted on the graph, the very last data point always shoots up along the y-axis. This is because all data points beyond this get added into the last value, hence a lump is seen for the last data value as in the graph in Figure 5.19.

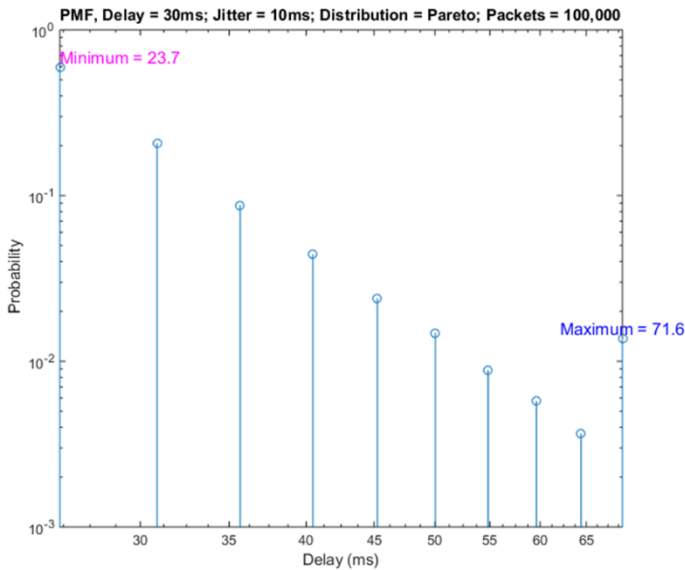


Figure 5.21: Pareto distribution with higher transmitted packets, 100,000

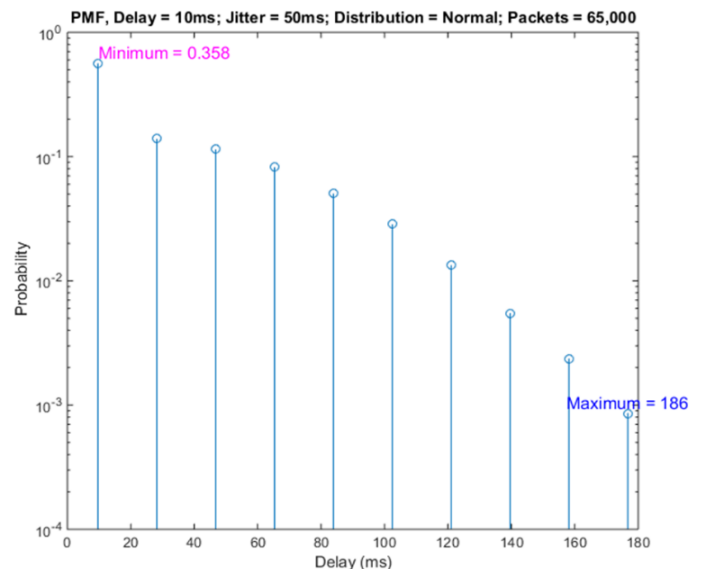


Figure 5.22: Normal delay distribution with higher number of transmitted packets, 65,000

5.8 Jitter Implementation with Normal and Pareto Distributions

Although the Normal distribution does allow a jitter implementation, it does not allow an implementation of delays larger than the mean delay added to four times the standard deviation.

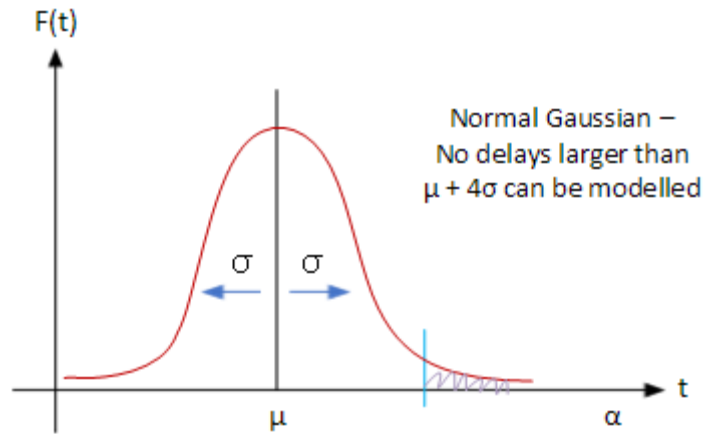


Figure 5.23: A Normal (Gaussian) distribution

Hence delay values higher than the mean delay added with four times the standard deviation⁹ will be truncated. Therefore, the jitter model achieved with Normal distribution was not close to what would be expected following the delay pattern in actual Internet traffic and the literature suggested that Pareto distribution could give a better jitter representation for an end-to-end delay model [41] [42] [43].

Hence, it was considered that the use of the Pareto distribution for adding jitter could potentially give a more appropriate jitter model since the shape parameter alpha can be used to control the shape of the curve targeting a specific standard deviation for a jitter delay.

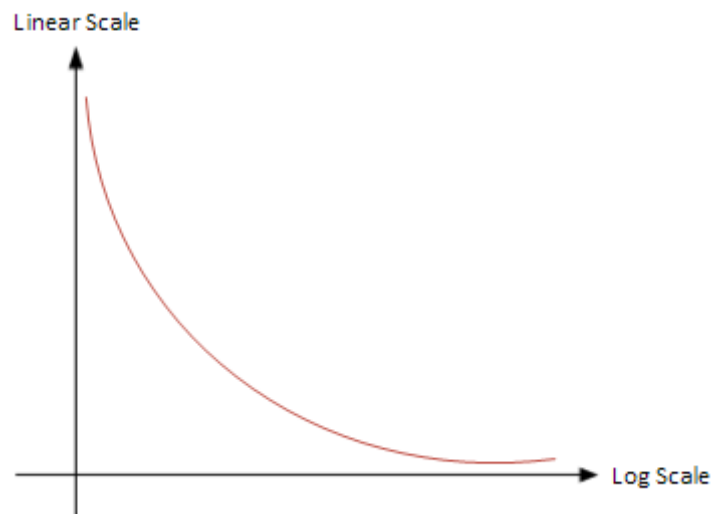


Figure 5.24: A Pareto distribution

It was then decided, as agreed by the researchers in [41] [42] [43], that the use of the Pareto distribution could be a better choice to model real-world traffic behaviours for studying end-to-end delays. A further investigation was then conducted to analyse the behaviour of the Pareto distribution as to what factors out of delay and jitter govern the shape of the Pareto.

⁹ As the use of a range of standard deviations was required including large standard deviations and small ones.

5.9 Activating Normal and Pareto distributions on a NetEm Interface

When modelling delay and jitter with the Normal (Gaussian) distribution in NetEm, the required delay and standard deviation/jitter can be input directly. For example, for a mean delay of 100ms and jitter¹⁰ of 20ms, we would activate this on NetEm interface as follows:

```
# tc qdisc change dev eth0 root netem delay 100ms 20ms distribution normal
```

However, for a Pareto distribution, the mean delay must be calculated first in order to use this in the NetEm command. This is because unlike the Normal distribution where the μ (the mean) and σ (standard deviation) are separate so they can be set individually, with the Pareto distribution, the standard deviation and mean both depend on the shape parameter α . So essentially, we are setting the standard deviation to achieve a certain value for α , which implies we will then have the mean.

In this thesis, jitter was always parametrized by using the standard deviation σ of delays. Derivation of the mean from the standard deviation (in the context of jitter in this thesis) for Pareto can then be calculated.

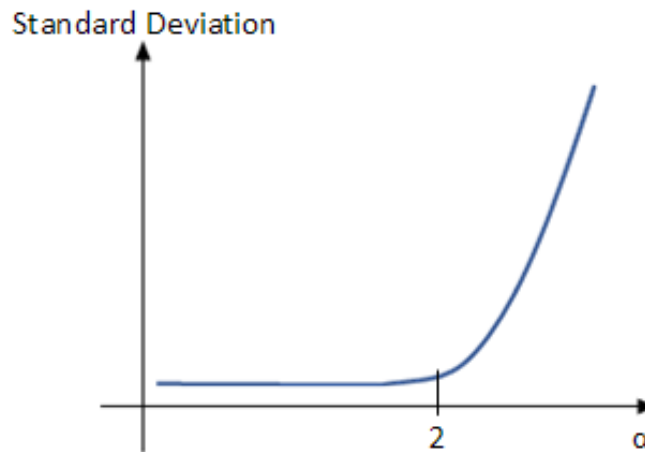


Figure 5.25: Derivation of Mean from the standard deviation

Setting the distribution with the shape parameter α .

Assuming $x_{min} = 1ms$

$$mean = \frac{\alpha}{\alpha - 1} = \frac{1}{1 - \frac{1}{\alpha}} \quad (5.12)$$

$$var = \frac{\alpha}{(\alpha - 1)^2(\alpha - 2)} \quad (5.13)$$

In order to get the variance, if we substitute $\alpha = 2.01$ in equation (5.13), the calculated variance will be $var = 201$. However if α is less than 2, e.g. 1.99, the variance will come out to be ∞ (infinity) which is not useful, hence not acceptable.

¹⁰ Because standard deviation is being used as jitter throughout this thesis.

Now because the scenarios under focus features α just a little bigger than 2.

This can be simplified to:

$$var = \frac{\alpha}{(\alpha - 2)} \tag{5.14}$$

$$(\alpha - 2).var = \alpha \tag{5.15}$$

$$\alpha.var - 2.var = \alpha \tag{5.16}$$

$$-2.var = \alpha (1 - var) \tag{5.17}$$

$$\alpha = \frac{-2.var}{(1 - var)} \tag{5.18}$$

s = standard deviation = var^2 , so:

$$\alpha = \frac{-2.s^2}{(1 - s^2)} = \frac{2.s^2}{(s^2 - 1)} \tag{5.19}$$

$$mean = \frac{1}{1 - 1/\alpha} = \frac{1}{1 - 1/\frac{2.s^2}{(s^2 - 1)}} = \frac{1}{1 - \frac{(s^2 - 1)}{2.s^2}} \tag{5.20}$$

It was then seen from the derived formula in equation (5.20) that for the Pareto distribution α is set only by the mean value, while the second input value ("jitter" or "standard deviation") provides only the right-hand limit to the width of the distribution. For Pareto:

$$largest\ delay = mean + (4 \times right_hand_limit) \tag{5.21}$$

The mean was then calculated with the help of the formula in equation (5.20) for each standard deviation value from the range used in previous chapters. The range of standard deviation values that were used in the previous experimentation are 0ms, 5ms, 10ms, 80ms, 100ms.

The mean that was calculated from each of the above standard deviation value for inputting in NetEm is shown in Table 5.7.

Table 5.7: Mean calculated for each standard deviation value

Jitter/Standard Deviation	Mean
0 ms	0 ms
5 ms	1.923076923 ms
10 ms	1.98019802 ms
80 ms	1.999687549 ms
100 ms	1.99980002 ms

Now this calculated mean for each case with a different standard deviation can be inputted into NetEm command like so:

```
# tc qdisc change dev <interface> root netem <mean delay> <jitter>
distribution <dist-type>
```

The “jitter” value for inputting in NetEm was initially chosen to be 500ms¹¹ and it was decided that this could be varied if required.

So, the NetEm command with the calculated mean for a given standard deviation, the standard deviation being 5ms (Table 5.7) in this case, along with the jitter (500ms) value became:

```
# tc qdisc change dev eth3 root netem 1.923076923ms 500ms distribution pareto
```

When the decimal mean value given above in NetEm command were inputted in NetEm, NetEm rounded it to one decimal place making it 1.9ms. That raised the concern whether this behaviour of NetEm would give a unique streaming output for each of the above mean values in Table 5.7. To investigate this further, it was analysed whether NetEm was accepting the inputted value with full precision and then rounding to two figures only for printing back to the terminal, or NetEm only allows two significant figures internally hence rounds it to two figures before even accepting it as an input.

It was decided to firstly check NetEm documentation to establish if it mentioned this in any way, which did not provide any meaningful information. The second approach to investigate the concern was to draw graphs with different mean delay values and then compare those graphs to evaluate the precision. This way, the graphs would show if NetEm rounded the mean value before applying it to the interface, or rounded it to two figures after its application only for notifying the user in the terminal.

To proceed with the second approach to solving the problem, two different scenarios were formed where the mean delay values were chosen to be different. These two mean delay values were taken for jitters 10ms and 100ms respectively (refer to Table 5.7). These two scenarios, based on two different mean delays were expected to give very different delay distributions in a transmission of a large number of packets, 50,000 ping packets in this case. If, however, the achieved results are not different, then the use of Pareto distribution would not give a full control over mimicking real-world networks, as it will mean that NetEm allows only two significant figures as input. To verify the expected outcome, the ping probing for 50,000 packets was conducted in each case.

The obtained graphs from the experimentation incorporating these two scenarios are given in

Figure 5.26 and Figure 5.27.

¹¹ The jitter value of 500ms was chosen to start off because this is a mid-range value, being not too low or high.

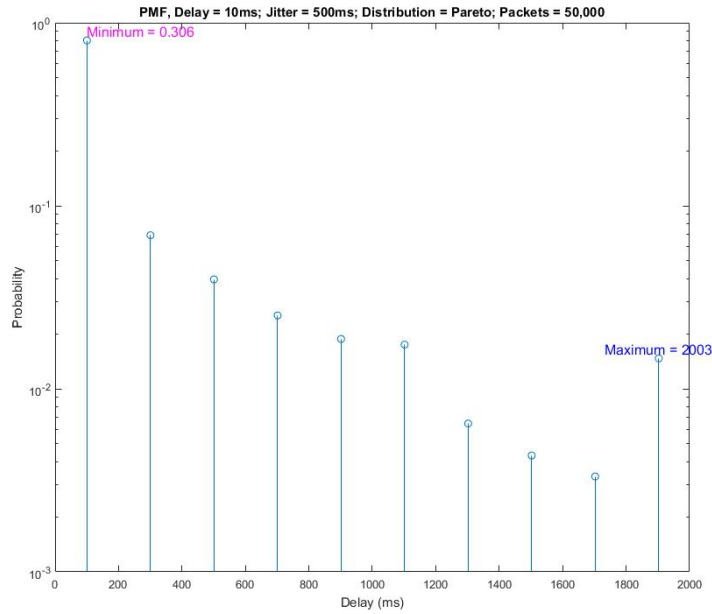


Figure 5.26: Graph to analyse Pareto distribution behaviour in the presence of 10ms mean delay

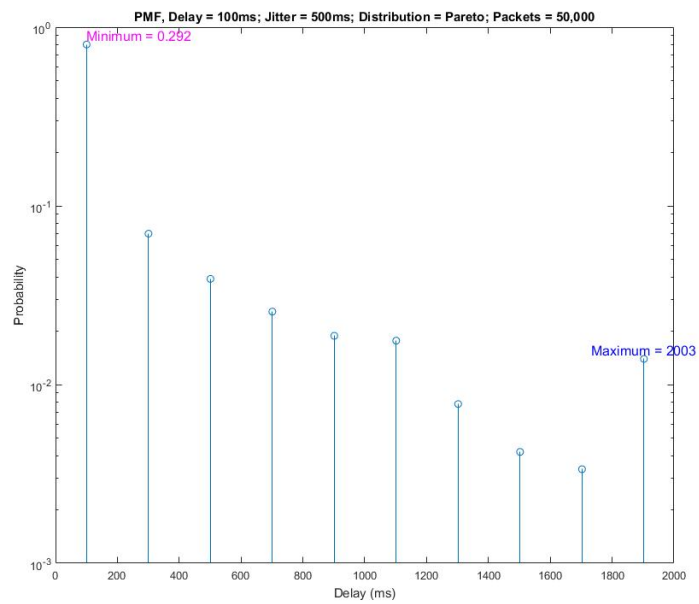


Figure 5.27: Graph to analyse Pareto distribution behaviour in the presence of a higher, 100ms mean delay

The graphs produced from the two different scenarios did not show different patterns on the achieved plots. This showed that NetEm simply rounds the mean value to a two significant figures value, which in both cases above was 2.0ms.

As the shape parameter α for a Pareto distribution is very sensitive to delay, to model a jitter model with a desired delay requires a high precision of α input. Due to this delay-dependant sensitivity, a delay distribution with a value $\alpha = 2.01$ would give a different shape as compared to a delay shape when $\alpha =$

2.0001. Furthermore, $P(\text{delay} > T)$ also varies hugely for a small change in α . The fact that the graphs in

Figure 5.26 and Figure 5.27 are identical verifies that NetEm does not allow an input value with a higher precision than 2 decimal places. Therefore, it can be concluded that the use of NetEm built-to-type Pareto distribution is effectively impractical for modelling network delay distributions with any decent degree of precision.

Although it is not possible to have very precise control over the use of Pareto distribution we can still use the built-to-type distributions the way NetEm instructs to study networks, which has been done in [123] where the researcher has not specified how they activate Pareto in NetEm. They have just mentioned the mean to be 20ms or 10ms, so it seems they are inputting it directly in NetEm like so:

```
# tc qdisc change dev eth0 root NetEm delay 100ms 20ms distribution pareto
```

This way of inputting parameters will use NetEm's built-in Pareto distribution, which will only reflect the real-world network scenarios very vaguely. Also, the author of [123] does not seem to be calculating the mean from the formula used in this thesis. If that were the case, the delay value that they used would have been in decimal format. Nonetheless, the Pareto formula in (5.19) describes the Pareto parameters that govern the shape of the Pareto.

5.10 Experimentation with Jitter Using TCP and UDP Protocols

TCP, as described in Section 3.4.1 on page 44, is a connection-oriented and robust transport layer protocol. TCP offers a reliable delivery of byte-stream data. This reliability and robustness offered is determined by various mechanisms. For example, for ordering the received packets/segments and to detect duplicate or missing packets, a sequence number is used. Error detection is facilitated with the use of checksums, and timers/acknowledgements are used to adapt the transmission rate to the condition of the link in the presence of loss and delay. However, initiating the connection between the sender and receiver, and then to maintain the transmission with all aforementioned mechanisms, TCP can be an overkill for delay-sensitive applications. UDP, due to its simple transmission principles relying on best effort but without offering a guaranteed packet delivery, can be a better choice for real-time applications. The developers of VLC media player, used for network video streaming in this research, also agree with this (direct quote) [124]:

“TCP is not appropriate for many applications as for example, real-time applications. They often don't need, and will suffer from TCP's reliable delivery mechanisms. In those types of applications it is often better to deal with some loss, errors or congestion than to try to adjust for them. Example applications that do not typically use TCP include multimedia streaming, real-time multiplayer games and voice over IP (VoIP). In many cases, the User Datagram Protocol (UDP) may be used in place of TCP when just application multiplexing services are required.”

VLC media player uses TCP and UDP configurations that have been set up as part of the operating system installation. So, for Ubuntu 14.04, which was used in this research, the TCP version installed in the OS kernel is the NewReno as stated in Ubuntu manual [125]:

“It is an implementation of the TCP protocol defined in RFC 793, RFC 1122 and RFC 2001 with the NewReno and SACK extensions. It provides a reliable, stream-oriented, full-duplex connection between two sockets on top of IP, for both v4 and v6 versions. TCP guarantees that the data arrives in order and retransmits lost packets. It generates and checks a per-packet checksum to catch transmission errors. TCP does not preserve record boundaries.”

Similarly, for UDP, Ubuntu manual describes as follows [126]:

“This is an implementation of the User Datagram Protocol described in RFC768. It implements a connectionless, unreliable datagram packet service. Packets may be reordered or duplicated before they arrive...”

Proceeding with experimentation, VOD was streamed over both transport protocols: TCP and UDP. To introduce imperfections on the network link, jitter, by itself, was added to the video traffic stream going out of the server to the client computer. Jitter is defined as the standard deviation of the delay distribution as outlined in [127]. Table 5.8 shows the MOS score recorded by the participants for both Internet protocols against the applied jitter.

Table 5.8: Jitter only experimental results with MOS values

Jitter (ms)	MOS	
	UDP	TCP
0	5	5
5	5	5
10	4	5
80	2	5
100	1	4

Over a real-world network link where traffic from a range of applications is present, UDP performs better for video streaming applications. However, in this particular test, the network link did not have real world network traffic from any other application, hence the real capabilities of TCP did not get exploited to the full extent, i.e. TCP protocol did not have to do much which would delay packets in an attempt to achieve 100% packet delivery retaining packet order. If the network link under experiment had real multi-source traffic then, in the process of attempting the re-transmission of any lost packets, TCP would result in delaying video packets which would degrade the QoE. So, in a comparison, TCP would have got a lower MOS score and UDP would have performed better for streaming a video, as expected.

In this particular test scenario, it was noted that when transporting the video traffic on UDP, the video quality of the streamed video degraded as the amount of added jitter went beyond 5ms. So, when 10ms jitter was present, the artefacts were noticed in the video and the MOS reflected that. When the added jitter reached 100ms, the video was unwatchable; hence the MOS dropped massively and was recorded to be a 1 (see Figure 5.28).

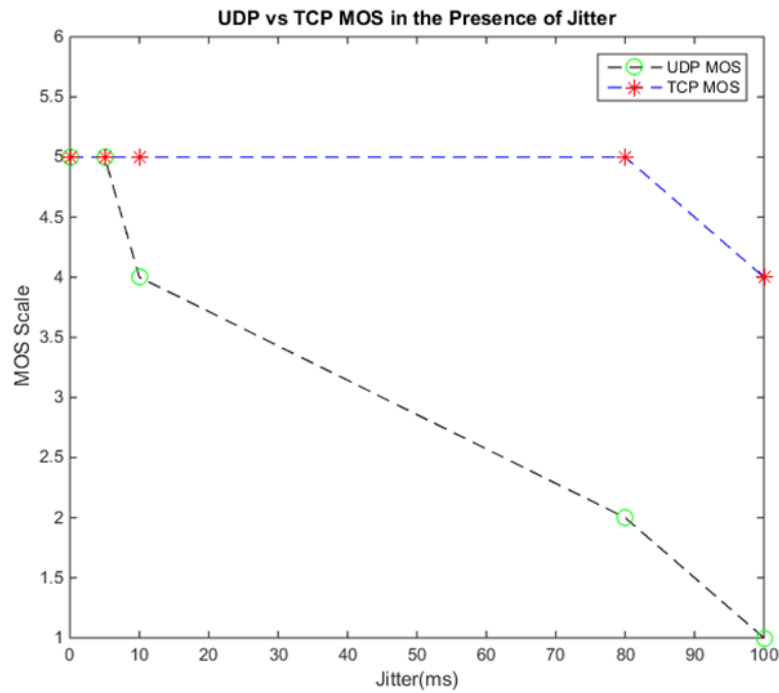


Figure 5.28: QoE experimentation on UDP and TCP protocols in the presence of NetEm implemented Jitter

5.11 Experimentation with Packet Loss Using TCP and UDP Protocols

The same video was streamed again under this new scenario where only packet loss was applied on the video stream for both UDP and TCP. The packet loss was added in increments starting with 0.01% and going up to 0.5%. Table 5.9 shows this incremental increase in packet loss along with the experimental results.

Table 5.9: Experimentation with VOD using UDP and TCP protocols with loss only implementation

PLP (%)	MOS	
	UDP	TCP
0	5	5
0.01	4	5
0.025	4	4
0.05	3	4
0.1	2	3
0.25	1	2
0.5	1	1

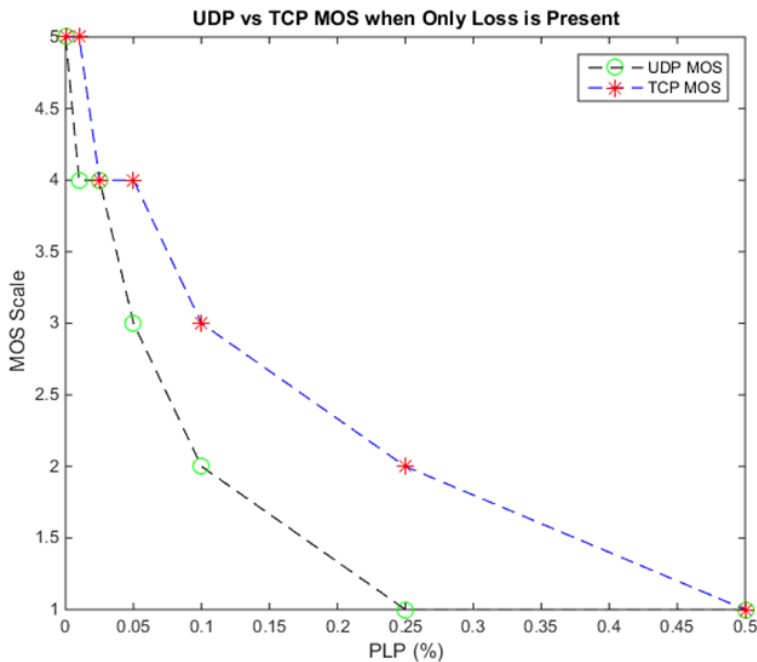


Figure 5.29: QoE experimentation on UDP and TCP protocols in the presence of NetEm implemented loss

It was noted that in the presence of packet loss, video quality is affected regardless of which Internet protocol is in use whether UDP or TCP.

However, UDP was affected earlier than TCP and the MOS dropped for a packet loss as low as 0.01%. The video became unwatchable when packet loss reached 0.25%. Conversely, with TCP, the MOS dropped to a 4 when the packet loss reached 0.025%, which is later than the case of UDP. With a further increase in the PLP, the video was rated to be 'very annoying' at 0.5% packet loss, which is twice as high PLP as in the case of UDP.

5.12 Custom Delay Distributions for Jitter and Packet Loss

In this set of experiments, custom delay distributions were created and called from within NetEm to be able to produce (or at least bring close to) a delay pattern that is found in real world networks. For instructions on how to design such delay distribution tables and then implement the delays from within NetEm, refer to the Section 4.11.1.

Having completed the experimental setup, the experimentation was started by combining the loss and jitter parameters and the same video was streamed again over UDP and TCP transport protocols. From the collected results, it was noted that, in the presence of different jitter levels in the video stream, the TCP MOS did not change as rapidly as it did in the case of UDP. UDP showed significant degradation in the MOS with an increase in jitter. It was seen that, regardless of the low level of implemented packet loss on the stream, for a jitter of 100ms the MOS for UDP video stream became 1 rendering the video unwatchable which corresponds to 'very annoying' part of the MOS scale (see Figure 5.33).

For TCP, as the implemented jitter did not primarily affect the video quality of the streamed footage, the points where TCP MOS would catch one's attention was the PLP range of 0.1% through 0.5% in the video stream. In the case of a low added jitter such as 5ms, the MOS firstly dropped from 5 to 3 at 0.1% PLP, then the MOS dropped to a 2 when the PLP was 0.25% (Figure 5.30). For the jitter values higher than 5ms, the MOS dropped to a 1 when the PLP reached 0.25% (see Figure 5.31, Figure 5.32, and Figure 5.33).

It was noted that for a low jitter of 5ms, TCP QoE is significantly affected when the packet loss reached 0.5% (Figure 5.30), and beyond that point, the video became unwatchable. For higher jitter values of

10ms, 80ms, and 100ms, even though TCP performs better than UDP to retain the QoE, it still dropped to a 1 MOS rating when the applied PLP reached 0.25%. At that point and beyond, the video becomes unwatchable (Figure 5.31, Figure 5.32, and Figure 5.33).

Table 5.10: MOS experimentation with UDP and TCP protocols having implemented both loss and jitter simultaneously

Jitter (ms)	PLP (%)	UDP MOS	TCP MOS
5 (5e-3)	0.01	5	5
5	0.1	3	3
5	0.25	1	2
5	0.5	1	1
5	1	1	1
5	10	1	1
10 (1e-2)	0.01	4	5
10	0.1	2	2
10	0.25	1	1
10	0.5	1	1
10	1	1	1
10	10	1	1
80 (8e-2)	0.01	3	5
80	0.1	2	2
80	0.25	1	1
80	0.5	1	1
80	1	1	1
80	10	1	1
100 (1e-1)	0.01	1	5
100	0.1	1	2
100	0.25	1	1
100	0.5	1	1
100	1	1	1
100	10	1	1

Figure 5.30 to Figure 5.33 show the graphs for a combination of jitter and PLP values for both UDP and TCP.

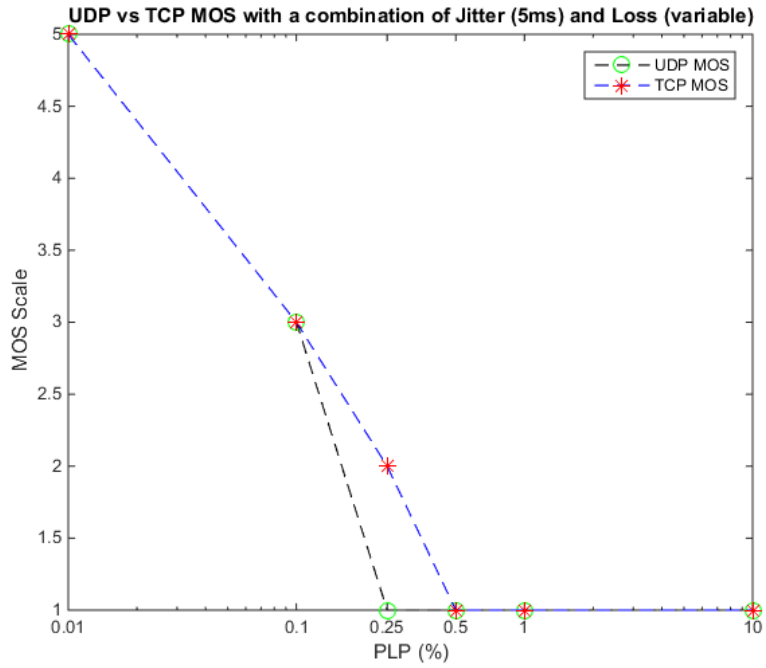


Figure 5.30: UDP and TCP MOS comparison graph with loss and jitter applied simultaneously, jitter = 5ms, and loss is variable

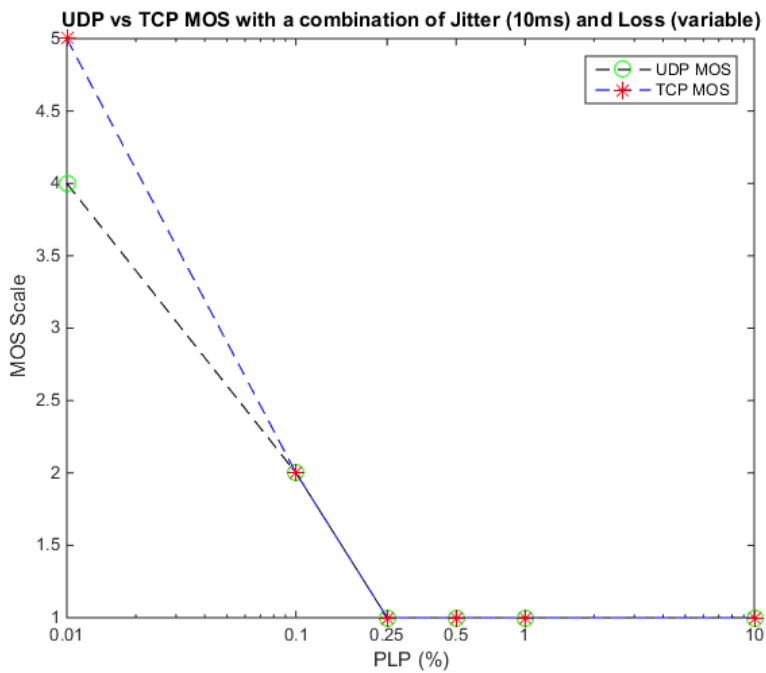


Figure 5.31: UDP and TCP MOS comparison graph with loss and jitter applied simultaneously, jitter = 10ms, and loss is variable

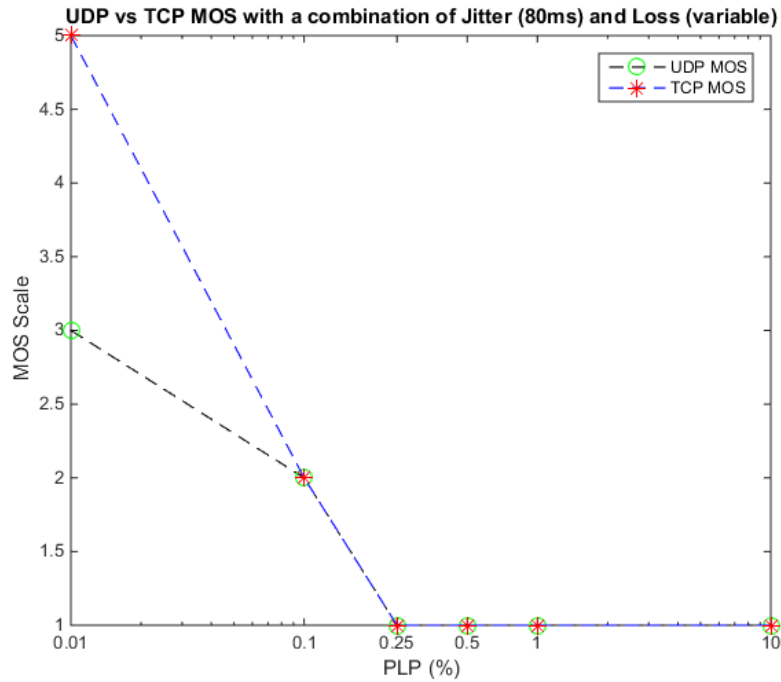


Figure 5.32: UDP and TCP MOS comparison graph with loss and jitter applied simultaneously, jitter = 80ms, and loss is variable

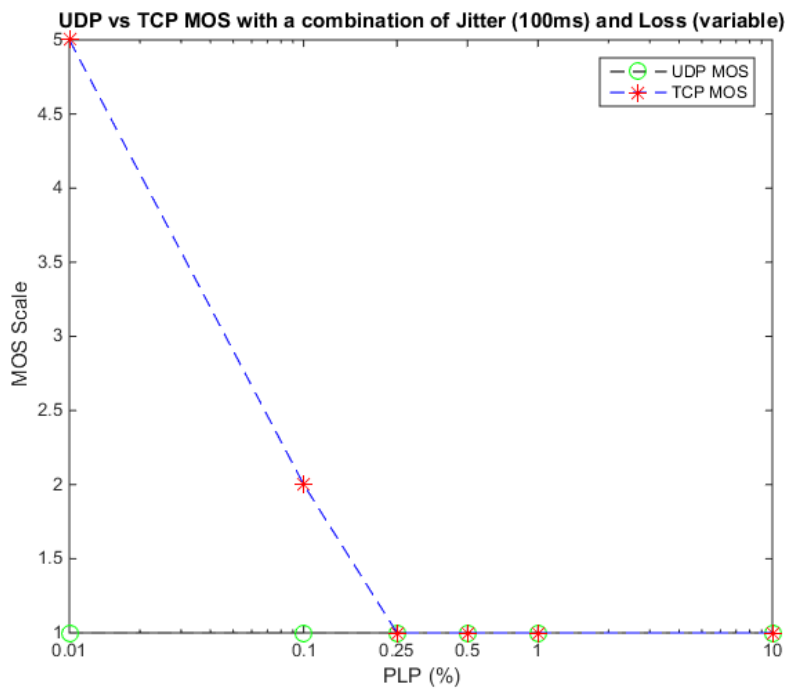


Figure 5.33: UDP and TCP MOS comparison graph with loss and jitter applied simultaneously, jitter = 100ms, and loss is variable

Looking at the MOS trend for UDP, it was seen that in all video streams with different jitter levels, when the loss reached 0.25% it severely affected the QoE and the evaluated MOS was always 1 regardless of what amount of jitter was applied. However, with an increase in jitter level too, the MOS dropped to an

even lower MOS score and quicker than the previous case where a lower amount of jitter had been applied for the same level of PLP. This behaviour was witnessed for the jitter levels 10ms, 80ms, and 100ms.

5.12.1 Effect of Standalone Jitter and Loss on TCP vs UDP Protocols

From the experimentation in this Chapter, UDP was seen to be very sensitive to jitter. The key points that significantly affect the QoE of a UDP stream are when the jitter is 10ms and 100ms. As there was more than one user involved in the research, it was noted that when jitter was 10ms the recorded MOS was 4 on average. But looking at the MOS score from the participants on an individual basis, it was seen that users rated a lower individual MOS score based on the video quality that had started to degrade. With an increase in the implemented jitter being 100ms, the quality of the video was very poor and very annoying for the user. Hence at this point, all participants rated the video with a MOS score of 1 in every run of the experiment.

The QoE of the UDP stream is also affected by packet loss. Looking at the graph in Figure 5.29, it is noted that the quality of the video started to degrade when the applied packet loss reached 0.025%, which was reflected in the average recorded MOS of 4 for UDP at that point in the graph. When the PLP reached 0.25%, the evaluated MOS levels out to a 1. The QoE against this loss benchmark is of vital importance, which can be seen in service level agreements of typical network metrics guaranteed to network customers in [4] and [128].

TCP on the other hand, was not as severely affected by jitter, but loss did show a direct impact on the QoE of the TCP video stream. The initial degradation in the QoE of video becomes apparent when the applied packet loss reaches 0.1%. As the implemented loss is increased and reaches 1%, TCP QoE worsens to an extent that the video becomes unwatchable; hence the MOS drops to a 1.

5.12.2 Effect of Combined Loss and Jitter on Network Traffic QoE

The implementation of both loss and jitter on video traffic produces a more realistic scenario of real world effects on VOD. This series of experiments show that the collective effect of loss and jitter significantly affects the QoE of VOD. Another outcome from these experiments was that the choice of protocol being used whether UDP or TCP, also has an impact on the QoE. With the collected results, it was noted that UDP QoE was more severely affected across all the combinations of loss and jitter. The MOS collected with the application of TCP, however, was not as badly affected and was witnessed to be almost consistent for all tested combinations of loss and jitter. The MOS collected when using the UDP protocol showed that UDP has a lower tolerance to jitter and loss, and that having even a very low implemented loss such as 0.01%, the video can become unwatchable in the case of UDP.

In the loss / jitter independent experiments MOS was captured to be 5 for TCP and 4 for UDP when the implemented jitter was 10ms (Table 8.1). Similarly, the MOS was 2 when the artificially added loss was 0.25% as a standalone parameter (Table 8.2). However, when both of these values for loss and jitter are applied at the same time, the MOS dropped to a 1 (Table 8.3). This sudden drop in MOS upon adding both network phenomena in this set of experiments outlined an important discovery – jitter and loss are not linearly additive, knowledge of which can be significantly important to network operators.

5.13 Chapter Summary

With the use of this new NetEm testbed, the research in this chapter was conducted on a range of network scenarios. Initially, in the first part of the experimentation described in this chapter, it was shown that in a controlled network environment, packet delay is independent of packet loss. In other words, regardless of the level of packet loss added to a network traffic stream, the delay seen across the network will not vary with the applied packet loss. If there is a change in the delay experienced by data packets across the network, it will not be due to the fact that packet loss has been implemented on the traffic stream, but it will be as a result of other factors such as network congestion or a bad link. Results showed that the delay found on the network was about the same for all levels of applied packet loss: 0.01%, 0.1%, 1%, 5%, and 10%. Refer to Section 5.3 on page 69 for details. This knowledge is useful for network researchers that wish to study packet delay and packet loss in a laboratory environment. They can, for example, focus on studying the effect of packet loss without having to go through various additional tests to determine whether their work will be affected by the presence of packet delay across the network. In this way, time and effort can be utilised working on the main concept without having to be concerned about parameters that may not be of great interest directly, such as the network delays in a study of packet loss.

The experimentation then examined if a NetEm-powered testbed accurately corresponds to the input instructions. For instance, does the amount of inputted packet loss reflect in the resulting behaviour of network traffic at the output. For this reason, experiments were run having different levels of packet loss. The traffic at the output was then analysed and compared against the analytical equivalent of a network with same individual levels of packet loss. Refer to Section 5.4 on page 74 for this part of experimentation. It was seen that there was a very close match between the analytical packet loss and the actual applied packet loss on NetEm.

Next, two types of traffic were transmitted on the same network link to investigate whether this would result in a packet loss higher than the actual packet loss applied at the NetEm interface. Video streaming was used along with ping probing on the same network link and it was noted that the packet loss in this case was not any worse than that seen when only video streaming was on this link. Details on this can be seen in Section 5.5 on page 84. It was then concluded that if the network has sufficient bandwidth, a higher level of packet loss may not be seen only as a result of transmitting network data from more than just one type of application.

This verified that the network experimentation testbed was operating as expected and ready to be used for advanced level network studies covering real world network scenarios.

The aim of the work done in the second part of this chapter was to further investigate this level of a 'better' approximation of delays through delay distributions. Firstly, the Normal distribution was studied to determine whether it gave real-world-like delay distribution. It was noted that the Normal distribution was not a good match to the expected output jitter patterns. This because the Normal distribution was unable to implement delays larger than the mean delay plus four times the standard deviation. This means that the tail of the distribution will be chopped off and not taken into account, which essentially means losing accuracy. For this reason, the Pareto was thought to be a better choice for modelling end-to-end delay with a 'realistic' view of jitter, which is also agreed upon by the researchers in [41] [42] [43]. The Pareto distribution was then explored. It was noted that the shape parameter that governs the shape of Pareto is very sensitive. As a result of this, controlling the shape to

model a certain level of jitter is near impossible. Experiments were then done to obtain delay distributions for a range of standard deviation values. However, due to the sensitivity of the Pareto shape parameter, the calculated mean delays for a range of standard deviations had long decimal values with a very small difference between them. For instance, the mean values for 80ms and 100ms standard deviations were 1.999687549ms and 1.99980002ms respectively. It can be seen that, in both mean values, counting from left to right the first few decimals are the same. Although a very small difference between the two, with careful consideration being able to input these mean values would output delay distributions based on 80ms and 100ms standard deviations which could then be used in NetEm to implement jitter patterns. However, it was not possible to input values with such large decimal precision in NetEm. This is because NetEm does not accept any input with higher than 2 significant figures. It was concluded that in NetEm the Pareto option was seen not to give a full control over the distribution shape, which would thus not accurately implement the delay patterns for a specific standard deviation in NetEm.

Further experiments in the third section of this chapter were done with loss and jitter using a particular transport protocol: UDP or TCP. Bespoke delay distributions were used in NetEm to model jitter in packet networks. Firstly, experiments were done to assess the effect on QoE of using UDP versus TCP with applied jitter. It was noted that in this particular experimental set up, in the presence of jitter, the QoE of network traffic that is being transported over UDP gets affected more severely as compared to the use of the TCP protocol. In the next series of experiments the performance of UDP versus TCP was studied in the presence of network packet loss. The UDP stream showed a significantly degraded QoE when packets are dropped in the transmission, and upon reaching a packet loss with a PLP of 0.25% the participants rated the video to be of 'bad quality' aligning with the user perceptive behaviour of the video being 'very annoying'. Service providers are aware of the impact of this level of PLP on network QoE, and as a consequence, typically network operators guarantee their customers a service where network will not go beyond such a level of PLP [4] [128]. In the case of network jitter, although TCP traffic was not affected as severely as UDP, in the presence of packet loss, the TCP stream is also seen to have a direct impact on the QoE the participants recorded. Further experiments were then conducted with the application of loss and jitter as a combination and with a custom delay distribution. It was noted that a network, with both jitter and packet loss present, results in the VOD QoE to be worsened a lot more quickly and more severely. TCP performs slightly better than the UDP due to its underlying mechanisms to combat packet loss.

Although TCP can be a better choice over UDP in many network applications where loss and jitter are of focus, the implementation of TCP is not as simple as UDP and comes with many additional challenges. For example, its initial connection setup for synchronisation of sources for handshake-based transmission can introduce set up complexity. Furthermore, data transmission will not start until both the source and destination hosts have formally established a connection – this delay could affect the QoE of delay sensitive applications. Therefore, TCP is not ideal for applications such as online gaming and skype. UDP, as compared to TCP, is more sensitive to jitter and loss but useful for applications where transmission speed has a higher priority and delay or loss of a few packets is not considered a significant drawback.

Chapter 6 Evaluating MOS for Varying PLP Over a Range of Different Age Groups

6.1 Chapter Introduction

In this chapter, QoE of VOD is studied by evaluating a MOS score from volunteers in different age groups who were invited to take part in the study of QoE. The videos were played and streamed across the network allowing the volunteers to record their MOS based on the quality of the video they experienced. The videos included a Jaguar car advert, The Hobbit movie trailer, and some slow-paced videos of animals and birds. From the survey results, it was noted that the youngest people in the 10-18 age group are most demanding when it comes to QoE. It was seen that 10-18 years old participants, due to their modern usage patterns, are likely to get frustrated even when video quality is degraded slightly. The participants from other age groups, on average, showed a little more tolerance when assessing the QoE. Details of which are given in the subsequent sections.

6.2 Mean Opinion Score (MOS) to Assess QoE for Video on Demand (VOD)

Participants were invited to take part in this QoE study by providing their MOS over a range of QoS scenarios. A survey sheet was produced that was presented to the research participants. This sheet described what the research was and gave instructions for participants to record their MOS. To see a copy of the survey sheet, please refer to the Appendix H on page 212.

The set of collected MOS given in this chapter is with UDP streaming. UDP protocol, as compared to TCP, loads more quickly on the recipient computer, but in return, shows the streaming output with a lower quality compromising QoE. When using TCP, VOD may take slightly longer to load the video in the beginning. However, the overall quality of the video being streamed with TCP is better for the same amount of artificially added jitter, delay, or loss. This is because TCP attempts to re-transmit any lost packets utilising its built-in mechanism. UDP protocol on the other hand, as compared to TCP, loads

more quickly on the recipient computer but shows the streaming output with a lower quality whilst compromising on QoE. This is because UDP does not keep track of any lost packets. Hence, rather than waiting for packet delivery acknowledgement, it continuously transmits packets which makes it quicker than TCP.

MOS was recorded by participants from different age groups. The defined age groups were 10-18 years, 19-30 years, 31-45 years, 46-65 years, and over 65 years old. The same video with the same range of added packet loss was shown to all the participants. As children and adults in different age groups perceive visuals differently [87] [129] [130], hence it was expected that participants of different ages may perceive the QoE slightly differently for the same video streaming. Table 6.1 shows the number of participants for each age group.

Table 6.1: Number of participants from each age group

Age Group	Number of Participants
10-18	9
19-30	20
31-45	15
46-65	15
65+	10

The participants from all age groups were shown a full HD video clip of 30 second duration. The PLP was applied starting with 0.01% through 10% and the streaming was done over the network. To allow the participants to fully concentrate on viewing the video once the streaming had started, their sitting and viewing position were arranged to be comfortable. This carefully set-up interactive environment facilitated a relaxed sitting position and a good viewing angle of the client computer screen to see the video being sent from the server across the network. This purpose to have a comfortable arrangement was to minimise any external distractions, helping the participants to focus fully on viewing the streamed video being played to them to assess QoE and record their MOS.

Having collected the opinion of different people in the same age group, an average was then calculated to form the MOS for that certain age group. A graph for MOS (y-axis) was plotted against PLP (x-axis) for each age group involved in the research.

The results are detailed and explained in the proceeding sections along with the MOS tables and resulting graphs.

6.2.1 QoE Evaluation by the Participants from 10-18 Age Group

Table 6.2 shows the MOS captured from teenagers. Note that the MOS in this table has been given as an average, which was calculated from all individual MOS scores recorded by these participants.

Table 6.2: MOS recorded by participants from 10-18 age group

Applied PLP (%)	Recorded MOS (5: Excellent – 1: Poor)	Participant's Age Group
0.01	4.3	10 -18 years' old
0.1	3	
1	2	
5	1.3	
10	1	

The MOS given in Table 6.2 was then changed into a graph, given in Figure 6.1.

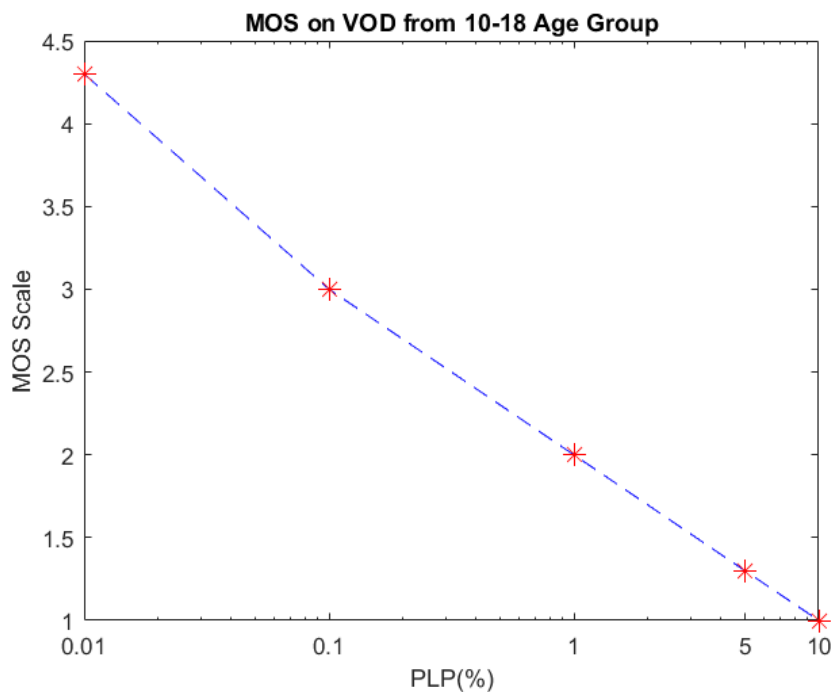


Figure 6.1: MOS graph plotted against PLP, MOS recorded by participants from 10-18 age group

It is obvious that QoE is of vital importance to teenagers who spend a significant amount of their time on digital devices these days. According to a report from CNN [131], teenagers currently spend about 9 hours on their smartphone whilst using media about 100 times a day. Another report from The Telegraph [132] also emphasises the large amount of time teenagers spend these days online through their smartphones. This amount of time spent when connected to the Internet through handsets has, of course, changed significantly in the last decade as technology has now taken many digital forms.

With smartphones being so affordable, and advanced in terms of their designs and specifications, young people have trendy and high value phones mostly with a full HD (1920x1080 pixels) display, and even 4k (4 times the number of pixels of a standard HD) in some cases. Similarly, the affordable Internet facilities

and widely available Wi-Fi hotspots have put them in a habit of experiencing high quality and very responsive streaming of online videos.

Therefore, with this mindset, a teenager can lose interest in a video if it does not play smoothly or with their expected playback quality. This behaviour can be witnessed in the MOS recorded by the teenager participants in this research. It is noted that these participants had noticed flaws in the video quality right from the beginning even with a very low PLP value. Consequently, not all of them rated the video to be 'excellent' and some of them did not even consider the video quality to be 'good' on lower PLP levels. It was seen that these young participants rated the video to be 'slightly annoying' (MOS = 3) earlier than participants from any other age group involved in this study. One of these participants, if seen individual MOS scores, rated the video to be 'very annoying' (MOS = 1) for last three streaming playbacks out of five and dropping the MOS to a 1 far earlier than expected.

Due to these teenage participants being very impatient with the degraded video quality, the average MOS from their responses, for all five levels of applied PLP, was almost one level lower than expected. Although this degree of variation shows the level of annoyance they had over the video not playing smoothly, one thing that is worth noting from these young participants' MOS is that they were never 100% satisfied with the quality of the video in the presence of packet loss even at the lowest level of PLP.

Another significant fact noted by seeing this group of participants' responses was that the average MOS, calculated from their individual MOS scores, started with a 4.3 for the lowest level of PLP being 0.01%, as compared to all other age groups who recorded a MOS of 5 for this level of PLP. This underpins a very vital finding that QoE has a significant effect on young people's lives. Also, that the QoE evaluation needs to account for the variability in the target user group(s).

6.2.1.1 *MOS Curve Fitting*

In this thesis, curve fitting has been used to find a formula that best represents the experimental data. The curve fitting has been done in MatLab, and MatLab uses the following measures of "goodness" of the fitted curve [133].

The sum of squares due to error (SSE): This statistic measures the total deviation of the response values from the fit to the response values. A value closer to 0 indicates that the model has a smaller random error component, and that the fit will be more useful for prediction.

R-Square: This statistic measures how successful the fit is in explaining the variation of the data. R-square can take on any value between 0 and 1, with a value closer to 1 indicating that a greater proportion of variance is accounted for by the model. For example, an R-square value of 0.8234 means that the fit explains 82.34% of the total variation in the data about the average.

Adjusted R-Square: Degrees of Freedom Adjusted R-Square, or simply the Adjusted R-Square uses the R-square statistic defined above, and adjusts it based on the residual degrees of freedom. The adjusted R-square statistic can take on any value less than or equal to 1, with a value closer to 1 indicating a better fit.

Root Mean Squared Error (RMSE): This statistic is also known as the fit standard error and the standard error of the regression. It is an estimate of the standard deviation of the random component in the data. Just as with SSE, an RMSE value closer to 0 indicates a fit that is more accurate.

Curve fitting statistics for the MOS from the 10-18 age group participants have been shown below along with a curve fitting plot.

General model Exp2:

$$f(x) = a \cdot \exp(b \cdot x) + c \cdot \exp(d \cdot x)$$

Coefficients (with 95% confidence bounds):

$$a = 2.402 \quad (-0.4813, 5.285)$$

$$b = -9.918 \quad (-37.14, 17.3)$$

$$c = 2.127 \quad (0.065, 4.188)$$

$$d = -0.08282 \quad (-0.2947, 0.129)$$

Goodness of fit:

SSE: 0.01797

R-square: 0.9975

Adjusted R-square: 0.9901

RMSE: 0.1341

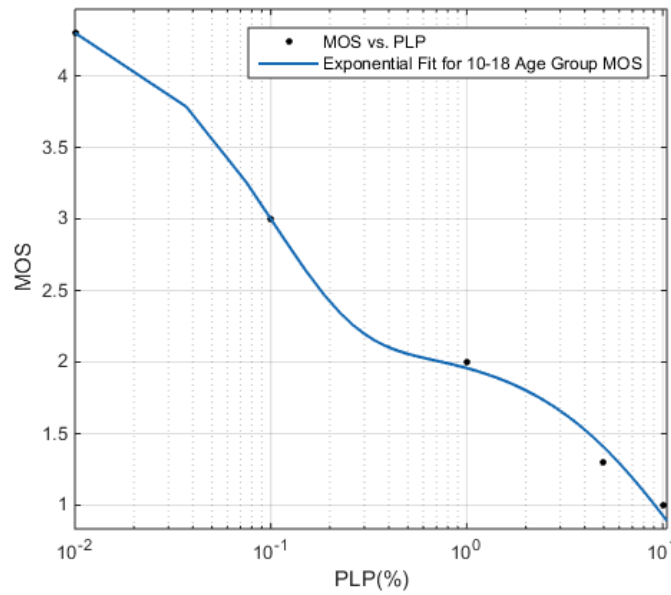


Figure 6.2: MOS vs PLP curve fitting for 10-18 age group, with Exponential function and two number of terms

6.2.2 Participants from 19-30 Age Group

Table 6.3 shows the recorded MOS by participants in the 19-30 age group.

Table 6.3: MOS recorded by participants in 19-30 years' age group

Applied PLP (%)	Recorded MOS (5: Excellent – 1: Poor)	Participant's Age Group
0.01	5	19 – 30 years' old
0.1	4	
1	3	
5	1.8	
10	1	

The recorded MOS was averaged from the individual MOS of research participants. The MOS graph based on the QoE perception of the 19-30 age group is given in Figure 6.3.

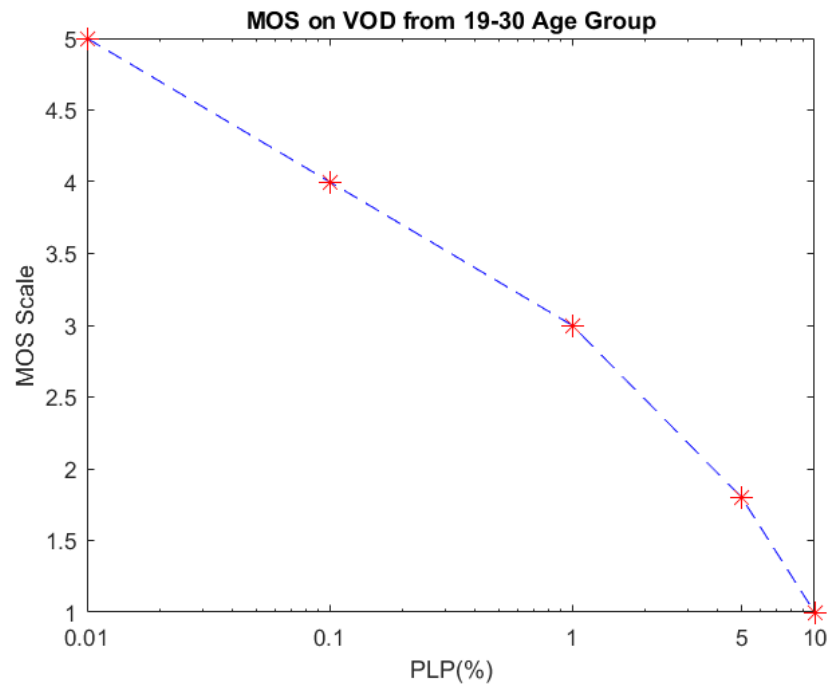


Figure 6.3: MOS graph obtained from a 19-30 years' age group participants

Figure 6.3 shows that when the PLP was as low as 0.01%, users did not get distracted by the small level of imperfection present in the video, and therefore gave a MOS of 5. With a rise in the PLP the quality of the video worsened gradually, and the degradation in QoE that caught the users' attention which was witnessed in the participants' opinion is indicated in Figure 6.3. As the PLP reached 5%, the participants showed frustration towards the perceived video quality and their average score worked out to be 1.8. Furthermore, with the maximum implemented PLP being 10%, almost all participants rated the video to be of poor quality, and hence the average MOS came out to be 1 as would be expected for a high level of PLP such as 10%.

The age group 19-30 seems to have been the most popular among QoE researchers for evaluating MOS. This could be because in an academic research environment such as a university, on average most of students who are invited to participate in MOS evaluation fall in this age group [134] [135].

6.2.2.1 MOS Curve Fitting

General model Exp2:

$$f(x) = a \cdot \exp(b \cdot x) + c \cdot \exp(d \cdot x)$$

Coefficients (with 95% confidence bounds):

a = 1.788 (1.103, 2.473)

b = -10.05 (-18.71, -1.381)

c = 3.387 (2.875, 3.899)

d = -0.1239 (-0.1639, -0.08386)

Goodness of fit:

SSE: 0.00094

R-square: 0.9999

Adjusted R-square: 0.9996

RMSE: 0.03066

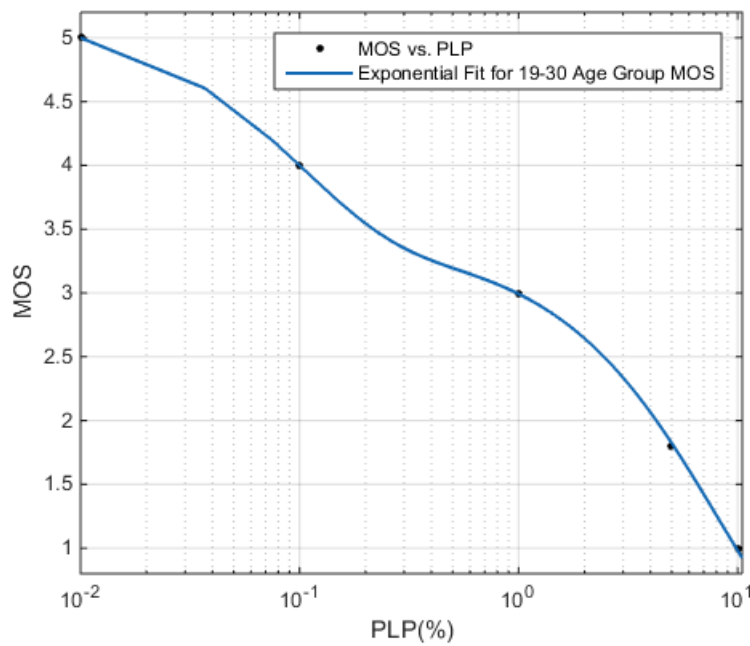


Figure 6.4: MOS vs PLP curve fitting for 19-30 age group, with the Exponential function and two number of terms

6.2.3 Participants from 31-45 Age Group

Table 6.4 shows the average MOS collected from participants from the 31-45 age group.

Table 6.4: MOS collected from participants from 31-45 age group

Applied PLP (%)	Recorded MOS (5: Excellent – 1: Poor)	Participant's Age Group
0.01	5	31-45 years' old
0.1	3.86	
1	2.85	
5	1.85	
10	1	

The MOS graph in Figure 6.5 visualises the MOS from this age group.

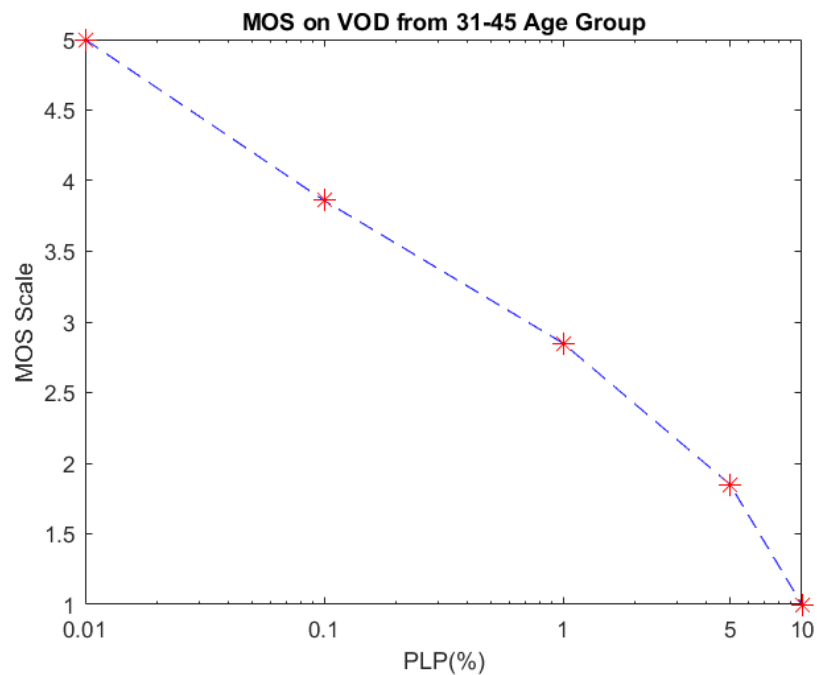


Figure 6.5: MOS graph against PLP, MOS collected from 31-45 age group participants

After the age group 19-30, the second largest age group of students at university according to the statistics from the past few years are the people between the age of 31-45 years [136] [137] [138].

Having done MOS evaluation on participants from this age group, it was noted that although the participants had different opinions on the quality of the video at different levels of added PLP, Figure 6.5 shows the average MOS denoting a decrease in MOS with an increase in PLP. However, the participants showed concern regarding the affected video quality when the added loss went over 0.01% which can be seen in Table 6.4.

This MOS graph in Figure 6.5 shows a slightly different trend as compared to the MOS recorded from the participants of the 19-30 age group shown in Figure 6.3, in which the average MOS at PLP levels of 0.1% and 1% were 4 and 3, respectively. However, in the graph in Figure 6.5 the MOS scores are 3.86 and 2.85, respectively.

6.2.3.1 MOS Curve Fitting

General model Exp2:

$$f(x) = a \cdot \exp(b \cdot x) + c \cdot \exp(d \cdot x)$$

Coefficients (with 95% confidence bounds):

$$a = 2.001 \quad (0.9862, 3.017)$$

$$b = -10.81 \quad (-23.36, 1.741)$$

$$c = 3.207 \quad (2.457, 3.958)$$

$$d = -0.1138 \quad (-0.173, -0.05468)$$

Goodness of fit:

SSE: 0.002107

R-square: 0.9998

Adjusted R-square: 0.9992

RMSE: 0.0459

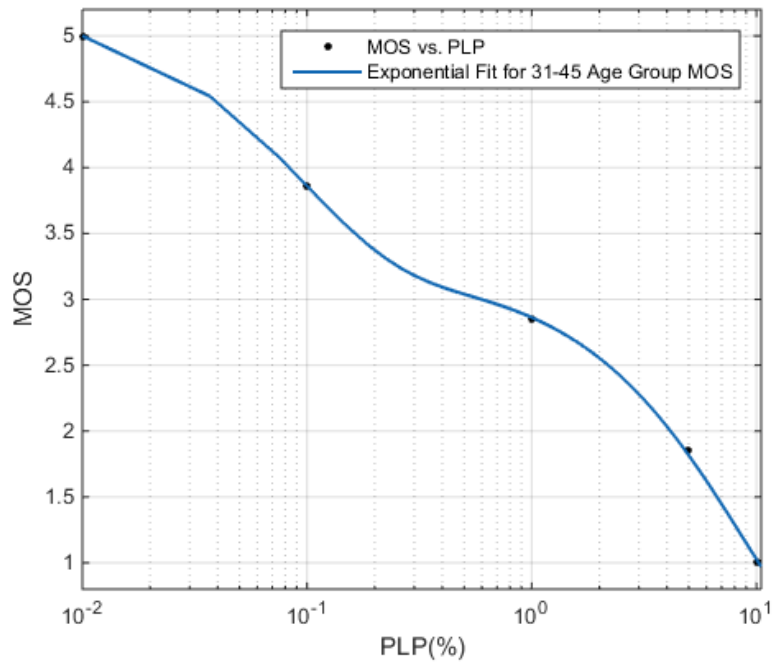


Figure 6.6: MOS vs PLP curve fitting with 2 terms for 31-45 age group.

6.2.4 Participants from 46-65 age group

Table 6.5 shows the average MOS collected from participants in the 46-65 age group.

Table 6.5: MOS collected from participants in 46-65 age group

Applied PLP (%)	Recorded MOS (5: Excellent – 1: Poor)	Participant’s Age Group
0.01	5	46-65 years’ old
0.1	3.8	
1	2.8	
5	1.8	
10	1	

Table 6.5 shows averaged MOS scores. However, by viewing the individual MOS recorded on the hard copy of the research survey sheet¹², it was noted that in response to the quality of the video played to them, some participants from this age group of 46-65 skipped the MOS rating of 4 for higher levels of added PLP.

The graph produced from the average MOS results (Table 6.5) is given in Figure 6.7.

¹² The survey sheet used to collect MOS is given in Appendix H on page 179.

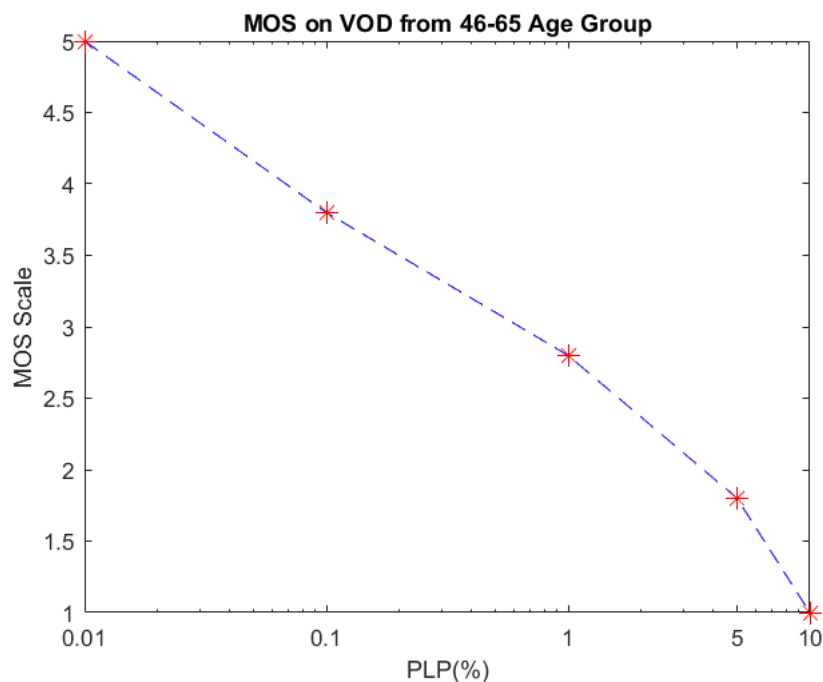


Figure 6.7: MOS and PLP graph, MOS collected from 46-65 age group

It was noted that some participants went from a MOS 3 straight to a 1 whilst skipping the 2 MOS value altogether for a PLP of 5%. Although this varying behaviour was noted in the individual recorded MOS of participants, the average MOS shown in Table 6.5 and as seen in the above graph in Figure 6.7, still gives a satisfactory trend of MOS plot.

In expressing their views on their recorded MOS, one participant stated that in his view, if a video has interesting content then even a degraded video quality does not affect the viewing experience significantly. Another participant, when asked why he chose to give MOS score of a 3 to a high packet loss such as 10%, told that although the video did not have a good quality at higher loss rates, after having viewed it a few times she was able to follow the story in the video, and hence was not massively frustrated so rated the video with a MOS of 3.

The trend seen in the plot in Figure 6.7 from this age group is very similar to that of 31-45 age group's MOS.

6.2.4.1 MOS Curve Fitting

General model Exp2:

$$f(x) = a \cdot \exp(b \cdot x) + c \cdot \exp(d \cdot x)$$

Coefficients (with 95% confidence bounds):

a = 2.077 (1.605, 2.55)

b = -10.97 (-16.7, -5.245)

c = 3.142 (2.793, 3.491)

d = -0.1132 (-0.1412, -0.08523)

Goodness of fit:

SSE: 0.0004564

R-square: 1

Adjusted R-square: 0.9998

RMSE: 0.02136

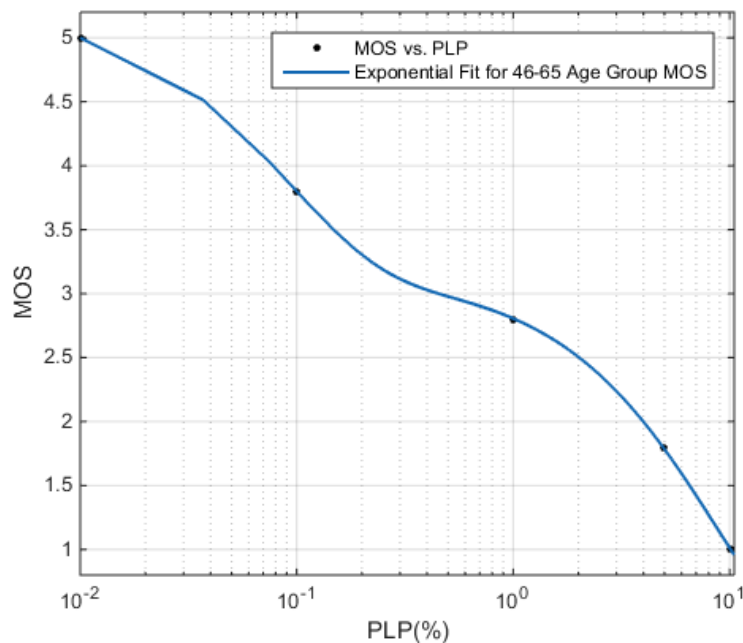


Figure 6.8: MOS vs PLP (%) curve fitting for 46-65 age group, with fitted 2 term Exponential function

6.2.5 Participants from over 65 Age Group

People in the age group of over 65 responded differently to 19-30 or 31-45 age groups. It was noted that initially with low levels of PLP, e.g. 0.01% and 0.1%, some participants did not show any frustration over the video quality at all. This was probably because they were not particularly able to notice any impairments in the video at those lower levels of packet loss. One participant even recorded a MOS of 5

at 0.1% PLP and a 4 at 1% showing considerable leniency as compared to all other age groups for the same levels of PLP. Thus, in this particular case, the participant rated the video to be of excellent quality even though some imperfections were present in the streamed video.

Furthermore, at 5% applied PLP, 3 participants from this age group recorded a MOS of 1 and rated the video quality as being ‘very annoying’. This was probably because with some video frame freezing, they lost track of the visual story and were unable to interpret the video content, hence rated the video to be of bad quality.

Regardless of the varying opinions of the participants on the quality of the video, the average MOS still gave an acceptable trend with gradual increase in the PLP. Table 6.6 shows an average of the captured MOS from this group of over 65 years of age.

Table 6.6: MOS by participants in 65+ age group

Applied PLP (%)	Recorded MOS (5: Excellent – 1: Poor)	Participant’s Age Group
0.01	5	Over 65 years’ old
0.1	4	
1	3	
5	1.5	
10	1	

The MOS graph is given in Figure 6.9. From this figure, it is noted that MOS decreased as the PLP increased.

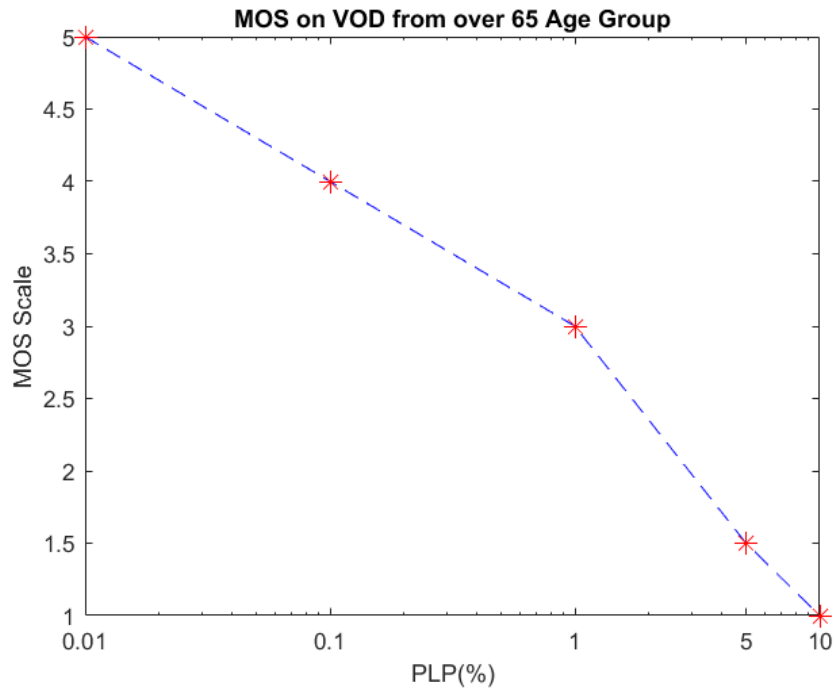


Figure 6.9: MOS graph plotted against PLP, MOS recorded by participants from 65+ age group

6.2.5.1 MOS Curve Fitting

General model Exp2:

$$f(x) = a \cdot \exp(b \cdot x) + c \cdot \exp(d \cdot x)$$

Coefficients (with 95% confidence bounds):

- a = 1.791 (-3.922, 7.504)
- b = -9.92 (-80.55, 60.71)
- c = 3.383 (-0.9518, 7.717)
- d = -0.1397 (-0.5053, 0.2259)

Goodness of fit:

- SSE: 0.06331
- R-square: 0.9943
- Adjusted R-square: 0.9774
- RMSE: 0.2516

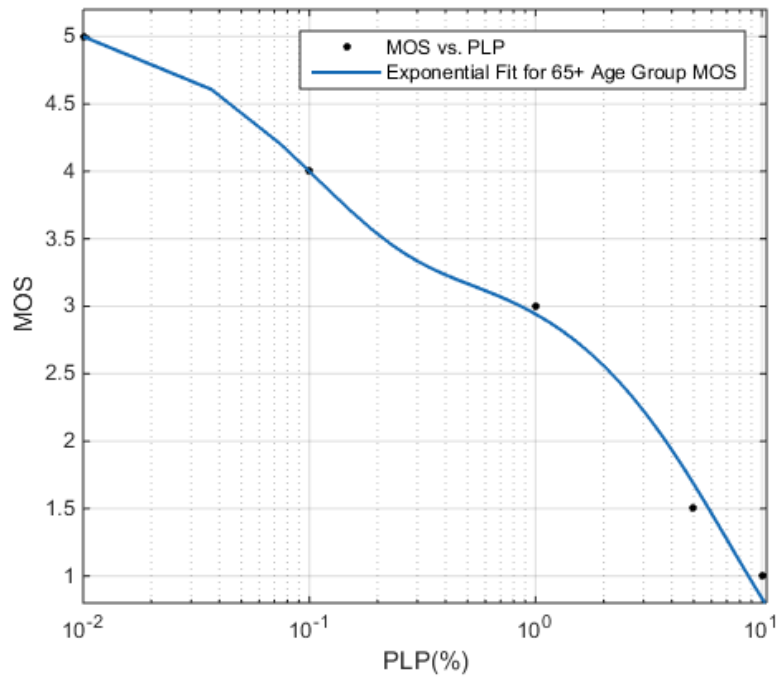


Figure 6.10: MOS vs PLP (%) curve fitting for 65+ age group, with fitted 2 term Exponential function.

A comparison graph was then produced by combining all MOS plots by different age groups, and this is given below in Figure 6.11.

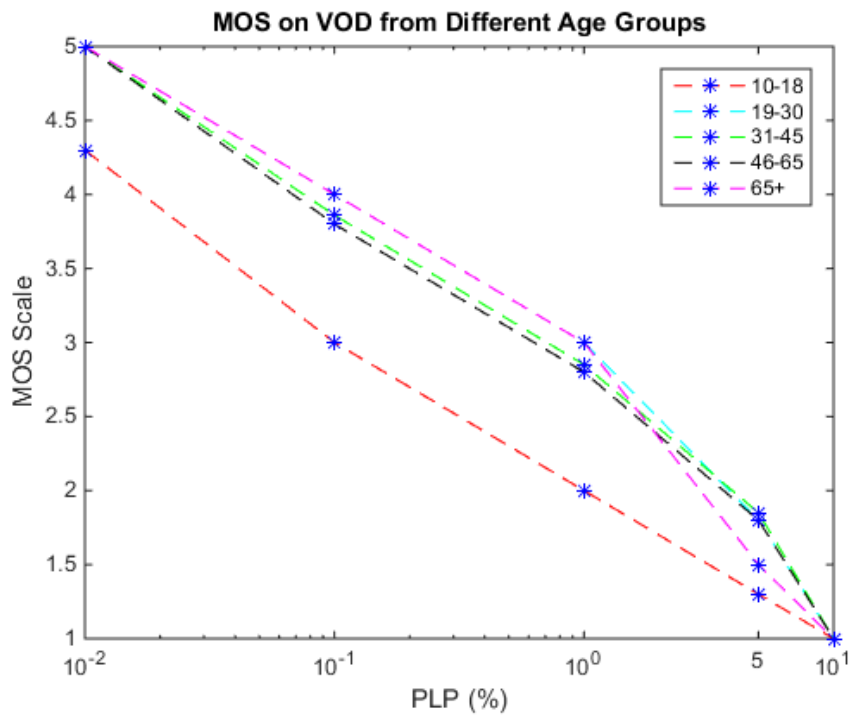


Figure 6.11: Combined MOS graph comparing MOS from different age groups

6.3 Chapter Summary

The results of this chapter in all cases show that QoE (MOS) is well modelled as an exponential, or two-term exponential function of QoS (PLP). This confirms similar results already in literature and extends the prior work to more fully examine different age groups.

By looking at the MOS graphs in this chapter, a very good fit to two term exponentials is seen when drawn against the PLP on the x-axis. This trend seen in MOS graphs is very commonly seen. In light of that, there would still be cases, even though rare, where the graph may be showing an unusual trend in MOS. Besides other potential reasons of the evaluated MOS showing such a trend, this research shows that the age group of the participant has considerable impact on the perception of QoE. For instance, in the MOS recorded in the case of an over 70 years old participant shown in Figure 6.9 on page 118, following the logic or usual QoE impressions, the score would gradually decrease with an increase in the packet loss. However, some of the elderly participants continued to rate the video with a better rating until reaching higher levels of PLP such as 1% and even 5%.

As elderly people may perceive visuals differently compared to adults or young people (see Section 2.14 on page 35 for details) so, a possible reason of this oddity in MOS behaviour could be that the elderly participants were not able to focus on the detailed aspect of the video playback in order to acknowledge the degrading video quality at lower levels of applied PLP. Or perhaps one or two of them had a certain medical condition that made the QoE interpretation somewhat challenging for them. Or, if seen on an individual basis, probably one participant merely recorded the score of a 3 accidentally when she meant to write either a 2 or a 1.

Nevertheless, it can be concluded that although we can predict what the recorded score may look like from a participant, we cannot be 100% sure of the behaviour that would be seen in an actual collected MOS score from a participant, especially when the participants are from different age groups and backgrounds. Considering the likelihood that every participant has different mental capabilities due to their age or otherwise, it is possible that every individual may perceive the QoE differently. Hence, there may be rare cases where a common predicted MOS trend may not be seen as witnessed in this research.

This research shows that young people have higher demands in terms of the digital services available to them with a desire to have maximum uptime with higher bandwidths to allow the fastest download/upload speeds. This allows them to use the network with ease any time during the day or night. However, in general, elderly people may not be as concerned about the QoE as long as the provided service is of an acceptable level. This could be because elderly people may not be downloading content several times a day, and may even not be using Internet or such services at all in the evening as compared to a young adult who may be spending a large proportion of the day on the Internet, streaming, downloading, uploading, or browsing.

Chapter 7 Evaluating YouTube MOS for Varying PLP and Round-Trip-Time (RTT)

7.1 Chapter Introduction

This part of the experimentation examined MOS study on YouTube traffic. Firstly, the effect of packet loss on the QoE of YouTube traffic was studied. The metrics of interest in this set of experiments were buffering events, the time taken to start a YouTube video, and the time taken to fully load the video.

Next, the effect of RTT and PLP as a combination was investigated to understand the behaviour of real-life networks because real packet network traffic is affected by both, packet loss and packet delay. In order to deliver the content from server to a host, TCP works very effectively to overcome quality issues as a result of packet loss and delay on a link. As such, many streaming and video download websites such as YouTube use TCP as the choice of transport protocol. Hence, to experiment with the VOD QoE being very close to real world data networks, RTT and PLP were combined in these experiments.

Experimentation was done for a range of the RTT values: 1ms, 50ms, 100ms, and 150ms. This is because many service providers such as [1] [2] [139] [5] [3] have listed the permitted latency values in their SLAs within this range across different segments of their network: core network and customer premises network. None of these companies have guaranteed a latency higher than 150ms on average. Therefore, in this part of experimentation, the delay/jitter values were chosen with an even gap in between but not exceeding 150ms to be in line with the commercial networks. Having the same value of RTT, the applied packet loss was started with 0.01% then increased to 0.1%, 1%, 5% and so on (if applicable). The applied packet loss was increased gradually until the recorded MOS dropped down to a 1, which then became the last iteration of the experiment. However, if the MOS was still not a 1 at 5% loss, the added packet loss was then incremented in steps of 5 (5%, 10%, 15% and so on) until the MOS dropped to a 1.

7.2 Experimentation Setup

The experimental setup was same as the one shown in Figure 4.11 on page 61. The server computer, referred to as NetEmBox-1 in Figure 4.11, ran the traffic control mechanism through NetEm. By default, NetEm functions in such a way that when jitter is added it re-orders packets to maintain the quality of the service. However, as the primary object of these experiments was to analyse the QoE, hence the default behaviour of NetEm was modified to be able to have a full insight into the degrading performance of the network. Therefore, a queueing discipline Packet First In First Out (PFIFO) [140] was enabled in NetEm to retain the order of packets entering and leaving an interface. The PFIFO queue discipline overwrites NetEm's default behaviour and even when the jitter is added, it keeps data packets in the original order of transmission. To add jitter, delay, and loss alone or in combination multiple network statuses were formed on the Ethernet port to emulate various network scenarios.

NetEmBox-1 was used as a proxy server between the host computer and a YouTube server. As the experimentation also involved the study of delay jitter, to ensure that no significant external delay was being added on the video stream originating from YouTube, a fast wired-Internet connection was used to connect the server computer to the Internet.

Although precautions were taken to minimise the potential external delays and losses, these precautions were only based on this end of the link and any such factors affecting the video quality across the Internet link from the university out to the YouTube server were not controllable. Therefore, to ensure such a situation did not affect the experimental results, a speed test on the link was carried out before and after running each experiment. This ensured that the link speed, even in the presence of any external delays, was fast enough to allow the experimentation whilst making any external delay and loss effects negligible. On every speed test, the Internet link that connected the testbed server to the YouTube server reported a speed higher than 100Mbps (and sometimes even as high as 700Mbps).

When experimenting with Internet traffic such as YouTube, there was certainly a part of the network that was outside of the lab network infrastructure, as such, packet delays could not have been controlled over that network. However, to ensure the collected results were not being affected by any delays from the external network from the YouTube server to the university gateway router, a speed test was conducted before after every experimental run for a given RTT value. This test showed the network speed or bandwidth in Mbps, which was always greater than 80Mbps. The download of a full HD video requires a minimum of 1Mbps of network speed at advised by Google (owner of YouTube) shown in Figure 7.1.

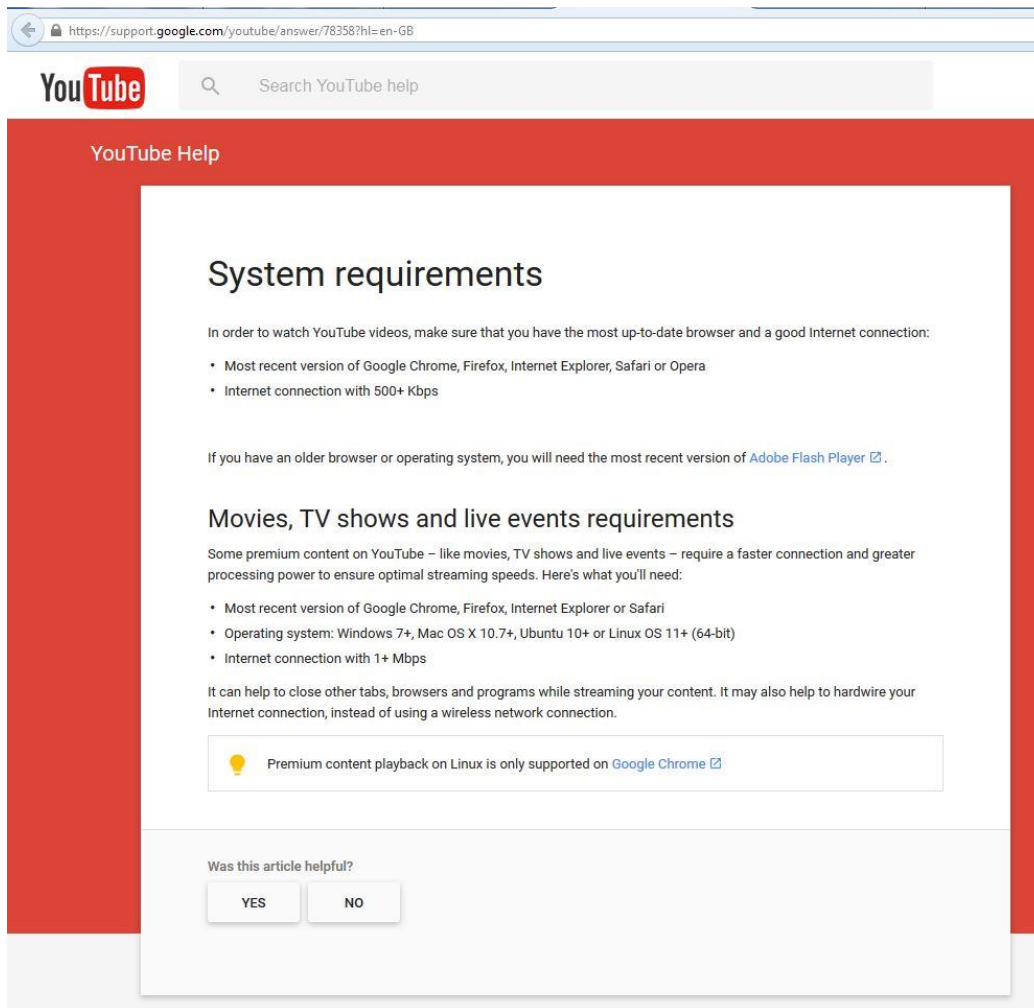


Figure 7.1: Minimum Internet connection speed required to watch YouTube videos [141].

Therefore, this suggested that if the external network had a speed of minimum 1Mbps, or 10Mbps to be on safe side, the experimentation would not suffer from any unplanned delays added by the external network. Hence, running the Internet speed test before and after every set of experiments and seeing the connection speed in the range of 100Mbps-800Mbps, it was clear that the external network delays had no real effect on the YouTube traffic that was under examination. Consequently, the delays that were added from NetEm reflected the expected behaviour.

7.2.1 Use of Bash Scripts

This part of the experimentation also required both the packet loss and RTT to be applied to the video traffic simultaneously through NetEm. However, NetEm takes one line of command at one time, and to achieve the multi-step functionality would require many lines of commands to be executed several times during one iteration of the experiment. This could have been time consuming and prone to errors if relying on human typing for the whole snippets of the commands.

Therefore, to simplify this traffic controlling task from NetEm, I prepared a bash script that would, upon execution, do the following tasks:

- 1) Accept values for loss and RTT respectively;

- 2) Inform the user of the acceptable input options if an input is entered in an unexpected format;
- 3) Show detailed user instructions if the 'help' command¹³ issued;
- 4) Show the status of the Ethernet interface NetEm is running on every time NetEm settings are changed, or 'status' command is issued;
- 5) Clear all NetEm settings if 'clear' command is issued.

```

netembox2@netembox2: ~
qdisc netem 2: parent 1:1 limit 1000 delay 1.0ms
qdisc pfifo 8028: parent 2:1 limit 1000p

netembox2@netembox2:~$ sudo livestreaming clear
-----
=> Clearing Netem at eth3...
=> eth3 cleared.
*** CURRENT STATUS of eth3 ***
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1
1

netembox2@netembox2:~$ sudo livestreaming loss 5 rtt 1
-----
=> Adding 5% loss
=> Adding 1ms RTT
=> Using pfifo buffer with Loss and RTT

*** CURRENT STATUS of eth3 ***
qdisc netem 1: root refcnt 2 limit 1000 loss 5%
qdisc netem 2: parent 1:1 limit 1000 delay 1.0ms
qdisc pfifo 8029: parent 2:1 limit 1000p

```

Figure 7.2: The bash script in action, a user can input parameters and the NetEm commands are applied in the background.

Not only does this script simplify the operation of NetEm to add packet loss and RTT, but also instructs the user on avoiding common input errors.

7.2.2 Experimentation Strategy

These days online video content is often delivered in HD, hence experimenting on HD video was a necessity to study the behaviour of modern Internet video traffic. HD videos are categorised under many resolutions starting from 720p and going up to 4K. As the lowest video resolution for a HD video is 720p, this set of experimentation was done on a 720p HD video. This is because an average Internet user may not have access to a high-speed Internet link. As such, choosing any higher resolution than 720p may not give a realistic view of a large number of users that may be viewing the content from all over the world.

This set of experiments was done in three parts. Firstly, a QoE evaluation was carried out with packet loss as a standalone metric on the YouTube traffic and MOS was recorded. Secondly, PLP and RTT were added to the link as a combination. Also, the playback quality of the YouTube video was set to be automatic (YouTube default behaviour) whilst limiting the automatic video download quality to HD with a resolution of 720p. This way, depending on the link state, the video quality could be any resolution up

¹³ The program is able to take a range of input commands at the time of execution.

to 720p but not higher. The MOS was recorded again with the application of PLP and RTT as a combination.

The third part of the experimentation was to study the QoE with PLP and RTT, but this time having the YouTube video quality 'manually' set to HD 720p. This forced the video to always play with 720p resolution. It was expected that this would show a low MOS score for the same amount of added PLP and RTT that earlier showed a better MOS in the case of leaving the YouTube video player's video quality on 'Auto'. This is because when the video quality is on auto, YouTube prioritises the latency attempting to play the video smoothly and dropping the video quality to a lower resolution if necessary, depending on the connection bandwidth. So essentially, if the Internet connection is slow YouTube will reduce the video quality but will try to keep the buffering to a minimum. However, when the user chooses a particular video resolution for the playback such as 720p, YouTube does not then drop the video quality even if the connection is slow. Consequently, the user will always have the video playing with their desired resolution which may not be an issue on a fast Internet link. However, having the HD resolution selected on a poor Internet link, the video buffering increases as YouTube tries to deliver the video content in HD e.g. 720p resolution but the limited capacity or a bad network link makes it hard.

During the experimentation, another practice that proved to be useful was restarting the browser before every run of the experiment, i.e. every download of the video after having modified the interface state of NetEm. This is because it was noted that if the video was just refreshed in the same browsing session it would sometimes deviate from the expected playback behaviour. During the research on the subject, some discussion threads on the Google products forum showed that other people had discussed similar issues [142] [143] [144]. Some of these users suggested that occasional unplanned background tasks running on the browser can affect the YouTube video delivery, therefore restarting the browser can improve things significantly. This was witnessed when an improved and expected video download behaviour was seen following the browser-restart approach during experiments.

Although restarting the browser more often overcame the common video playback issues, a new browser session would reset the customisation of video playback controls on the YouTube player itself. For example, previously used volume level or selected video quality for the playback would have to be re-adjusted for every new session after the browser restarted. Some sort of mechanism was needed to be in place to maintain video quality at the same resolution regardless of the browser closing or new sessions being initiated. This desired functionality was achieved with the use of a browser extension. This installed extension would set the YouTube playback quality to a predefined resolution, this has been discussed further in Section 7.7 on page 140.

7.2.3 Analytical MOS Calculation

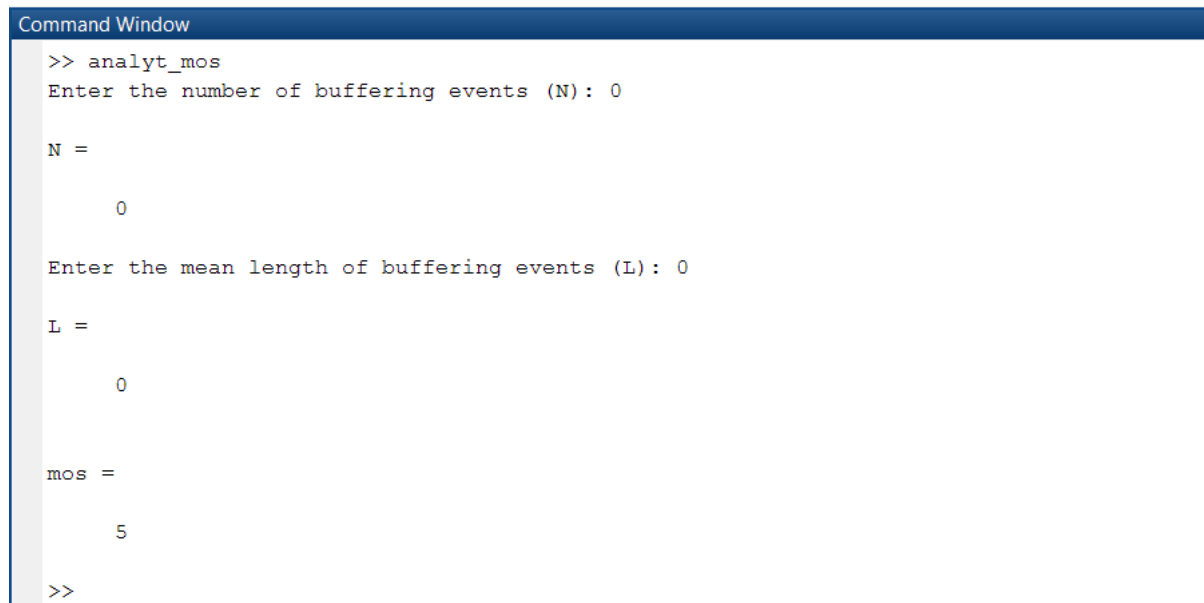
T. Hobfeld et al [145] derived an equation that can be used to predict YouTube QoE. In this thesis, this equation is referred to as the theoretically predicted MOS and is represented as equation (7.1).

$$f(L, N) = 3.50e^{-(0.15L+0.19).N} + 1.50 \text{ for } L \in \mathbb{R}^2, N \in \mathbb{N} \quad (7.1)$$

In equation (7.1), L refers to the length of stalling or buffering of the video, and N denotes the number of buffering/stalling events.

With the help of this equation, the analytical MOS was calculated for the parameters used in the experiment. This predicted MOS was then plotted against the experimental MOS. One limitation of this formula however, is that it does not give a MOS lower than 1.5. So even if the recorded counterpart of the MOS is a 1, the above MOS formula will not go lower than 1.5.

To save time and to minimise human errors, a Matlab program was designed incorporating equation (7.1). This program takes the number of buffering events N along with length L for each buffering event and outputs the subsequent MOS for the YouTube QoE.



```

Command Window
>> analyt_mos
Enter the number of buffering events (N): 0

N =

    0

Enter the mean length of buffering events (L): 0

L =

    0

mos =

    5

>>

```

Figure 7.3: Screenshot of MatLab program in action for calculating YouTube QoE.

This designed program asks the user to provide the number of buffering events and the mean length of buffering events. Based on the provided values it calculates the analytical MOS.

7.3 How the YouTube Video Download Works

YouTube mainly offers VOD in the form of progressive download of the video in the client's browser. It has over two billion video streams daily, making it the most popular VOD service on the Internet. YouTube started in 2005 like any other online video content sharing website, and gained popularity very rapidly. YouTube was later acquired by Google, which meant the way content was to be delivered changed significantly with the backend powered by Google's robust infrastructure.

7.3.1 From Requesting a YouTube Video to Viewing it

When a user opens their web browser and opens a certain video, this request is not served by only one YouTube server. The layout (i.e. how the content is displayed, styles present on the page or opened video) of the YouTube website is served by a front-end server. However, the video content itself, on loading the page, is actually delivered by the Content Delivery Network (CDN) servers (also called cache servers) of YouTube.

The YouTube content delivery infrastructure has been distributed in various locations all over the world. This, with the help of the DNS lookup mechanism, allows the user content to come from a CDN server

closest to them depending on their geographical location, which in return optimises the content delivery cutting down on delays in transmission. If a server in a certain location is overloaded, or does not have the requested video, the request is automatically redirected to another suitable CDN server without the user noticing it [146] [145].

If one were to outline the steps taken to deliver a YouTube video on request, it could be done in the following steps [145] [147]:

1. A video request is made by the user on the YouTube webpage, which goes to the YouTube front-end server that serves the HTML content of the YouTube webpage
2. After the HTML has been loaded in this page, it is determined how the video will be displayed to the user. If the YouTube page is using an HTML5 video player it is then ready to take the content and play it back to the user. However, if the YouTube website is using Flash technology, a dedicated flash player has to be present to activate the video flash container
3. If the video container to play the video is ready from the previous step, an automatic request for the actual video content is made to a cache server. In the case of this cache server being overloaded a redirect HTTP 302 message is sent to process the client redirect to another cache server
4. This redirect request for the required video is made to the other cache server, and the desired video is sent over to the client from the chosen cache server. This video then starts playing on client browser in the form of a progressive download.

The delivery of the video content from one cache server or another depends on various factors. Some of these could be load balancing across the data centres, DNS server variations in a network, and the delivery of spare video content that may not be necessary to replicate across all data centres [147].

7.4 YouTube Content Delivery Infrastructure

Google officially does not disclose much about the YouTube datacentres. However, seeing that Google has its datacentres around the globe it is believed that the YouTube CDN servers are located in these datacentres. The map below outlines the major Google datacentres in various countries:



Figure 7.4: Google datacentres worldwide [148].

Although in the above map Google has shown its significant datacentres, however, it is believed [149] that very detailed information about such infrastructure could be very valuable to its competitors, hence Google maintains its confidentiality in that respect. Regardless, it is not unlikely that Google is always investing in its infrastructure expansion to competitively stay ahead in the industry.

The researchers in [150] built a distributed measurement infrastructure in order to reverse-engineer YouTube content delivery cloud. They then collected and analysed a large amount of data. This data analysis reflected the underlying mechanisms of YouTube. It was noted that the YouTube video content delivery cloud consists of three major components [150]:

Video ID space: Each YouTube video has a unique 11 literals long identifier. This identifier can be formed of a combination of [A-Z], [0-9], - or _. For example, in this YouTube video link, the outlined part of the URL is the video ID for this particular YouTube video: <https://www.youtube.com/watch?v=e7gR7EYjcP8>

The above video ID is of course for just one video, however, the YouTube video space has a total of 64^{11} IDs to uniquely identify all videos present on YouTube.

3-tier physical server cache hierarchy: Although Google has its own cache servers at various locations that host the YouTube content, additionally there are more physical video cache server locations spread across five continents that are part of this video delivery cloud and some hosted inside other ISPs including Comcast and Bell-Canada. Having comprehended this layout, it is noted that YouTube has a 3-tier physical cache hierarchy. This three-tier hierarchy comprises primary (P), secondary (S), and tertiary (T) video cache locations.

Multiple DNS namespaces: These are made of well-organized logical video servers. Videos from YouTube are linked to a physical cache hierarchy with a set of anycast and unicast namespaces. These namespaces are logical, anycast namespaces allowing the mapping of more than one IP address whereas a unicast namespace can only map one unique IP address.

In further analysis of the YouTube cache locations and cache server sizes, the authors of [146] report YouTube to have video cache servers in 45 locations. A geographical view of these locations can be seen in Figure 7.5 below.



Figure 7.5: Geographical view of YouTube cache sever locations showing Primary (P), Secondary (S), and Tertiary (T) locations.

It is noted that there are more than one tier type cache servers in one location. The primary, secondary, and tertiary locations have been denoted with letters P, S and T respectively in the diagram. 43 out of these 45 locations signify primary, 8 are secondary and 5 are tertiary cache locations.

7.5 YouTube QoE Experimentation with Packet Loss

As the experimental setup in Figure 4.11 on page 61 depicts, an http request for a YouTube video was made from the client computer to the YouTube server across the Internet. The requested video was sent back by the YouTube server in a web browser and, on its way to the client computer, it went through the testbed server, NetEmBox-1. This way, NetEmBox-1 was able to implement/modify network parameters such as loss and delay on the video stream. In this experiment, packet loss was applied from the NetEmBox-1 server on the YouTube video traffic and the QoE was assessed by evaluating the MOS on the client computer. Table 7.1 shows the results from this experiment.

One indication of whether a link is affected with excessive delay or packet loss is the presence of buffering in the video streaming. Buffering takes place when the rate of arriving packets of video traffic is less than the rate at which packets are being streamed to the user. Buffering affects the video QoE directly as the user gets annoyed when the video stalls during playback. Therefore, in VOD QoE, video buffering was also given focus. As such, buffering details of YouTube codec were also logged in this experiment to establish a relationship between buffering time, number of buffering events and the MOS score. With this relationship, a best fit equation was created by identifying the key points between the time to start, time to fully load and MOS fields. Refer to Section 7.2.3 for details. Time to start a video also played a vital role in the QoE assessment. The time to fully load a video was somewhat irrelevant if no buffering events took place and it would still lead to a MOS score of 5, rating the video quality to be excellent.

Table 7.1: YouTube QoE experimentation with loss in the presence of buffering

Packet Loss (%)	Time to start	Time to fully load	No. of buffering	Recorded MOS
0	2.78	5.56	0	5
0.01	2.95	5.81	0	5
0.1	3.22	6.79	0	5
0.25	3.09	6.23	0	5
0.5	2.87	6.45	0	5
1	4.72	12.18	0	5
10	18.47	92.09	5	2

Figure 7.6 shows that in the absence of buffering events, the graph showed a straight horizontal line across MOS value of 5 on the y-axis for packet loss of 0% through to 1% (see also Table 7.1). When the PLP reached 10%, buffering was seen in the video hence the MOS dropped to a 2.

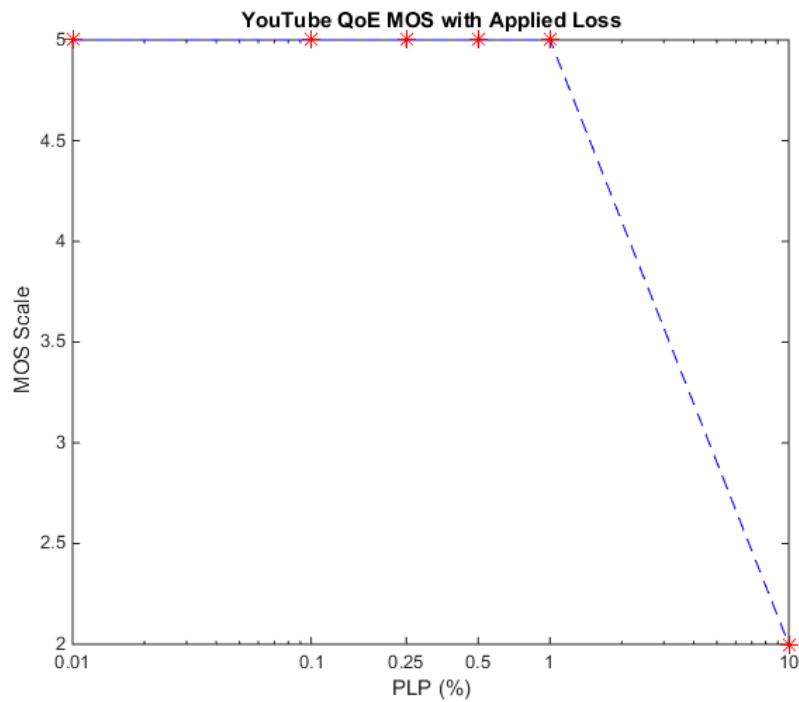


Figure 7.6: YouTube QoE experimentation with NetEm implemented loss

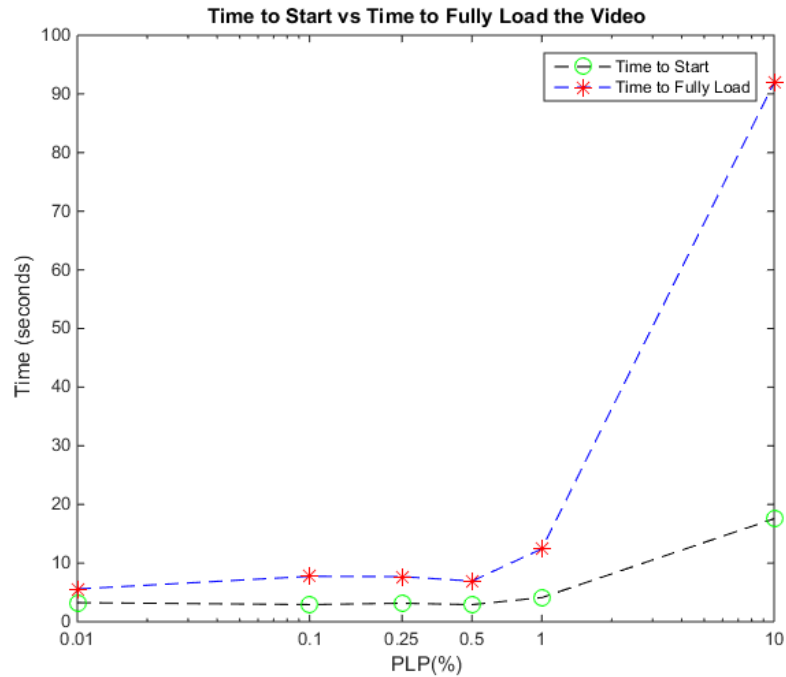


Figure 7.7: YouTube QoE experimentation, analysis of the time taken to start the video and to fully load the video.

As buffering took place more than once in the presence of a PLP of 10%, hence an average was taken for ‘time to start’ and ‘time to fully load’ fields. The experiment was run several times with a loss of 10%, and as expected, the values for buffering fields were slightly different for each run. But in general, it is concluded that the higher the PLP, the more the buffering events, and consequently, the longer the time required to start and fully load the video.

7.6 YouTube QoE Evaluation with Automatically Adjusted Download Quality

By default, when a video is requested from a YouTube server, in response, the browser determines the Internet speed and sets the video quality to match the Internet speed to play the video smoothly. If the link bandwidth is low the video quality will be lower and vice versa.

7.6.1 RTT = 1ms

During the experimentation, the packet loss was incremented starting with 0.01% and going up gradually until the recorded MOS dropped to 1, unacceptable video quality. The RTT was changed on a per experiment basis and not for every run of the same experiment. Hence to start off, RTT was set to 1ms for all runs of this experiment.

For each experiment, an Internet speed check was conducted on the network link. Figure 7.8 and Figure 7.9 show the link speed just before and after this experiment was carried out.

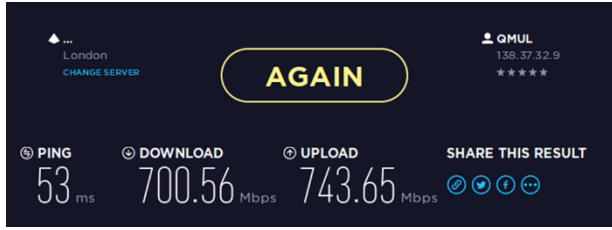


Figure 7.9: Speed test prior to the experiment

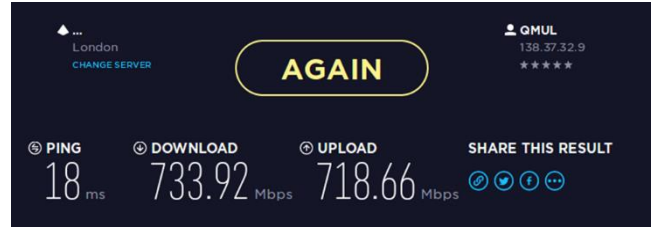


Figure 7.8: Speed test after the experiment

As seen in Figure 7.8 and Figure 7.9, the download speed is much higher than required for a smooth YouTube video playback of a high definition video. As seen at the beginning of this chapter, having the download speed higher than a few megabytes will guarantee that the Internet link to the YouTube server on its own will not cause the video to stall. Therefore, only the packet delay or loss added on the traffic from NetEm would cause the buffering events, if any.

The noted results are the number of buffering/stalling events (N) which refers to the number of times the video was stalled during the YouTube download/playback and the length of buffering (L) which denotes the total buffering time per experiment. The average buffering length column in the table shows the mean buffering duration calculated from all the buffering durations for all runs of the experiment. Then for MOS, the recorded MOS is based on the participant’s opinion of the quality of the video in one given scenario, whereas the analytical MOS is the equivalent MOS calculated for validation purposes. The analytical MOS is calculated with the formula given on page 125, which makes use of N and L.

Table 7.2 shows the results captured during the experiment.

Table 7.2: Experiment results when RTT = 1ms

RTT = 1ms				MOS	
Added Loss (%)	No. of buffering events (N)	Length(s) of Buffering (secs)	Average Length (L)	Analytical	Recorded
0.01	0	0	0	5	5
0.1	0	0	0	5	5
1	0	0	0	5	5
5	0	0	0	5	5
10	1	1.90	1.90	3.67	4
15	1	5.48	5.48	2.77	3
20	2	6.65 0.51	3.58	2.31	3
25	2	3.33 9.86	6.60	1.83	2
30	3	8.07 1.85 10.46	6.79	1.59	2
35	4	23.36 1.21 2.68 0.33	6.90	1.53	1

It was noted that when the RTT was as low as 1ms, the video was playing perfectly for first four increments of the added packet loss. When the packet loss reached 10%, as it is a significant amount of packet loss hence the video started buffering. Then with further 5% increments of packet loss, the video quality was linearly worsening with an increase in the buffering events or duration. However, surprisingly the MOS did not drop to a 1 until the packet loss reached as high as 35%. That is when this 30 seconds video buffered four times and the average duration of the buffering was 6.90 seconds, hence the participant considered the video to be unwatchable and gave a MOS score of a 1.

The graphs for both the recorded and the analytical versions of MOS are given in Figure 7.10 below.

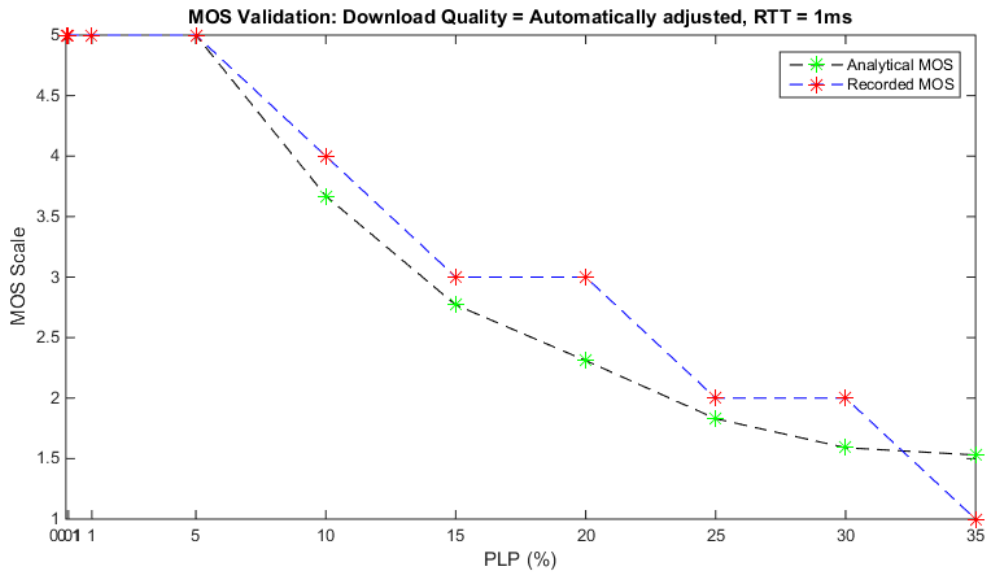


Figure 7.10: Experimental and analytical MOS for automatic YouTube download quality and RTT of 1ms

Figure 7.10 shows that although plots from both the MOS recorded by the participant and the predicted MOS show slightly different trends, there is still some similarity between them. For the first five runs of experiment the MOS from both models was 5. Then for the packet loss of 10% and onwards there was slight variation in both models of MOS. For the last experimental run, when the packet loss was 35% and the participant gave a MOS of 1, the analytical MOS at that point was 1.53 since the lowest the analytical MOS can go is 1.5. This is a known limitation in the QoE formula for calculating MOS.

7.6.2 RTT = 50ms

Proceeding further with the experimentation, the RTT was increased from 1ms to 50ms and the experiment was repeated with the same values of added packet loss until the recorded MOS reached 1.

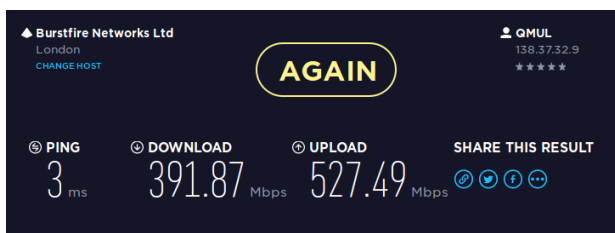


Figure 7.12: Speed test prior to the experiment

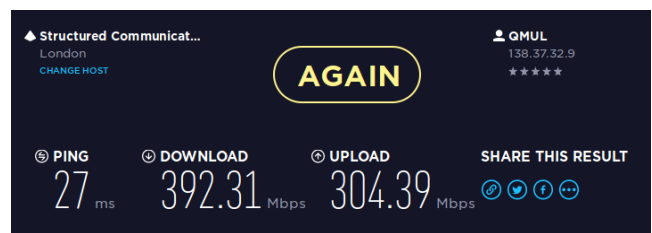


Figure 7.11: Speed test after the experiment

Again, the external Internet speed was fast enough to not being affected by any external delays on the link.

Table 7.3 shows the captured results for this experiment.

Table 7.3: Experiment results when RTT = 50ms

RTT = 50ms				MOS	
Added Loss (%)	No. of buffering events (N)	Length(s) of Buffering (Secs)	Average Length (L)	Analytical	Recorded
0.01	0	0	0	5	5
0.1	0	0	0	5	5
1	0	0	0	5	5
5	0	0	0	5	5
10	2	01.54 01.44	1.49	3.03	3
15	2	02.33 02.16	2.24	2.72	3
20	2	06.14 06.93	6.53	1.84	2
25	4	09.29 03.19 03.24 02.39	18.11	1.50	1

The table above shows that the MOS was 5 for the first four runs of the experiment, again the same as in the last experiment when the RTT was 1ms. However, this time having RTT to be 50ms, as the added packet loss reached 10% there were two buffering events as compared to the previous experiment when there was only one buffering event. Furthermore, in this experiment the added packet loss could only make it to 25% before the MOS dropped all the way to 1.

The plot below in Figure 7.13 compares both versions of the MOS for this experiment.

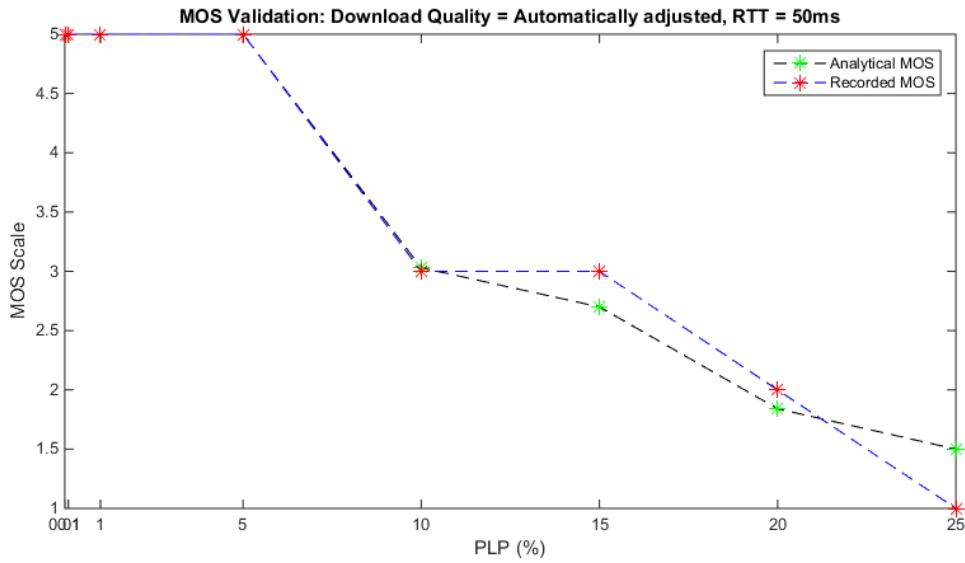


Figure 7.13: Experimental and analytical MOS for automatic YouTube download quality and RTT of 50ms

In the plot, it is noted that although there is slight discrepancy between both versions of the MOS, the plots from both the predicted MOS and the one recorded by the participant do not follow completely different paths. Again, the QoE formula limits the analytical MOS to go any lower than 1.5.

7.6.3 RTT = 100ms

The experiment was then repeated with an increase in the RTT value from 50ms to 100ms. The Internet speed was checked by running a speed test again before and after the experiment, which is shown in the screenshots in Figure 7.14 and Figure 7.15 below.



Figure 7.15: Speed test prior to the experiment

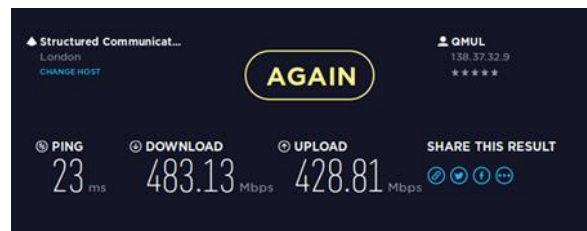


Figure 7.14: Speed test after the experiment

The experimental results for this run of the experiment are given in Table 7.4.

Table 7.4: Experiment results when RTT = 100ms

RTT = 100ms				MOS	
Added Loss (%)	No. of buffering events (N)	Length(s) of Buffering (Secs)	Average Length (L)	Analytical	Recorded
0.01	0	0	0	5	5
0.1	1	0.11	0.11	4.35	5
1	1	0.36	0.36	4.24	5
5	1	1.04	1.04	3.98	4
10	2	0.49 0.41	0.45	3.59	4
15	2	1.70 1.04	1.37	3.08	3
20	4	6.98 1.55 0.52 1.93	2.74	1.81	2
25	2	8.20 7.18	7.69	1.73	2
30	3	38.56 08.67 03.29	16.84	1.50	1

As it is evident from the captured results in Table 7.4, having a higher RTT value i.e. 100ms affected the quality far more as compared to the previous two experiments where the RTT was 1ms and 50ms. The buffering in the video was seen at as low as 0.1% packet loss. But despite this low amount of packet loss, the overall QoE was affected due to a high delay i.e. RTT = 100ms. The recorded MOS, however, did not reach 1 until the packet loss was 30%, which is entirely based on how the participant perceived the QoE of the video. There were four buffering events in this 30 second video clip at PLP = 20%. Generally speaking, the recorded MOS could have well dropped to a 1 even at 20% packet loss if for instance the video was assessed by another user who was a little more concerned by the longer waiting time as a result of buffering.

The MOS validation graph is shown in Figure 7.16.

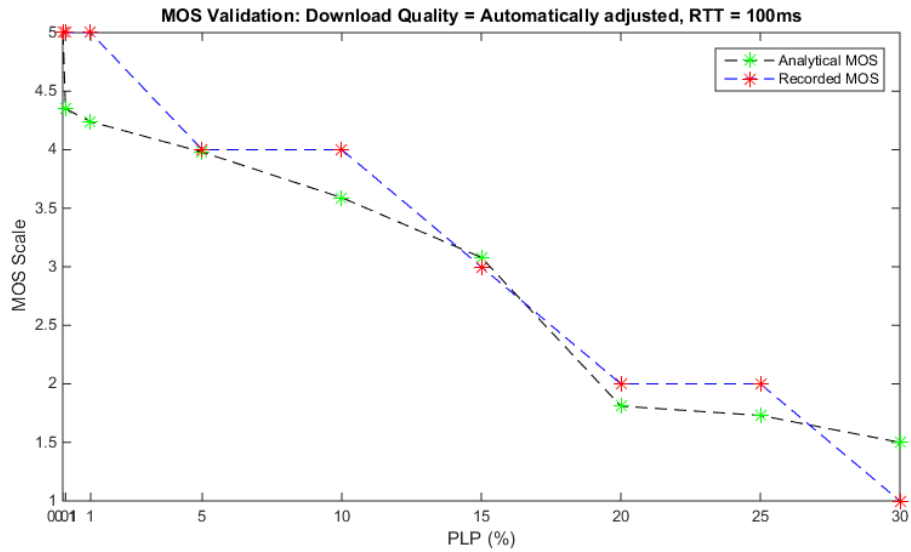


Figure 7.16: Experimental and analytical MOS for automatic YouTube download quality and RTT of 100ms

This time, there was a considerable variation in both models of the MOS. This may be because the participant was a little too lenient in this run of the experiment and gave a generous MOS rating for video playbacks. However, the overall trend that is seen in both plots does not show a completely different shape.

7.6.4 RTT = 150ms

This was the last experiment in this series of experiments where the resolution of the YouTube video would get set automatically considering the link quality and internet speed between the host computer and the YouTube server. The RTT delay was pushed up to as high as 150ms. It was expected that the QoE of the VOD would be the worst compared to the previous experiments in the same part of experimentation. The experiment was then repeated with same increments in the added packet loss until the participant MOS dropped to 1.

The speed test on the Internet link was carried out before and after the experiment, as shown in Figure 7.17 and Figure 7.18. These figures show that the download speed at the time of the experiment was far higher than required to download a high definition video from YouTube.

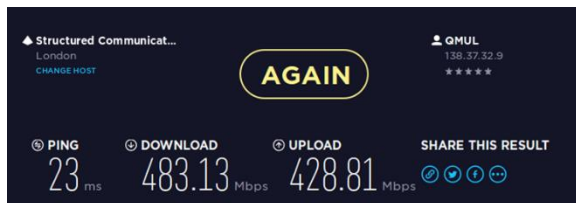


Figure 7.18: Speed test prior to the experiment

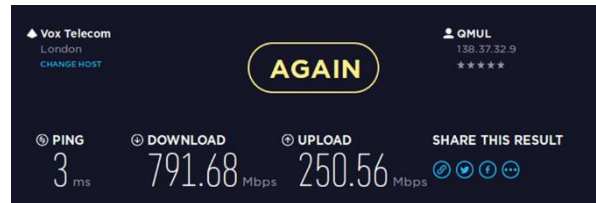


Figure 7.17: Speed test after the experiment

The experimental results are shown in Table 7.5 below. As expected, the QoE in this experiment was significantly affected.

Table 7.5: Experiment results when RTT = 150ms

RTT = 150ms				MOS	
Added Loss (%)	No. of buffering events (N)	Length(s) of Buffering (Secs)	Average Length (L)	Analytical	Recorded
0.01	1	0.80	0.80	4.07	5
0.1	1	1.12	1.12	3.95	5
1	1	2.03	2.03	3.63	4
5	1	4.38	4.38	3	3
10	4	1.09 0.21 0.46 0.48	0.56	2.67	3
15	4	3.50 1.61 2.11 4.50	2.93	1.78	2
20	5	4.02 4.98 9.79 28.34 0.33	9.49	1.50	1

The results for this experiment show that video buffering was witnessed right from the beginning at the lowest added packet loss of 0.01%. But as the RTT was very high (150ms), the QoE was seen to have been affected severely. The recorded MOS did not drop to a 1 until the packet loss was 20%, however, looking at the results it is evident that there were far more buffering events and with longer durations when compared to the previous set of experiments.

The graph to visualise the MOS validation is shown in Figure 7.19.

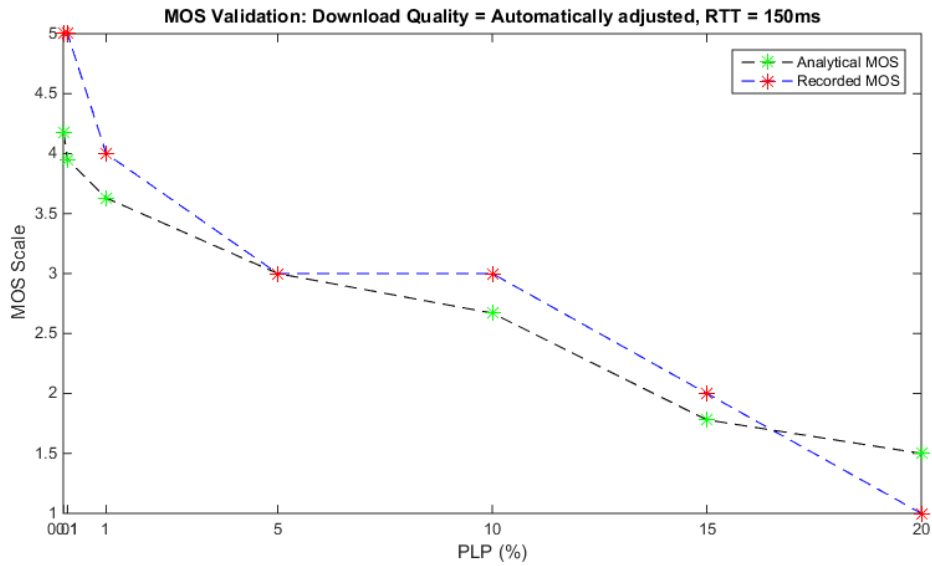


Figure 7.19: Experimental and analytical MOS for automatic YouTube download quality and RTT of 150ms

In this experiment when the RTT is 150ms, it is noted that recorded MOS did not fully agree with the analytical MOS. However, the overall shape of both individual plots did show similarities in the trend.

Furthermore, it was noted that the QoE for this last experiment with an RTT of 150ms was the worst of all previous three experiments with lower RTT values of 1ms, 50ms, and 100ms, which is what was expected.

7.7 YouTube QoE Evaluation with a Resolution of 720p as the Download Quality

In the first phase of the experimentation, when playback quality was kept on auto, restarting the browser for each experimental run after a change in loss and RTT did not require any particular attention. However, in this second phase of the experimentation, the video quality had to be set to one particular resolution regardless of the Internet connection speed and the PLP and delay present on the link. As such, the default behaviour of YouTube raised a concern as reopening the browser for every run would jump back to the default settings of YouTube where the video quality is chosen depending on the Internet speed [151].

Therefore, this default YouTube behaviour was to be modified. One of the most convenient ways to accomplish this was to install a Mozilla browser extension that forced the video to play in the desired HD format every time YouTube loaded in a new browser session [152].

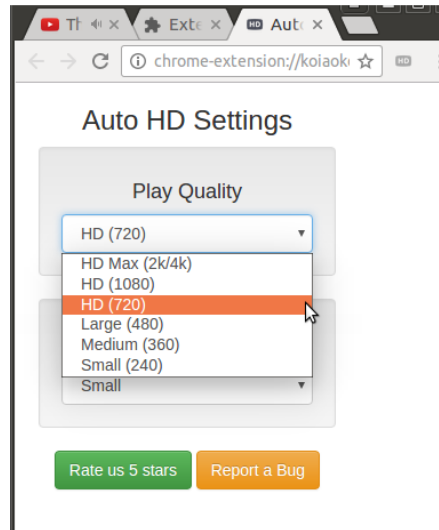


Figure 7.20: Mozilla Firefox extension for forcing YouTube video quality to a desired resolution on every reload of the page.

As shown in the screenshot in Figure 7.20, selecting HD 720p in this Mozilla Firefox extension would ensure that the quality of the YouTube video playback is set to be 720p every time YouTube is opened in a Firefox browser. This setting will be stored in the browser preferences until this is play quality option is changed or the extension is uninstalled.

As in the last set of experiments, again, there were four experiments with different RTT values. The first experiment was with an RTT value of 1ms. The remaining experiments were with RTT being 50ms, 100ms, and 150ms.

7.7.1 RTT = 1ms

The experimental setup in this series of experiments was the same as in Section 7.6 where YouTube quality was set to be 'Automatic'. The RTT value in this experiment was set to be 1ms, however, the packet loss was added in increments starting with 0.01% until the participant recorded MOS dropped to a 1.

As this set of experiments requires the video to be played in HD 720p, the speed of the Internet link was of vital importance since having the Internet speed lower than a threshold would directly cause buffering in the video. Therefore, the speed test was done again in every experiment prior to and after taking the readings.

The screenshots shown in Figure 7.21 and Figure 7.22 show the approximate link speeds before and after this experiment.

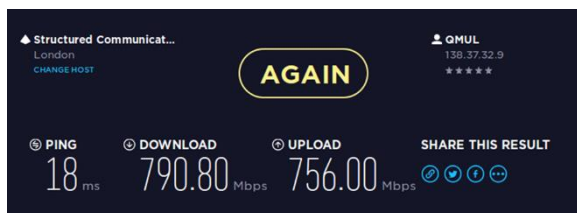


Figure 7.22: Speed test prior to the experiment

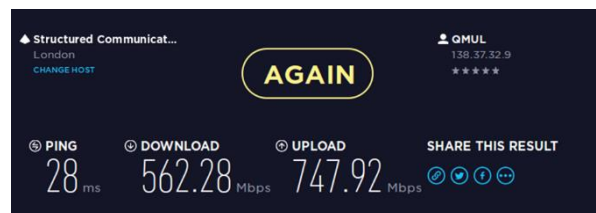


Figure 7.21: Speed test after the experiment

Seeing these very high link speeds returned by the speed test verifies that the external link condition from the university lab to the YouTube server did not cause any buffering at all. Hence it was safe to regard the behaviour of the collected results to be reflecting only the delay/loss metrics set through NetEm. Table 7.6 shows the captured results.

Table 7.6: Experiment results when downloading at 720p YouTube quality for RTT = 1ms

RTT = 1ms				MOS	
Added Loss (%)	No. of buffering events (N)	Length(s) of Buffering (Secs)	Average Length (L)	Analytical	Recorded
0.01	0	0	0	5	5
0.1	0	0	0	5	5
1	0	0	0	5	5
5	0	0	0	5	5
10	1	1.07	1.07	3.96	5
15	6	0.97 1.12 0.93 1.68 1.96 2.67	1.55	1.78	2
20	9	4.26 1.46 8.32 1.94 2.29 1.59 3.30 1.37 4.77	3.25	1.50	1

It was seen that having the RTT to be as low as 1ms, the initial low amounts of applied packet loss did not affect the QoE and a MOS 5, i.e. excellent video quality, was scored (see Table 7.6). For low packet loss values, the QoE of the video was similar to the ones seen in this experiment's counterpart (where RTT = 1ms) previously when YouTube download quality was automatic. However, as the applied packet loss increased, the QoE was affected far more excessively. This is because previously YouTube kept switching between various playback qualities depending on the Internet link bandwidth. For example, previously when the Internet link introduced degradation because of the combination of the applied RTT and packet loss, YouTube dropped to a lower download quality but still kept the video playing whilst avoiding any buffering.

However, this time YouTube was forced to stay at 720p regardless of the state of the Internet link. Therefore, if the combination of the added RTT and packet loss went over an acceptable level, YouTube was not able to drop to a lower quality to avoid buffering. Consequently, the video would start buffering

for an even lower combination of the applied network deficiencies. That is reflected in Table 7.6 where there are more buffering events for packet losses of 15% and 20% in comparison with the last case.

Figure 7.23 below shows a plot of both types of MOS values. It is seen that there was a good agreement between both models of MOS, the one recorded by the participant and the analytical MOS.

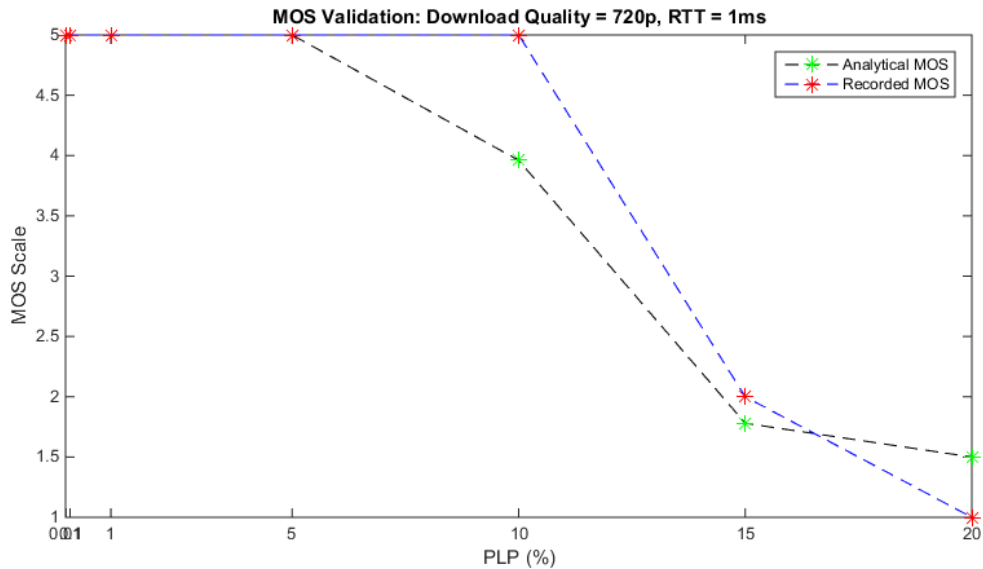


Figure 7.23: Experimental and analytical MOS for 720p YouTube download quality and RTT of 1ms

For PLP up to 5%, both plots were overlapping. However, when the PLP reached 10% the analytical or expected MOS was 4 but the participant was not too concerned so still gave a MOS score of 5 to the quality of the video. Then for 15% and 20% PLP, the MOS points were not far off from each other. As previously stated, due to the limitation in the QoE MOS formula the last point of MOS given by the formula would never be less than a 1.5.

7.7.2 RTT = 50ms

The screenshots of Internet link speed test in Figure 7.24 and Figure 7.25 show that link bandwidth was more than required to download this 30 second YouTube video. Hence, the external link condition would not affect the download of the video.

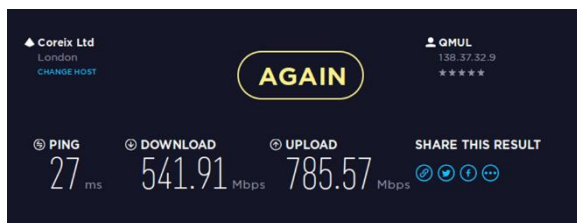


Figure 7.25: Speed test prior to the experiment

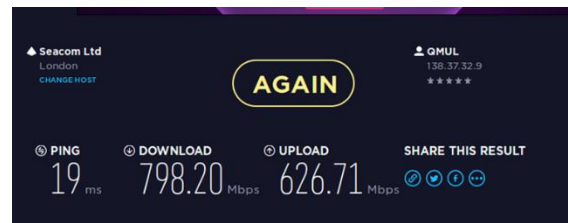


Figure 7.24: Speed test after the experiment

The results collected from the experiment are shown in the Table 7.7. As expected, with an increase in the applied RTT the QoE of the video degraded rapidly.

Table 7.7: Experiment results when downloading at 720p YouTube quality for RTT = 50ms

RTT = 50ms				MOS	
Added Loss (%)	No. of buffering events (N)	Length(s) of Buffering (Secs)	Average Length (L)	Analytical	Recorded
0.01	0	0	0	5	5
0.1	0	0	0	5	5
1	0	0	0	5	5
5	1	2.11	2.11	3.61	4
10	2	3.40 1.31	2.35	2.68	3
15	6	4.59 2.28 1.06 1.49 1.42 0.32	1.86	1.71	2
20	11	23.99 2.61 2.77 3.24 4.95 6.80 4.29 4.49 3.21 4.92 3.78	5.91	1.50	1

As a comparison with the earlier case when the RTT value was the same i.e. 50ms, buffering events were seen in the video download due to the download quality being forced to stay at 720p. Figure 7.26 below shows a plot of both the predicted and the recorded MOS.

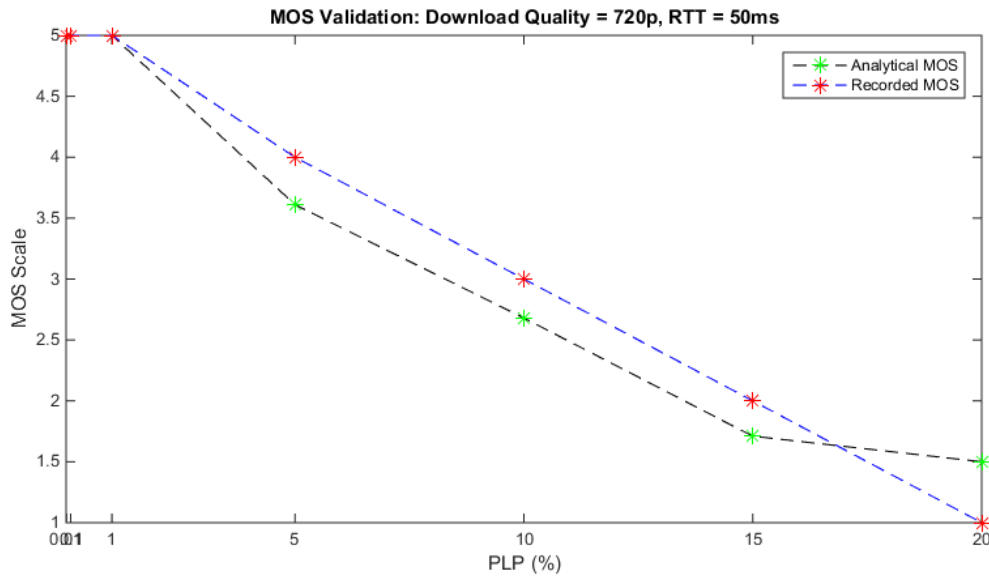


Figure 7.26: Experimental and analytical MOS for 720p YouTube download quality and RTT of 50ms

This MOS validation plot shows that both types of MOS (predicted and actual) have the same value for first three runs of the experiment and for the packet loss up to 5%. From that point on, there was a slight difference in the MOS values recorded by the participant and that of the predicted MOS. Despite this slight offset in both set of MOS points, the overall trend of the curves is identical.

7.7.3 RTT = 100ms

This experiment was run having the RTT value increased to 100ms. It was expected that the QoE of the video would degrade more quickly than in the case of the same experiment when the download quality was automatic.

As always, the before and after experiment speed test was run on the Internet link out to the YouTube server and the approximate returned speed was much higher than the benchmark of the required speed.

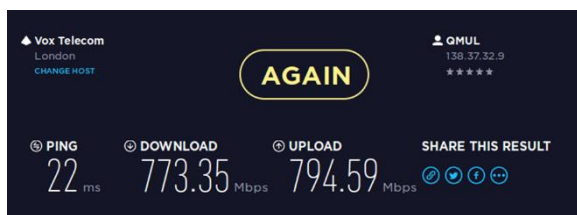


Figure 7.28: Speed test prior to the experiment

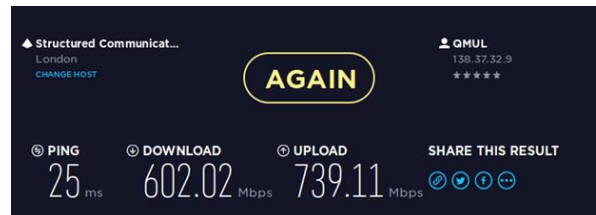


Figure 7.27: Speed test after the experiment

Table 7.8 outlines the experimental results from this part of the experiment. As expected, it can be seen that the presence of buffering was seen starting from an even smaller amount of added packet loss of 1%. Then with an increase in the packet loss, many more buffering events were noted during the video download.

Table 7.8: Experiment results when downloading at 720p YouTube quality for RTT = 100ms

RTT = 100ms				MOS	
Added Loss (%)	No. of buffering events (N)	Length(s) of Buffering (Secs)	Average Length (L)	Analytical	Recorded
0.01	0	0	0	5	5
0.1	0	0	0	5	5
1	1	1.19	1.19	3.92	4
5	2	1.55 1.39	1.47	3.04	3
10	2	1.23 4.11	2.67	2.57	3
15	3	6.07 4.33 6.60	5.66	1.65	2
20	5	9.96 7.97 7.17 5.52 7.38	7.6	1.50	1

A graph was plotted to visualise the MOS recorded by the participant to that of the expected MOS, see Figure 7.29.

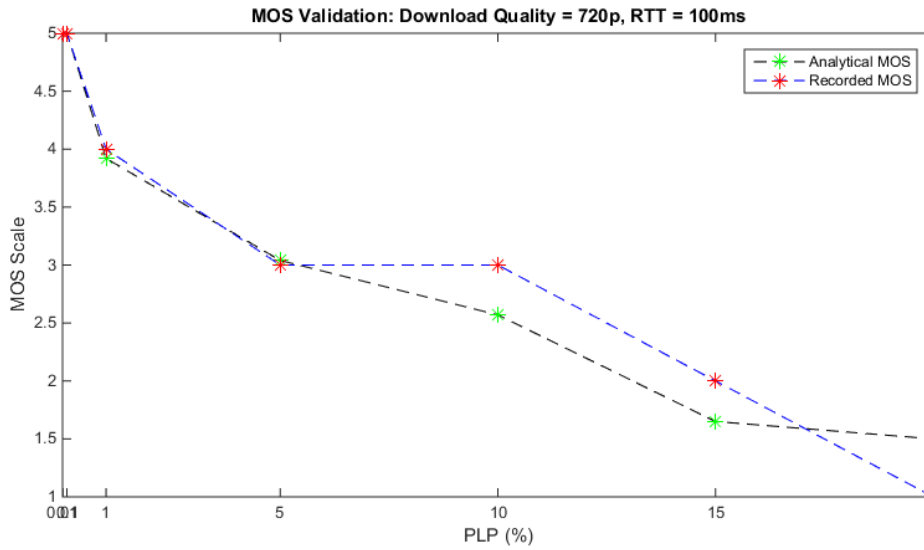


Figure 7.29: Experimental and analytical MOS for 720p YouTube download quality and RTT of 100ms

It was noted that for the first four instances of added packet loss of up to 5%, both models of the MOS are very closely aligned with each other. After 5% PLP though, the plotted MOS lines show that the participant gave a higher MOS score than predicted.

7.7.4 RTT = 150ms

This was the last experiment in this set of experiments. The RTT value was made 150ms. It was no mystery to expect or see the lowest level of QoE in this case of the VOD study. The speed test on the link was carried out before and after the experiment. The shown approximate Internet speed was sufficient to prove that the external link condition would not affect the experiment and the collected results.

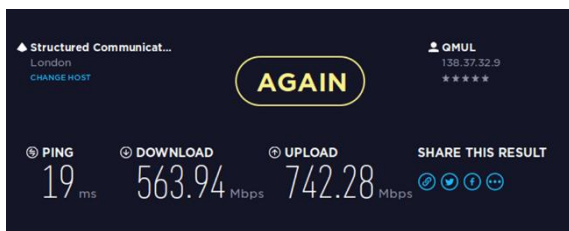


Figure 7.31: Speed test prior to the experiment

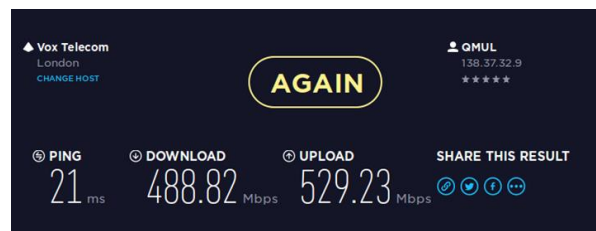


Figure 7.30: Speed test after the experiment

Table 7.9, which is much smaller than all other tables present in this series of experiments, shows that the record MOS dropped to a 1 much more quickly than all previously conducted similar experiments. The collected results laid out in Table 7.9 show that buffering in the video was seen right from the beginning of the experiment with the applied packet loss being as low as 0.01%. Then by the point of reaching a PLP of 5%, the participant had dropped the MOS all the way to a 1. At this point, 6 buffering events were noted in the video download.

Table 7.9: Experiment results when downloading at 720p YouTube quality for RTT = 150ms

RTT = 150ms				MOS	
Added Loss (%)	No. of buffering events (N)	Length(s) of Buffering (Secs)	Average Length (L)	Analytical	Recorded
0.01	1	0.49	0.49	4.19	5
0.1	1	0.87	0.87	4.04	4
1	1	1.03	1.03	3.98	4
5	6	11.58 2.13 8.22 0.90 1.00 1.59	4.24	1.52	1

Another interesting thing in this last experiment is that the participant skipped some of the ratings of the MOS altogether, namely 3 (fair) and 2 (poor). This is because the QoE on the video was changing so rapidly with only a slight addition of the packet loss that the participant perceived the video quality to be jumping from a 4 (good) straight to a 1 (bad) without assessing the quality of the video against the scores of a 3 (fair) and a 2 (poor).

This behaviour of the video QoE was of course expected as the combination of an excessive amount of applied RTT of 150ms, and the applied packet loss in addition, would degrade the video quality significantly.

Figure 7.32 shows a MOS comparison plot of both the MOS recorded by the participant, and the MOS expected from a given number of buffering events and their length. As some MOS points were not recorded at all by the participant, this plot looks different to the previous plots in this series of experiments. Unlike previous plots, this graph in Figure 7.32 shows that both MOS models start at different points of the MOS scale on the y-axis. This is because buffering events were present in the video download from the beginning and even for the PLP being as low as 0.01%. Hence the analytical MOS would take those buffering events into consideration and then calculate the MOS, whereas the participant ignored the buffering events in the first run of the experiment and still gave a generous MOS score of a 5.

Then the last point of the MOS on the plot differs in both cases too, which is due to the limitation of the QoE formula which is not being able to go any lower than 1.5 against the lowest rating of a 1 on the MOS scale.

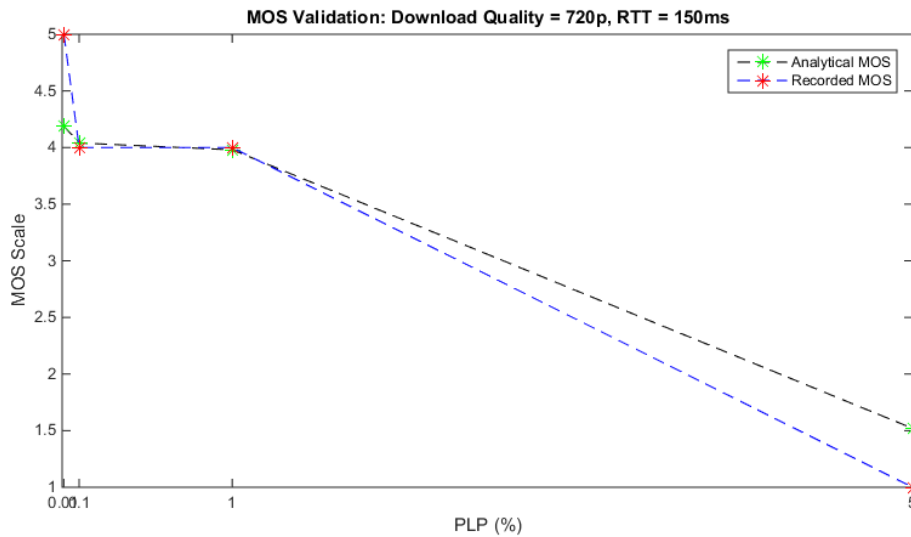


Figure 7.32: Experimental and analytical MOS for 720p YouTube download quality and RTT of 150ms

However, one important fact that is outlined by the graph in Figure 7.32 is the knee point in the case of both models of the MOS. This step drop in the MOS is seen at a PLP of 1% and is at the same point for both versions of the MOS.

7.8 Conclusion for YouTube QoE Experimentation

7.8.1 YouTube Formats and Codecs

The video formats supported by YouTube are: .Mov, .MPEG, MP4, .AVI, .WMV, .MPEGPS, .FLV, 3GPP, and WebM as outlined on YouTube support page [153]. YouTube has a guideline as to what the best settings are for a video that is to be uploaded. It is suggested to use the MP4 container, with a H.264 video codec. YouTube advises the content recreators to upload videos with a video frame rate that the videos were originally recorded in, e.g. 24, 25, 30, 48, 50, 60 frames per second [154].

As video bit rate is a very important factor that directly affects the quality and size of the video, YouTube has given a very detailed table of the supported bit rates for both the standard and the high definition video resolutions. These tables that outline the recommended bit rates for SDR and HDR are given in Figure 7.33 and Figure 7.34.

Type	Video Bitrate, Standard Frame Rate (24, 25, 30)	Video Bitrate, High Frame Rate (48, 50, 60)
2160p (4k)	35-45 Mbps	53-68 Mbps
1440p (2k)	16 Mbps	24 Mbps
1080p	8 Mbps	12 Mbps
720p	5 Mbps	7.5 Mbps
480p	2.5 Mbps	4 Mbps
360p	1 Mbps	1.5 Mbps

Figure 7.33: Recommended video bitrates for SDR uploads

Type	Video Bitrate, Standard Frame Rate (24, 25, 30)	Video Bitrate, High Frame Rate (48, 50, 60)
2160p (4k)	44-56 Mbps	66-85 Mbps
1440p (2k)	20 Mbps	30 Mbps
1080p	10 Mbps	15 Mbps
720p	6.5 Mbps	9.5 Mbps
480p	Not supported	Not supported
360p	Not supported	Not supported

Figure 7.34: Recommended video bitrates for HDR uploads

Recommended audio codec is AAC-LC with audio bit rates Mono at 128 kbps, Stereo at 384 kbps, and 5.1 at 512 kbps [154]. It is recommended that to view 4K uploads, a browser or device with a support of VP9 should be used.

When a video is uploaded on YouTube, it is encoded regardless of what format it was originally in. Supposedly this YouTube conversion of video is done in H.264 [155].

7.8.2 Effect of Packet Loss on QoE of YouTube Traffic

Having found previously that YouTube codec is sensitive to network jitter, during this experimentation, YouTube traffic’s behaviour was studied in the presence of loss only. To maintain the accuracy, a short YouTube video was chosen for the study to avoid longer caching process which would be the case for a long duration video. A short video would also allow the buffering events to be less in number, hence the monitoring of buffering events would be easy with a higher control. The quality of this video was not high definition, and it was played back with a normal speed and at a 360p resolution.

When users were asked for feedback after the experiment was complete, users provided information that the most annoying aspect which lead to a drop in their given MOS rating was the number of

buffering events and the duration of each buffering event. It was outlined that although the time taken to load the YouTube page and the initial time taken to play the video could contribute towards getting a low MOS, these two factors did not have a major impact unless they added up to a significantly longer waiting time. However, the stalling during the playback after the video has started playing leads to a very quick dissatisfaction of the user. This agrees with earlier research [156] [157] [158].

By looking at Table 7.1, it is witnessed that the video was continuously rated with a MOS of a 5 for all PLP levels from 0.01% to 1%, until the PLP reached 10%, when the MOS dropped to a 2. This was because the video starting time, in the presence of 10% PLP, as compared to a 1% PLP, took an extra 13.75 seconds (18.47 seconds – 4.72 seconds) and the time to fully load took an extra 71.91 seconds. Furthermore, the longer buffering times for the 5 buffering events which lasted for about 10 seconds on average all contributed towards the user being dissatisfied and losing interest in the video, hence they rated the video quality to be very annoying. These factors are the ones that mainly affect the QoE on streaming services such as YouTube.

If a comparison is made between the YouTube streaming experiment and the local network streaming with TCP protocol, the results from both experiments show quite a different behaviour. The MOS of YouTube video only dropped at a 10% PLP, and before that level of jitter the video was rated to be of a good watchable quality. However, in the TCP experiment the video quality was affected from packet loss with a PLP as low as 0.01%, and was severely affected when the PLP reached 0.1%.

The authors in [145] and [156] have carried out analysis on YouTube video QoE. When there are buffering events present during the playback of a video, the QoE is very likely to be lower than 5 [145]. It was suggested that MOS rating drops as the number of buffering events, and the duration of the buffering, increases. In the experiments in this thesis it is seen that with a PLP of 10%, the QoE of the YouTube degrades leading the QoE to drop to a MOS of 2, which is very close to MOS = 1 that the researchers suggest in [145].

7.8.3 Effect of RTT and Packet Loss as a Combination

Having seen the results and behaviours shown in this series of experiments for the overall study of the QoE, it can be witnessed that it is not just the packet loss that affects the QoE, but the packet delay in the form of RTT also plays a major part in improving or worsening the QoE for the VOD. Therefore, the combination of the PLP and RTT has a direct impact on the QoE of VOD. These two factors affect the QoE of VOD in such a way that the more the amount of the packet loss and packet delay whether individual or taken as a combination, the worse the QoE and vice versa.

7.9 Chapter Summary

In this chapter, experimentation was conducted on YouTube traffic. The aim was to scale the research to network traffic from across the Internet with the aim to investigate the “real world” network traffic. The first series of experiments included a QoE evaluation in the presence of packet loss on the link the YouTube traffic was transported over. This study was regarded to be of vital importance since with Internet traffic in general, YouTube accounts for about 18% of the overall Internet traffic [159] [160] and the YouTube share of the multimedia traffic alone on the Internet is about 78.8% [161]. These experiments were done with applied packet loss and TCP was used as a choice of transport protocol since TCP is used by YouTube over the use of UDP. User behaviour was studied against the VOD QoE in two cases: when a video does not load immediately or stalls during the playback. Researchers in [156]

[157] [158] report that once the initial playback of video has started and buffering takes place it frustrates the user enormously. During this part of the experimentation, due to different levels of applied packet loss, the video occasionally took slightly longer to load initially but played continuously afterwards. However, in other instances, especially with higher level of applied packet loss, video also buffered during the playback in addition to preliminary loading of the YouTube interface and loading the video for initial playback. Seeing the user-recorded MOS it was noted that users were bothered more by the mid-playback buffering than the initial loading of the page and the video itself. This is in line with earlier findings in [156] [157] [158]. With buffering present in the video, the recorded MOS will possibly be less than a 5 [145]. With the work presented in this chapter, it was then concluded that the factors that affect the QoE of video streaming services are video starting time, time to fully load the video, and mid-stream buffering with longer durations.

To stay in line with the behaviour of traffic from conventional networks, in the next set of experiments for YouTube QoE evaluation, packet loss and packet delay were applied as a combination on the Internet video traffic coming from the YouTube server and going into the NetEm server, then out to the client computer on the testbed. Experimentation was carried out with a range of RTT/packet delay values: 1ms, 50ms, 100ms, and 150ms. With each level of applied RTT, the applied packet loss was varied among 0.01%, 0.1%, 1%, 5%, and 10%. The collected results showed that it is not just the packet loss that affects the QoE of VOD; Packet delay (round trip time) also has a direct impact on the QoE of VOD which agrees with the research findings in [48] and [45]. It was seen that the higher the amount of the packet loss and packet delay present on the link, the worse the QoE. The packet loss and delay could be standalone, or affecting the network traffic as a combination.

Chapter 8 Significance of VOD QoE Evaluation Across Different Age Groups

8.1 Chapter Introduction

The experimentation in this Chapter was conducted using a custom delay distribution. For details about using NetEm with advanced functionality refer to Section 4.11 on page 59. The implemented delay model is based on [80] and [79].

Through experimentation, the relationships between PLP and the user-perception metric QoE for VOD was evaluated. A full range of age groups was included in this study since experimentation over different age groups has been largely absent in earlier work mapping QoS metrics to QoE. The results of this experiments were recorded as a MOS score. See Figure 2.1 for a description of MOS scale and its relation to the service quality. It is well known that these MOS scores vary depending on the protocols, applications and transport layer protocol being used. This was further explored details of which are given in the following sections.

8.2 Experimentation

As seen from the results in Chapter 6 on page 104, as well as the fact outlined in [82], and [83], it is noted that the perceived QoE can differ widely for users of different age groups, people of youngest age group being the most demanding. This demanding age group (the youngest) requires (at the critical Service Level guaranteed PLP of 0.001) an order of magnitude lower packet loss probability to achieve the same QoE. Then the results were applied using a known relationship between PLP and network capacity to provide an insight into the potential cost of re-dimensioning a network to probabilistically guarantee the lower of the two PLPs. The lower PLP is the one associated with the higher network capacity required to probabilistically assure the lower PLP.

MOS scores were recorded from participants over different age groups: 10–18 years, 19-30 years, 31-45 years, 46-65 years, and over 65 years old. The same video with the same range of added packet loss was shown to all the participants. Having collected the opinion of different people in the same age group, an average was then calculated to form the MOS for that age group. In general, there was little observed difference in MOS scores over the ranges of participants, except for the youngest group.

The results of fitting MOS results for two key groups of experimental subjects are summarized in equations (8.1) and (8.2). Equation (8.1) gives the MOS formula for subjects in the age group 19-30, the most commonly studied age group, while equation (8.2) gives the MOS formula for the most demanding age group (10 to 18-year-olds).

$$MOS_{19to30} = 1.788e^{-10.05PLP} + 3.387e^{-0.1239PLP} \quad (8.1)$$

For (8.1) the goodness of fit measures was:

SSE: 0.00094

R – Square: 0.9999

RMSE: 0.03066

$$MOS_{10to18} = 2.402e^{-9.918PLP} + 2.127e^{-0.08282PLP} \quad (8.2)$$

For (8.2) the goodness of fit measures was:

SSE: 0.01797

R – Square: 0.9975

RMSE: 0.1341

The standard, guaranteed PLP value for most commercial broadband networks is $PLP = 0.001$ as given in industry technical standards [162], Section 3 of [163], and BTnet SLA [1].

At a $PLP = 0.001$ the results for the two age groups as given in equations (8.1) and (8.2) are $MOS = 4$ and $MOS = 3$. These results show that to achieve $MOS = 4$ for the more demanding age group requires an order of magnitude improvement in PLP, i.e. from 0.001 (0.1%) to 0.0001 (0.01%).

Then, the possible consequences of this requirement to lower PLP on the dimensioning of network capacity were investigated, an issue of growing importance as this is seen to be of paramount significance in delivering a satisfying video quality to the end user seeing the massive growth of Over-the-top (OTT) online video [164]. Network capacity dimensioning was put under focus not because it is the only way in which these results are of significance, but because network capacity dimensioning is possibly the fundamental network engineering challenge.

8.3 Analysis of Effect of Age Related MOS Display on Capacity Dimensioning

In [165] the formula reproduced below as equation (8.3) was derived for multiplexed TCP sources through a bottleneck link. It is a piece of theoretical work where number of traffic sources is N . The sources are all TCP sources multiplexed together. The use of a bottleneck network topology structure is

quite traditional in performance evaluation studies in networking, and it still widely used now, e.g. see [166], and I adopt this topological structure here.

$$PLP = \frac{32 \cdot N^2}{3 \cdot b \cdot (m + 1)^2 (RTT \cdot C + Q)^2} \quad (8.3)$$

m = actor by which TCP sending rate is reduced on loss (usually ½)

b = number packets acknowledged by an ACK packet, usually 1

RTT = round trip time in seconds

Q = buffer length in packets

C = bottleneck capacity in packets per second

Equation (8.3) relates network bottleneck capacity, C , to the number of TCP sources, N , the round-trip-time, RTT , and the required PLP . I now use equation (8.3) to deduce the required increase in capacity that would be needed to reduce the PLP by a single order of magnitude, as required by the most demanding group of network users. I do this for a range of RTT 's and capacity values at the bottleneck link.

Thus:

$$PLP_{low} = \frac{32 \cdot N^2}{3 \cdot b \cdot (m + 1)^2 (RTT \cdot C_{lowPLP} + Q)^2} \quad (8.4)$$

$$PLP_{high} = \frac{32 \cdot N^2}{3 \cdot b \cdot (m + 1)^2 (RTT \cdot C + Q)^2} \quad (8.5)$$

Where:

PLP_{low} = packet loss probability low, i.e. achieved when capacity is the higher value

PLP_{high} = packet loss probability high, i.e. achieved when capacity is the lower value

C_{lowplp} = the higher capacity value, i.e. the capacity that gives the lower packet loss probability

C = the original (lower) capacity value, i.e. the capacity that gives the original (higher) packet loss probability.

Dividing (8.4) by (8.5) gives:

$$10 = \frac{(RTT \cdot C_{lowPLP} + Q)^2}{(RTT \cdot C + Q)^2} \quad (8.6)$$

The original value of C is known, as it will be the capacity value that was dimensioned into the network to ensure the QoE of the majority of users. I now determine the proportional increase in C needed to achieve a one order of magnitude improvement in PLP , as required by the most QoE-demanding users (the youngest group of users).

Rearranging (8.6) by taking logs ($\log(10) = 1$) yields:

$$= 2 \cdot \log(RTT \cdot C_{lowPLP} + Q) - 2 \cdot \log(RTT \cdot C + Q) \quad (8.7)$$

$$1 + 2 \cdot \log(RTT \cdot C + Q) = 2 \cdot \log(RTT \cdot C_{lowPLP} + Q) \quad (8.8)$$

$$10^{(1+2.\log(RTT.C + Q))/2} = RTT.C_{lowPLP} + Q \tag{8.9}$$

$$C_{lowPLP} = (10^{(1+2.\log(RTT.C + Q))/2} - Q) / RTT \tag{8.10}$$

Then the factor by which the capacity needs to be increased is found as:

$$\text{factor of capacity increase needed} = C_{lowPLP} / C \tag{8.11}$$

8.4 Results

Figure 8.1 and Figure 8.2 give results for a very large bottleneck capacity: 1Gbps. Both Figure 8.1 and Figure 8.2 show the ratio of the increased capacity for the lower PLP divided by the original capacity, C (i.e. the “factor of capacity increase needed”) plotted against the RTT for a range of RTT values increasing from 1 millisecond to 10 seconds. A study of this large range of RTT was chosen because the actual existence of such a wide range of RTT is supported by prior empirical studies in TCP RTTs [167].

Figure 8.1 is for a bottleneck queue size of 1000 packets, using guidelines in [168]. It is clear that the capacity increase required very quickly converges on a value just in excess of 3.

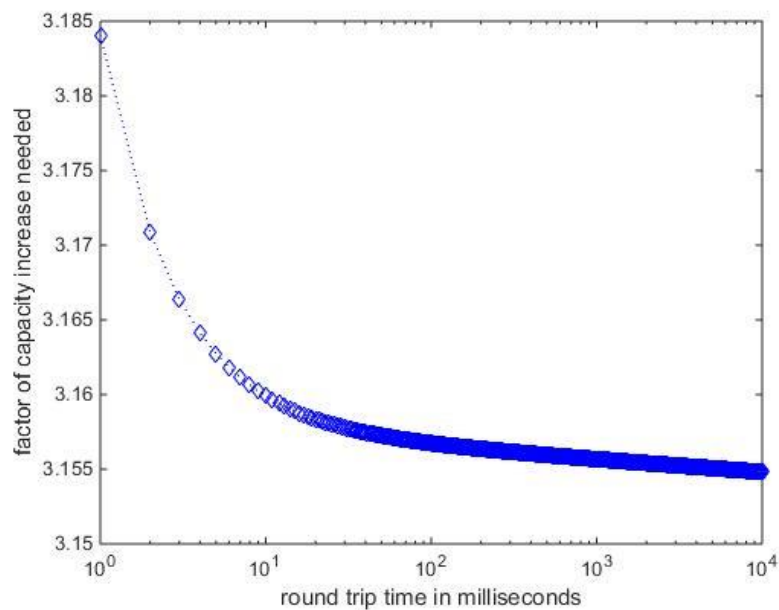


Figure 8.1: Capacity increase factor for C = 1Gbps and Q=1000

Figure 8.2 affirms this level of required capacity increase (just in excess of 3) even when the bottleneck buffer capacity is increased 10-fold.

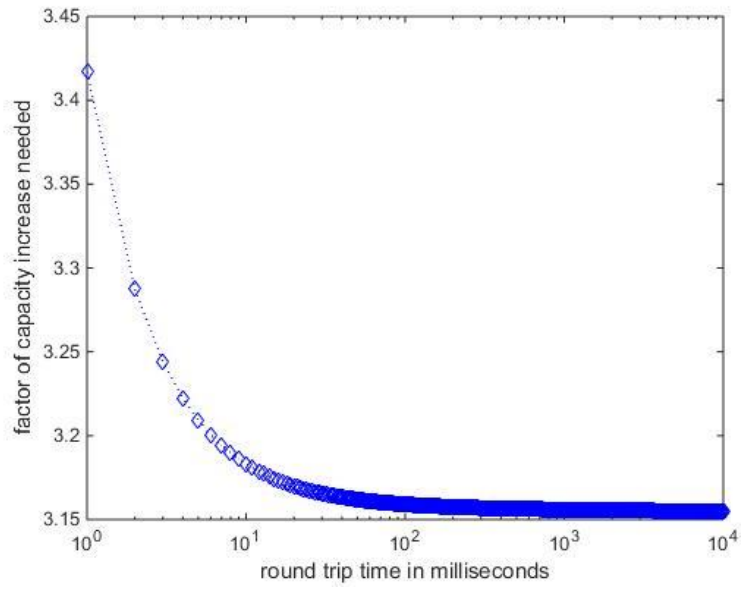


Figure 8.2: Capacity increase factor for C = 1Gbps and Q=10000

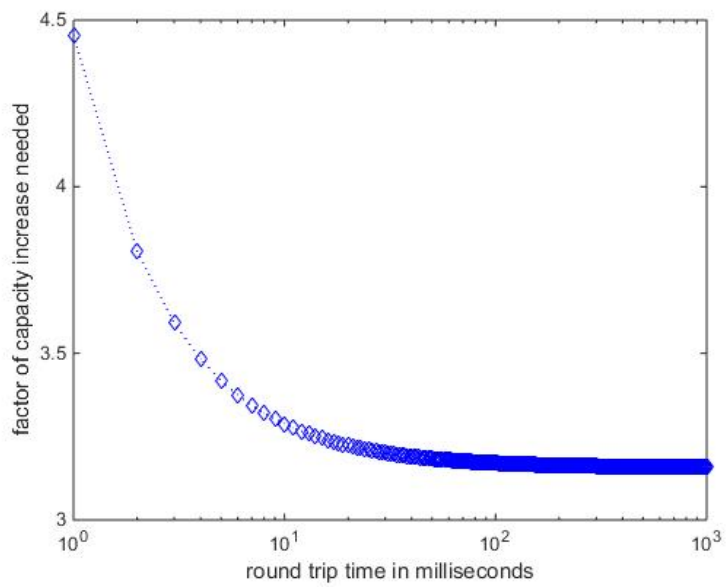


Figure 8.3: Capacity increase factor for C = 20Mbps and Q=1000

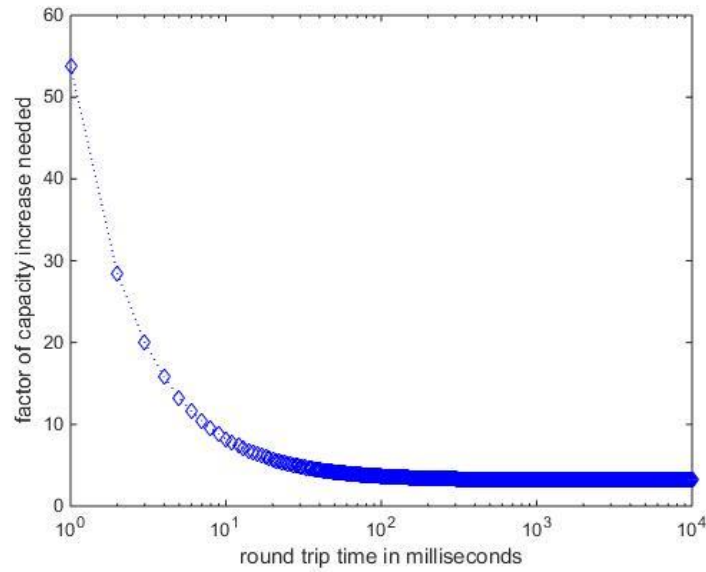


Figure 8.4: Capacity increase factor for C = 512kbps and Q=1000

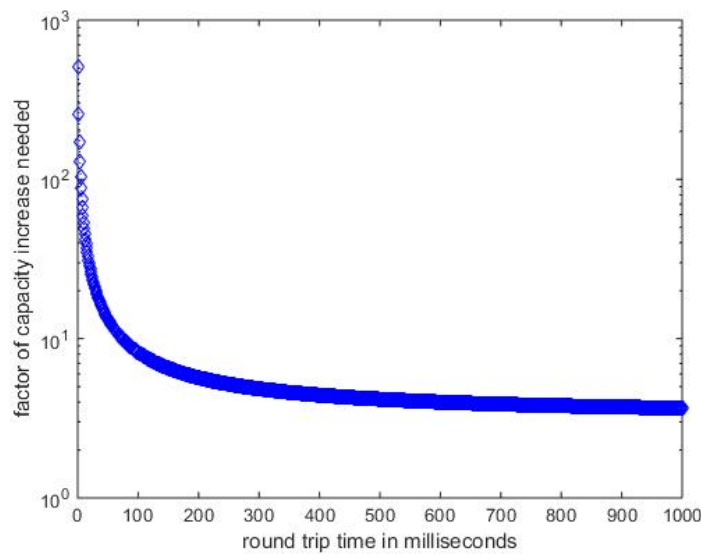


Figure 8.5: Capacity increase factor for C = 512kbps and Q=10000

Figure 8.3 shows the capacity increase factor for a 20Mbps bottleneck link, i.e. one that consist of 20 x E1 links [169]. This factor is again just larger than 3 for all RTT values that are not very small (i.e. are larger than 10ms). Figure 8.4 repeats for a much smaller bottleneck link capacity of 512kbps. It is here seen that the RTT has a much stronger effect, and the required capacity increase doesn't reach around a factor of 3 until the RTT has reached around 100 milliseconds.

Figure 8.5 shows that, at low levels of bottleneck capacity (512kbps), the effect of the buffer size becomes much more significant, with the capacity increase factor not reaching as low at 3 until RTT has reached some hundreds of milliseconds. For this reason, the manufacturer recommendations for queue sizing in the router buffers was examined.

8.5 Default Queue Sizing for Router Buffers

Now the effect of using the default queue sizing recommended for Cisco routers was considered, using Cisco's own instructions as a guide [170]. Specifically, the packet queue depth was adjusted such as to allocate the (default) of 50 milliseconds worth of buffering at the link capacity (assuming 1500 byte packet sizes), while ensuring, as per these instructions, that this allocated queue depth never falls below 64 packet spaces (again for 1500 byte packet sizes).

For a bottleneck link of 512kbps (the worst case of our earlier evaluations) $Q = 64$ packets was set. This produced the set of results shown as Figure 8.6, which should be contrasted with Figure 8.5.

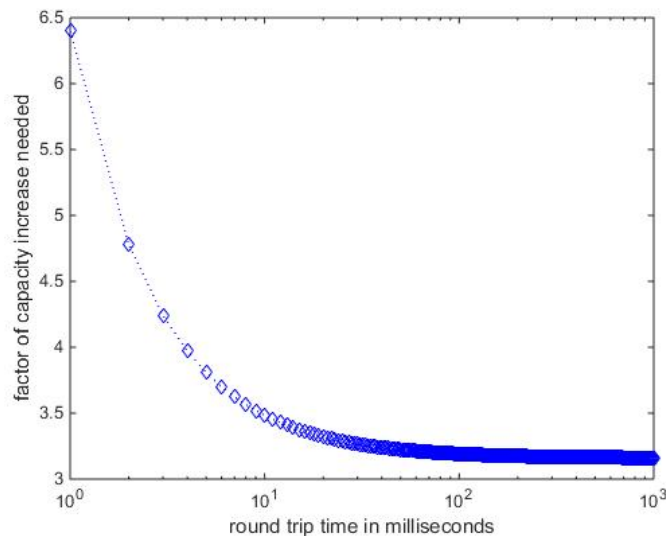


Figure 8.6: Capacity increase factor for $C = 512\text{kbps}$ and $Q=64$

In Figure 8.6 it is clearly seen that the capacity increase factor very rapidly converged, again, to just in excess of 3 times the original capacity.

It was concluded that the difference between the most demanding age group (10 to 18 years old) and the most frequently studied age group (19 to 30-year-old) is such as to require an order of magnitude improvement in the network PLP to achieve the same MOS score. The study was conducted on the capacity increase needed to reduce the PLP by an order of magnitude through a bottleneck link in which the queuing is caused by multiplexed TCP traffic. The results show that an order of magnitude improvement in PLP would require very close to a 3-times increase in bottleneck link capacity at all but very low RTTs

8.6 Chapter Summary

In the final part of experimentation in this research, the importance of QoE evaluation across different age groups on VOD was studied with some recommendations on network capacity required to meet the needs of most demanding age group. This Chapter advanced on the work done in Chapter 6 but with a

focus on a network capacity recommendation to fulfil the network needs of the most demanding age group emerged out of the study on QoE evaluation across different age groups.

Findings from this set of experimentation could be very valuable to network companies. From a service provider's prospective, if it is now possible to identify a certain group of customers, such as elderly people who may be showing a little less consideration to subscribe to the fastest available service with top end download speeds, this could help the ISPs to utilise the available bandwidth more efficiently by distributing it accordingly among casual users and heavy users. This bandwidth allocation could be done on a contractual level, or through load balancing dynamically by adapting to the traffic conditions over a certain period of time in question.

Chapter 9 Discussion, Recommendations for Future Work, and Conclusion

9.1 Discussion

Having a good level of service from a subscription is of great significance to customers who pay to receive service. If customers are not satisfied with the service provided, they will switch to another service provider where they are able to get a more satisfactory service as value for the money they pay. Henceforth, to maintain customer loyalty, it is very important to provide a satisfactory service to customers; this in return will increase the revenue and maintain a prosperous business.

QoE is the user-perceived quality of a service for a range of services offered today: VoIP, IPTV, and web browsing are a few to outline. Knowledge of QoE can be used by service operators to help develop and manage their networks [13]. For a detailed description of QoE, refer to Section 2.1 on page 21.

Many organisations approach QoE optimisation by improving broadband speeds, which potentially misses the role of jitter and loss in affecting QoE [171]. For example, studies of latency and app errors have shown a positive correlation between them [172]. Research in [171] suggests as many as 48% of users would uninstall or stop using an app if it regularly ran slowly and 53% of users would uninstall or stop using an app if it regularly crashed, stopped responding or had errors. It has been recently suggested [172], that, as users become more aware of the effects a network has, as opposed to blaming the application itself, they will use this as a criterion for whether the mobile service provider is delivering a satisfactory service.

This research was conducted with an aim to study QoE and to explore the key points that could give an insight into the ways customers expect service from a provider. For this study, VOD traffic was focused on to come up with findings that would align with consumer needs for the modern trend in network

services. As streaming accounts for 70% of total Internet traffic [159], hence it was decided that analysis of video traffic would be the focus of this state-of-the-art research on QoE.

To be able to conduct this research, a new testbed was designed that allows the experimenter to study the effects that a combination of PLP and jitter has on the quality of the videos, for both TCP and UDP. The effects of PLP and jitter on the videos was subjectively rated as QoE, using the standard MOS scale.

The designed testbed is based on Network Emulator (NetEm) from the Linux Foundation [70]. If used with the default settings, NetEm has some specific limitations as any simulation/emulation tool would have. These specific limitations compromise the authenticity of achieved results that must be analysed to reach a conclusion and make further useful decisions or recommendations for the design of a network in question. Therefore, it is very important to have these technical limitations addressed and resolved to be able to emulate realistic network behaviour.

The newly designed testbed addressed the aforementioned issues. To be able to model real world network patterns, the use of NetEm default jitter model was dropped and self-designed jitter models were implemented. These jitter models, also referred to as delay distribution tables in this research, were created by gathering a large set of statistical delay values. These delay distribution tables were created from many hundreds of delay patterns that had not been initially defined. These delay patterns statistically correlated to the delay parameter of a given standard deviation and were random in nature. Once implemented in NetEm, these custom delay distributions added random delays on all outgoing traffic on the egress port whilst bringing the network delay patterns very close to that seen in real world network traffic (refer to Section 4.11 on page 59 for details on implementing custom delay distribution tables in NetEm). The second NetEm flaw, losing the packet order in the presence of high jitter, was resolved by replacing the TFIFO queueing discipline with a PFIFO queue (see Section 4.8 on page 52 for details). Additionally, to be able to emulate real world network traffic realistically, this state-of-the-art testbed also facilitated the application of jitter, delay, and packet loss collectively which has not been done in any NetEm-based research previously.

In previous research on QoE, research participants for Mean Opinion Score (MOS) evaluation were only who are normally found in academia: people in the age groups 19 to 30 and 31 to 45 years of age. In this research, the effect of age on reported QoE (MOS) scores for VOD in broadband networks has been studied and MOS evaluation was done across a range of age groups, involving those not traditionally part of such research, teenagers and elderly people. It is concluded that most demanding age group, 10 to 18 years old, requires an order of magnitude improvement in the network PLP to achieve the same MOS score as compared to the most commonly studied age group in academia which is 19 to 30 years old. A bottleneck link was then investigated that has multiplexed TCP traffic queueing in it. This analysis was conducted to study the amount of capacity that would be required to give an order of magnitude improvement in the PLP. It was seen that it will require very close to 3 times the bottleneck link capacity to be able to give an order of magnitude improvement in the PLP.

The findings out of the experimentation in this research of VOD QoE evaluation across different age groups, are shown to offer potential commercial value. To control QoE on a contractual level for example, the ISPs could assess the usage and needs of a customer from a certain group. Discovering that a user group may not be using the service after, say, 11pm until 6am and will not have intensive usage during the day either, may allow the ISP to offer them a cheaper service package. This would benefit the customer as they will only be paying for the service to the extent they will be using it. Similarly, it would

benefit the ISP through not having to worry about the service down-time during the off-peak time of 11pm-6am. Hence the ISP would be able to focus on providing an efficient service to a heavy user group who could be annoyed if the service was down even for 1 minute in 24 hours. This way, the service provider will be able to stay within the terms stated in their SLA. This could be studied from service providers perspective as future work building on this research.

In dynamic load balancing on the other hand, the ISP could give all customers across all user groups the same service at a higher cost, while acknowledging that a certain group of customers may not be utilising the services to the fullest in off peak times. This could allow the ISP to temporarily provide a big proportion of these customer's bandwidth to other users who may need it more, but still charge all customers for a full featured service package, which would earn more revenue for the ISP. Although this approach will lead better profits for the ISP, however, this could be risky from the SLA point of view. This is because on an average day during the of peak time when normally a casual user does not use the service, if they decide to use the service at the time which generally falls out of their routine usage, the customer may face service shortage or a poor speed due to the usual off-peak bandwidth allocation, which could lead to breaching the service terms stated in an ISP's SLA.

From the way a service provider chooses to offer the level of service to a certain customer, whether on a contractual-level or dynamically-controlled service, being in a position to identify how a certain group of users respond to QoE can hugely contribute towards the service providers making important decisions for maintaining a prosperous business. Therefore, the research documented in this thesis can make a real impact in the industry.

9.2 Future Work

9.2.1 General Recommendations

If one was to take this research further, studying the effects of jitter and delay on video calling such as Skype could be the next area of importance. In video-calling applications, it is often seen that when network is up without any congestion the video and audio during the call is seen to be very clear. However, as soon as the network becomes slightly unstable, which could be due to congestion, packet loss or delays, the video quality starts to degrade. It is frequently seen that the video may not be of a good quality, but people can still speak and understand each other. Then if the network quality keeps getting worse, there comes a point when two people struggle to understand each other due to the delays in reaching their voice on the other end. Then if the network quality drops even further, eventually the audio stops too leading to a silence. At this point, the call may not drop and the automatic call reconnect is attempted. If this call quality cannot be improved after a certain time despite of several attempts, the call then drops and both parties have to manually establish the call again. A study into the above-mentioned thresholds, i.e. what level of jitter/loss makes the video freeze initially or drops the call in the end, could be very enlightening.

This research focused on progressive video-download, the mechanism used by YouTube to deliver a video to a user. An increment to this research could be studying the QoE on a file download, for example, many companies deliver commercial software by directing users to their download page on their website. An example is Ubuntu, when one wishes to get the open source operating system from Ubuntu they are asked to download the operating system in a file then proceed with the installation.

This downloading file could also be a high-quality video file from video subscription companies such as Netflix or Amazon Video. Research then could be done on the duration of the download normally versus in the presence of packet loss or/and jitter to determine what affects the file transfer more severely. Furthermore, it could be studied what causes the file to get corrupted during the download. Is it the packets getting out of order or the jitter becoming higher than a certain level? if so, what level of jitter could corrupt a file download?

9.2.2 Technical Recommendations

Researchers wishing to utilise the ground-breaking features of this testbed are advised to take the following steps to take a full advantage of this testbed and ensure the accuracy of the results:

1. It should be ensured that a stable version of Ubuntu operating system is used.
2. Download the latest version/commit of the iproute2 from Github, refer to Section 4.11.1 on page 59 for instructions on how to do that.
3. For NetEm commands, use the correct Ethernet interface that the client computer is connected to i.e. eth0, eth1, eth3, and so on, otherwise the input commands will throw errors.
4. If producing custom delay distributions, ensure to use meaningful names for dist files in such a way that the file name describes the standard deviation/jitter used for a certain delay distribution table, otherwise it can be a long process to understand the jitter range from the statistical values given in a table.
5. If using the bash scripts to automate the application of network metrics (jitter/delay, loss), ensure you are using correct names for delay/distribution tables and Ethernet interfaces.
6. When experimenting on internet traffic, it should be ensured that a highspeed wired Internet connection is used and not a wireless hotspot.

9.3 Conclusion

The findings out of this research have valuable commercial advantages equally for the service providers and service consumers. Service providers can use the research results to understand customer needs in a more meaningful way, hence offer a better service. After service providers have addressed the issue with the help of this study, service consumers will potentially be able to get a more promising service as a value for their money.

In addition to a use in commercial environment, the innovative testbed out of this research could be also be useful in academic research to study a range of network scenarios. This testbed is now being used by other researchers; such as the network research students at the university, and some start-up companies.

References

- [1] BT, "BTnet Service Level Agreement," [Online]. Available: https://business.bt.com/content/dam/bt/business/Networking/BTnet_leased_line_service_level_agreement.pdf. [Accessed 28 September 2017].
- [2] AT&T, "Service Level Agreement (SLA)," 3 August 2015. [Online]. Available: <http://cpr.att.com/pdf/se/0001-0003.pdf>. [Accessed 28 September 2017].
- [3] Broad Sky Networks, "Broad Sky Enterprise Service Level Agreement," Broad Sky Networks, [Online]. Available: <http://broadskynetworks.net/sla/>. [Accessed 10 September 2017].
- [4] Sprint Internet Services, "Service Level Agreement," Sprint Internet Services, 2016. [Online]. Available: <http://www.sprintinternetservices.co.uk/home/service-level-agreement/>. [Accessed 10 December 2016].
- [5] Spitfire Network Services Limited, "Service Level Agreement (SLA)," 15 April 2016. [Online]. Available: <https://www.spitfire.co.uk/wp-content/uploads/2016/04/Spitfire-Service-Level-Agreement-v2.pdf>. [Accessed 10 September 2017].
- [6] Witbe, "QoS vs. QoE," Witbe, [Online]. Available: <http://www.witbe.net/technologie/qos-vs-qoe/>. [Accessed 22 December 2014].
- [7] J. K. Choi, "Quality of Service (QoS) and Quality of Experience (QoE)," 10 09 2012. [Online]. Available: https://academy.itu.int/moodle/pluginfile.php/39877/mod_resource/content/1/session4.pdf. [Accessed 22 December 2014].
- [8] Technology Training, "Voice Quality Measurement," Technology Training, [Online]. Available: http://www.technology-training.co.uk/voicequalitymeasurement_39.php. [Accessed 22 December 2014].
- [9] TamoSoft, "MOS and R-factor," TamoSoft, [Online]. Available: http://www.tamos.com/htmlhelp/voip-analysis/mosandr_factor.htm. [Accessed 22 December 2014].
- [10] V. Troubleshooter, "Measuring Voice Quality," VoIP Troubleshooter, [Online]. Available: <http://www.voiptroubleshooter.com/basics/mosr.html>. [Accessed 22 December 2014].
- [11] J. Anuskiewicz, "Measuring jitter accurately," Spirent Communications, 24 April 2008. [Online]. Available: <http://www.lightwaveonline.com/articles/2008/04/measuring-jitter-accurately-54886317.html>. [Accessed 11 January 2015].

- [12] J. P. S. E. S. K. S. Poretsky, "IETF RFC 4689," Network Working Group, October 2006. [Online]. Available: <http://www.rfc-archive.org/getrfc.php?rfc=4689>. [Accessed 11 January 2014].
- [13] Huawei, "Moving beyond traditional network KPIs," Huawei, 12 June 2014. [Online]. Available: http://www.huawei.com/en/publications/communicate/60/HW_093296. [Accessed 12 May 2017].
- [14] University of Surrey, "Quality of Experience," University of Surrey, [Online]. Available: http://www.surrey.ac.uk/cvssp/research/quality_experience/. [Accessed 18 February 2015].
- [15] Webopedia, "QoE," Webopedia, [Online]. Available: <http://www.webopedia.com/TERM/Q/QoE.html>. [Accessed 18 February 2015].
- [16] M. Rouse, "Quality of Experience (QoE or QoX)," TechTarget.com, April 2008. [Online]. Available: <http://searchcrm.techtarget.com/definition/Quality-of-Experience>. [Accessed 18 February 2015].
- [17] AWTG, "Quality of User Experience- QoE," AWTG, 14 October 2009. [Online]. Available: <http://awtg.co.uk/tools.php?id=2>. [Accessed 11 February 2015].
- [18] K. K. P. R. Markus Fiedler, in *From Quality of Service to Quality of Experience*, Schloss Dagstuhl, 2009.
- [19] VideoClarity, "Understanding MOS, JND and PSNR," VideoClarity.com, [Online]. Available: <http://videoclarity.com/wpunderstandingjnddmospnr/>. [Accessed 16 August 2016].
- [20] Microsoft, "Mean Opinion Score and Metrics," Microsoft, [Online]. Available: <https://technet.microsoft.com/en-us/library/bb894481%28v=office.12%29.aspx>. [Accessed 13 February 2015].
- [21] Avvasi, "Measuring Quality of Experience for Over-the-Top Video Services," August 2011. [Online]. Available: https://www.2test.ru/upload/iblock/225/measuring-qoe-for-ott-video-_white-paper-_august-2011.pdf. [Accessed 03 March 2016].
- [22] T. H. Raimund Schatz, "Web QoE Lecture 1: Quality of Experience," 13-17 February 2012. [Online]. Available: <http://goo.gl/ermhvA>. [Accessed 12 February 2015].
- [23] QoE-systems, "Video Quality Testing," QoE Systems, 2010. [Online]. Available: Video Quality Testing. [Accessed 16 August 2016].
- [24] A. C. Martin Kastner, "Interpretation of MOS for video services," January 2011. [Online]. Available: <https://www.telchemy.com/reference/T09-SG12-C-0205!!MSW-E.pdf>. [Accessed 16 August 2016].
- [25] S. W. D. S. H. Robert C. Streijl, "Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives," *Multimedia Systems*, vol. 22, no. 2, pp. 213 - 227, 2016.

- [26] L. G. E. M. Y. I. Edgard Silva, "Mean Opinion Score Measurements Based on E-Model During a VoIP call," in *The Eleventh Advanced International Conference on Telecommunications*, Brazil, 2015.
- [27] A. Khan, "Video Mean Opinion Score (MOS) Test," Plymouth University, 10 March 2009. [Online]. Available: <http://www.tech.plymouth.ac.uk/spmc/staff/akhan/mostest/infocom.htm>. [Accessed 16 August 2016].
- [28] C. A. J. P. R. S. N. P. V. R. H. J. Dhvani R. Bhadra, "Packet Loss Probability in Wireless Networks: A Survey," in *Communications and Signal Processing (ICCSP), 2015 International Conference*, Melmaruvathur, 2015.
- [29] X. S. Y. B. M. W. A. V. V. H. W. Guan-Ming Su, "QoE in video streaming over wireless networks: perspectives and research challenges," *Wireless Networks*, vol. 22, no. 5, pp. 1571-1593, 2016.
- [30] "Interruption Probability of Wireless Video Streaming with limited video lengths," *IEEE TRANSACTIONS ON MULTIMEDIA*, vol. 16, no. 4, 2014.
- [31] T. S. Z. N. T. L. Peter Orosz, "A No-reference Voice Quality Estimation Method for OPUS based VoIP services," *International Journal On Advances in Telecommunications*, vol. 7, no. 1, 2, pp. 12-21, 2014.
- [32] R. M. M. A. K. S. R. M. C. Jiasi Chen, "A scheduling framework for adaptive video delivery over cellular networks," in *Proceedings of the 19th annual international conference on Mobile computing & networking*, Miami, 2013.
- [33] J. C. I. C. Andrea Fumagalli, "The MTIT Access Protocol for Supporting IP over WDM Ring Network," in *Communications, 1999. ICC '99. 1999 IEEE International Conference on (Volume:1)*, Vancouver, BC, 1999.
- [34] V. G. Karim Mohammed Rezaul, "A Novel Algorithm for Shaping Bursty Nature of Internet Traffic," in *Telecommunications, 2007. AICT 2007. The Third Advanced International Conference on*, Morne, 2007.
- [35] M. S. Islam, "Traffic Analysis of Wireless IP Networks," *Journal of Telecommunications*, vol. 5, no. 1, October 2010.
- [36] K. A. Jones, "QoE Informed Configuration of QoS Mechanisms," London, England, 2014.
- [37] Cisco, "The Internet Protocol Journal," *A Quarterly Technical Publication for Internet and Intranet Professionals*, vol. 13, no. 4, 2010.
- [38] S. U. John Schormans, "ECS724P - Network Modeling and Performance - 2013/14," QMUL, 2014. [Online]. Available: <http://qmplus.qmul.ac.uk/course/view.php?id=2414>. [Accessed 16 November 2014].

- [39] K. V. D. W. R. K. H. B. Michel Mandjes, "End-to-end delay models for interactive services on a large-scale IP network," in *7th IFIP workshop*, Belgium, 1999.
- [40] J. H. Wei Zhang, "Modeling End-to-End Delay Using Pareto Distribution," in *Second International Conference on Internet Monitoring and Protection (ICIMP 2007)*, Beijing, 2007.
- [41] S. A. M. M. Kouhei Fujimoto, "Statistical Analysis of Packet delays in the Internet and Its Application to Playout Control for Streaming Applications," *Special Issue on New Developments on QoS Technologies for Information Networks*, 6 June 2001.
- [42] J. H. Wei Zhang, "Modeling End-to-End Delay Using Pareto Distribution," in *Second International Conference on Internet Monitoring and Protection (ICIMP 2007)*, Silicon Valley, 2007.
- [43] P. L. Xiang-Li Zhang, "A new delay jitter smoothing algorithm based on Pareto distribution in Cyber-Physical Systems," *Wireless Netw*, pp. 1913-1923, 21 January 2015.
- [44] Cisco Systems, "Cisco Networking Academy Program: IP Telephony v1.0," 2005. [Online]. [Accessed 20 July 2016].
- [45] L. Z. K. S. N. Liren Zhang, "Effects of delay and delay jitter on voice/video over IP," *Computer Communications*, vol. 25, pp. 863-873, 2002.
- [46] A. P. S. G. Dipendra J. Mandal, "Effect of Packet Drop and Jitter on Perceived Video Quality for Various Encoded Video over Streaming Network," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 4, no. 8, pp. 07-11, 2016.
- [47] G. Liang, "Effect of Delay and Buffering on Jitter-Free Streaming over Random VBR Channels," in *International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine)*, Waterloo, Canada, 2006.
- [48] J. T. Mark Claypool, "The Effects of Jitter on the Perceptual Quality of Video," in *ACM Multimedia*, Orlando, 1999.
- [49] C. B. J.-M. G. Olivier Brun, "A Simple Formula for End-to-End Jitter Estimation in Packet-Switching Networks," in *International Conference on Systems and International Conference on Mobile Communications and Learning Technologies*, Toulouse, 2006.
- [50] A. G. B. S. Hamza Dahmouni, "An analytical model for jitter in IP networks," *Annals of telecommunications*, vol. 67, no. 1, pp. 81-90, 2011.
- [51] F. A. T. Mansour J. Karam, "Analysis of the Delay and Jitter of Voice Traffic over the internet," San Mateo.

- [52] F. R. J. S. M. V. Jan Rozhon, "Modelling the impact of network features on speech quality in voice over IP," *Telecommunications and Signal Processing (TSP), 2015 38th International Conference*, pp. 280-284, 9-11 July 2015.
- [53] M. V. A. K. ., M. H. Pavol Partila, "Jitter Buffer Loss Estimate for Effective Equipment Impairment Factor," *International Journal of Mathematics and Computers in Simulation*, vol. 7, no. 3, 2013.
- [54] M. H. Adrian Kovac, "E-Model MOS Estimate Precision Improvement and Modelling of Jitter Effects," *Information and Communication Technologies and Services*, vol. 10, no. 4, 2012.
- [55] M. Roughan, "A Comparison of Poisson and Uniform Sampling for Active Measurements," *IEEE Journal*, vol. 24, no. 12, pp. 2299-2312, 2006.
- [56] J. Hill, "Assessing the Accuracy of Active Probes for Determining Network Delay, Jitter, and Loss," The University of Edinburgh, Edinburgh, 2002.
- [57] L. Jorgenson, "Where could I find info on active probing technology to monitor network environments?," Techtarget.com, August 2003. [Online]. Available: <http://goo.gl/Lc07p7>. [Accessed 18 February 2015].
- [58] Cisco, "Chapter: Active Directory as a Probe and a Provider," Cisco, [Online]. Available: https://www.cisco.com/c/en/us/td/docs/security/ise/2-2/pic_admin_guide/PIC_admin/PIC_admin_chapter_01011.html. [Accessed 6 November 2014].
- [59] J. S. Maheen Hasib, "Limitations of Passive & Active Measurement Methods in Packet Networks," Queen Mary University of London, London, 2003.
- [60] D. A. Ma, "Queueing Theory," 27 September 2012. [Online]. Available: https://greenpages.eecs.qmul.ac.uk/courseinfo/ele302/documents/Networkplanning-lectureversion_000.pdf. [Accessed 02 November 2014].
- [61] H. Perros, "Queueing Theory - A Primer," [Online]. Available: http://www4.ncsu.edu/~hp/SSME_QueueingTheory.pdf. [Accessed 5 November 2014].
- [62] J. Virtamo, "Queueing Systems," 8 November 2005. [Online]. Available: http://www.netlab.tkk.fi/opetus/s383143/kalvot/E_jonojarj.pdf. [Accessed 11 November 2014].
- [63] G. Noubir, "Introduction to Queueing Theory," 2003. [Online]. Available: <http://www.ccs.neu.edu/home/noubir/Courses/CSG150/F03/lecture3.pdf>. [Accessed 14 November 2014].
- [64] S. B.-H. Chee-Hock Ng, *Queueing Modelling Fundamentals with Applications in Communication Networks*, Chichester, West Sussex, England: John Wiley & Sons Ltd., 2008.

- [65] D. B. I. a. G. D. Parrington, "Delayline: A Wide-Area Network Emulation tool," *The USENIX Association, computing systems*, vol. 7, no. 3, 1994.
- [66] "Scalability and Accuracy in large scale network emulator," in *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, Boston, 2002.
- [67] H.-w. Jin, S. Narravula, K. Vaidyanathan and D. K. Panda, "NemC: A Network Emulator for Cluster-of-Clusters," in *Proceedings of 15th International Conference on Computer Communications and Networks*, Arlington, 2006.
- [68] H. N. S. V. K. a. P. D. Jin, "NemC: A network emulator for cluster-of-clusters," p. 177–182, 2011.
- [69] O. R. Lucas Nussbaum, "A Comparative Study of Network Link Emulators," in *09 Proceedings of the 2009 Spring Simulation Multiconference*, San Diego, 2009.
- [70] LinuxFoundation, "Netem," Linux Foundation, 19 July 2016. [Online]. Available: <http://www.linuxfoundation.org/collaborate/workgroups/networking/netem>. [Accessed 19 July 2016].
- [71] S. Hemminger, "Network Emulation with NetEm," 2005.
- [72] J.-P. L. M. H. A. I. W. Audrius Jurgelionis, "An Empirical Study of NetEm Network Emulation Functionalities," in *Computer Communications and Networks (ICCCN)*, 2011.
- [73] Y. F. G. H. James Martin, "On the Efficacy of the Dynamic Adaptive Streaming Over HTTP (DASH) Protocol".
- [74] A. G. Moe, "Implementing Rate Control with Netem," University of Oslo Department of Informatics, Oslo, 2013.
- [75] D. B. A. K. A. D. K. C. A. E. Natalie Larson, "Investigating Excessive Delays in Mobile Broadband Networks," in *AllThingsCellular*, London, 2015.
- [76] Z. O. L. X. B. R. Jie Feng, "Packet Reordering in High-Speed Networks and Its Impact on High-Speed TCP Variants," Computer Science and Engineering University of Nebraska–Lincoln, Lincoln, NE.
- [77] I. Brugman, "The occurrence of TCP Packet Reordering," in *6th Twente Student Conference on IT*, Enschede, 2007.
- [78] J. Lawrence, "Application Performance Monitoring: Out of Order Packets," Plixer.com, 28 November 2012. [Online]. Available: <https://www.plixer.com/blog/application-performance-management-2/application-performance-monitoring-out-of-order-packets/>. [Accessed 27 September 2017].

- [79] J. V. J. R. Ugo Mocci, "Performance Evaluation and Design of Broadband Multiservice Networks," in *Final Report of Action COST 242*, New York, pp. 159-170.
- [80] D. M. M. M. S. D. Anick, "Stochastic Theory of a Data-Handling System with Multiple Sources," *Bell Labs Technical Journal*, vol. 61, no. 8, pp. 1871-1894.
- [81] P. A. G. Marta Bassi, "Academic Self-Efficacy Beliefs and Quality of Experience in Learning," *Journal of Youth and Adolescence*, vol. 36, no. 3, p. 301–312 , 2007.
- [82] OfCom, "Understanding Satellite Broadband Quality of Experience – Final Report," July 2011. [Online]. Available: https://www.ofcom.org.uk/__data/assets/pdf_file/0015/63312/understanding_satellite.pdf. [Accessed 28 September 2017].
- [83] OVUM, "Global Broadband Experience Scorecards," 2015. [Online]. Available: http://bes.ovum.com/downloads/Global_Broadband_Experience_Scorecards.pdf. [Accessed 2 October 2017].
- [84] T. M. L. S.-K. T. H. Mart'in Varela, "Towards an Understanding of Visual Appeal in Website Design," in *Proceedings of QoMEX 2013*, Austria, 2013.
- [85] B. A. A. M. M. Sajid Mushtaq, "QoE: User Profile Analysis for Multimedia Services," in *2014 IEEE International Conference on Communications*, Venice, 2014.
- [86] V. C. Arunkumar Rajasekaran, "Masters Thesis Electrical Engineering," August 2013. [Online]. Available: <http://www.diva-portal.org/smash/get/diva2:830726/FULLTEXT01.pdf>. [Accessed 16 June 2017].
- [87] UCL, "Children and adults see the world differently," ucl.ac.uk, 14 September 2010. [Online]. Available: <https://www.ucl.ac.uk/news/news-articles/1009/10091401>. [Accessed 07 February 2017].
- [88] Six Revisions, "Designing for Different Age Groups," [Webpagefx.com](http://webpagefx.com), 3 May 2011. [Online]. Available: <https://www.webpagefx.com/blog/web-design/designing-for-different-age-groups/>. [Accessed 13 February 2017].
- [89] O. Cambell, "Designing For The Elderly: Ways Older People Use Digital Technology Differently," smashingmagazine.com, 5 February 2015. [Online]. Available: <https://www.smashingmagazine.com/2015/02/designing-digital-technology-for-the-elderly/>. [Accessed 11 February 2017].
- [90] E. L. Glisky, "Chapter 1: Changes in Cognitive Function in Human Aging," 2007. [Online]. Available: <https://www.ncbi.nlm.nih.gov/books/NBK3885/>. [Accessed 11 February 2017].
- [91] J. Junaid, "Network-Based Monitoring of Quality of Experience," Blekinge Institute of Technology, Blekinge, 2015.

- [92] Dave, "Understanding Video Bit Rates and Why They Matter," Web Video University, 16 March 2011. [Online]. Available: <http://webvideouniversity.com/blog/2011/03/16/understanding-video-bit-rates-and-why-they-matter/>. [Accessed 10 August 2016].
- [93] Edchelp, "Understanding bitrates in video files," encoding.com, 2013. [Online]. Available: <http://help.encoding.com/knowledge-base/article/understanding-bitrates-in-video-files/>. [Accessed 9 August 2016].
- [94] R. Taylor, "A beginners guide to frame rates," AFrame, 18 July 2013. [Online]. Available: <http://aframe.com/blog/2013/07/a-beginners-guide-to-frame-rates/>. [Accessed 10 August 2016].
- [95] W. Bros., "See it in HFR 3D," Warner Bros., 2012. [Online]. Available: <http://www.thehobbit.com/hfr3d/faq.html>. [Accessed 10 August 2016].
- [96] M. College, "Aspect Ratios," Media College, [Online]. Available: <http://www.mediacollege.com/video/aspect-ratio/>. [Accessed 8 August 2016].
- [97] C. Deners, "What is the Aspect Ratio?," Rtngs, 27 January 2014. [Online]. Available: <http://uk.rtngs.com/tv/learn/what-is-the-aspect-ratio-4-3-16-9-21-9>. [Accessed 8 August 2016].
- [98] Red, "Video Aspect Ratios," Red Digital Cinema, [Online]. Available: <http://www.red.com/learn/red-101/video-aspect-ratios>. [Accessed 8 August 2016].
- [99] LG, "21:9 UltraWide Monitors," LG, [Online]. Available: <http://www.lg.com/uk/ultrawide-monitors>. [Accessed 8 August 2016].
- [100] G. Church, "Top 10 video formats," Imagen Ltd, 17 September 2015. [Online]. Available: <http://imagenevp.com/top-10-video-formats/>. [Accessed 11 August 2016].
- [101] K. Cassidy, "Video formats explained," Videomaker, 03 January 2012. [Online]. Available: <https://www.videomaker.com/article/c10/15362-video-formats-explained>. [Accessed 11 August 2016].
- [102] Edchelp, "Video Codec," Encoding.com, 2013. [Online]. Available: <http://help.encoding.com/knowledge-base/article/video-codec/>. [Accessed 11 August 2016].
- [103] C. Peters, "The current state of codecs," Videomaker.com, 7 November 2012. [Online]. Available: <https://www.videomaker.com/article/c3/15374-the-current-state-of-codecs>. [Accessed 11 August 2016].
- [104] AVC, "Video Codec," AVC (Any Video Converter), [Online]. Available: <http://www.any-video-converter.com/mac-tutorial/video-codec.php>. [Accessed 11 August 2016].

- [105] D. Hayek, "The basics of image resolution," Vimeo, 5 April 2012. [Online]. Available: <https://vimeo.com/blog/post/the-basics-of-image-resolution>. [Accessed 11 August 2016].
- [106] L. Colman, "Understanding video resolution," Animoto.com, 4 April 2014. [Online]. Available: <https://animoto.com/blog/news/hd-video-creation-sharing/>. [Accessed 11 August 2016].
- [107] L. Films, Director, *4K Video Beauty of Nature*. [Film]. LoungeV Films, 2015.
- [108] VideoLan, "VideoLAN, a project and a non-profit organisation," VideoLan.org, [Online]. Available: <http://www.videolan.org/>. [Accessed 15 June 2016].
- [109] Videolan, "Documentation: Streaming HowTo New," Videolan.org, 10 November 2014. [Online]. Available: https://wiki.videolan.org/Documentation:Streaming_HowTo_New/#Streaming_using_the_GUI. [Accessed 17 January 2016].
- [110] H. Arora, "TCP/IP Protocol Fundamentals Explained with a Diagram," Thegeekstuff.com, 2 November 2011. [Online]. Available: [http://www.thegeekstuff.com/2011/11/tcp-ip-fundamentals/?utm_source=dlvr.it&utm_medium=twitter&utm_campaign=Feed:%20TheGeekStuff%20\(The%20Geek%20Stuff\)](http://www.thegeekstuff.com/2011/11/tcp-ip-fundamentals/?utm_source=dlvr.it&utm_medium=twitter&utm_campaign=Feed:%20TheGeekStuff%20(The%20Geek%20Stuff)). [Accessed 16 August 2016].
- [111] Taltech, "A brief overview of TCP/IP communications," Taltech, [Online]. Available: http://www.taltech.com/datacollection/articles/a_brief_overview_of_tcp_ip_communications. [Accessed 16 August 2016].
- [112] J. E. Beasley, "Queuing theory," Brunel University, [Online]. Available: <http://people.brunel.ac.uk/~mastjib/jeb/or/queue.html>. [Accessed 16 August 2016].
- [113] SevOne.com, "A Guide to Ensuring Perfect VoIP Calls," [Online]. Available: <http://www.networkmanagementsoftware.com/wp-content/uploads/A%20Guide%20to%20Ensuring%20Perfect%20VoIP%20Calls.pdf>. [Accessed 27 September 2017].
- [114] M. M. S. S. Bastian Huntgeburth, "Open-Source Based Prototype for Quality of Service (QoS) Monitoring and Quality of Experience (QoE) Estimation in Telecommunication Environments," Deutsche Telekom AG, Leipzig, Germany.
- [115] SolarWinds Worldwide, "Fundamentals of VoIP Call Quality Monitoring and Troubleshooting," 2014. [Online]. Available: http://cdn.swcdn.net/creative/pdf/Whitepapers/Fundamentals_of_VoIP_Call_Quality_Monitoring_%20Troubleshooting.pdf. [Accessed 27 September 2017].
- [116] StackOverflow, "How to emulate jitter WITHOUT packet reordering, using TC and NETEM?," Stack Overflow, 3 May 2016. [Online]. Available: <http://stackoverflow.com/questions/36994367/how-to-emulate-jitter-without-packet-reordering-using-tc-and-netem>. [Accessed 11 August 2016].

- [117] Ubuntu, "Netem pfifo not working," Ubuntu, 10 March 2013. [Online]. Available: <https://answers.launchpad.net/ubuntu/+question/223860>. [Accessed 11 August 2016].
- [118] L. Foundation, "Netem," Linux Foundation, [Online]. Available: <https://wiki.linuxfoundation.org/networking/netem>. [Accessed 11 August 2016].
- [119] B. Hubert, "Classful Queueing Disciplines," lartc.org, [Online]. Available: <http://lartc.org/howto/lartc.qdisc.classful.html>. [Accessed 25 February 2016].
- [120] Q. W. C. G. S. G. James Nightingale, "Modeling QoE for streamed H.265/HEVC content under adverse network conditions," in *ICWMMN*, Paisley, San Diego, 2013.
- [121] Q. W. C. G. S. G. James Nightingale, "Subjective Evaluation of the Effects of Packet Loss on HEVC Encoded Video Streams," in *IEEE Third International Conference on Consumer Electronics - (ICCE-Berlin)*, Berlin, 2013.
- [122] A. F. K. M. A. A.-T. Ayman N. Murshed, "Quality of Experience Analysis of Real-Time Video Streaming over Lossy Networks," in *IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, London, Jordan, 2013.
- [123] L. Barman, "PoR2 – Proofs of Retrievability and Redundancy," 2015.
- [124] Videolan.org, "TCP," VideoLAN Organization, 12 January 2010. [Online]. Available: <https://wiki.videolan.org/TCP/>. [Accessed 18 June 2017].
- [125] Ubuntu, "tcp - TCP protocol," Ubutnu, 2010. [Online]. Available: <http://manpages.ubuntu.com/manpages/trusty/man7/tcp.7.html>. [Accessed 13 November 2016].
- [126] Ubuntu, "udp - User Datagram Protocol for IPv4," Ubuntu, 2010. [Online]. Available: <http://manpages.ubuntu.com/manpages/trusty/man7/udp.7.html>. [Accessed 13 November 2016].
- [127] K. D. V. M. F. D. E. Selim Ickin, "The Effects of Packet Delay Variation on the Perceptual Quality of Video," in *4th IEEE Workshop On User MObility and VEhicular Networks*, Denver, 2010.
- [128] NTT Communications, "Enterprise Cloud Service Level Agreement," June 2016. [Online]. Available: http://www.us.ntt.com/content/dam/nttcom/us/pdf/sla/NTTA_Enterprise_Cloud_Service_Level_Agreement.pdf. [Accessed 10 December 2016].
- [129] J. Gardner, "Millennial marketers see video metrics differently," 31 May 2017. [Online]. Available: <http://www.thedrum.com/industryinsights/2017/05/31/millennial-marketers-see-video-metrics-differently>. [Accessed 18 June 2017].

- [130] B. Bower, "Adults Fooled by Visual Illusion, But Not Kids," 11 November 2009. [Online]. Available: <https://www.wired.com/2009/11/optical-illusion-doesnt-fool-kids/>. [Accessed 15 March 2017].
- [131] K. Wallace, "Teens spend a 'mind-boggling' 9 hours a day using media, report says," CNN, 04 November 2015. [Online]. Available: <http://edition.cnn.com/2015/11/03/health/teens-tweens-media-screen-use-report/>. [Accessed 09 August 2016].
- [132] E. Anderson, "Teenagers spend 27 hours a week online: how internet use has ballooned in the last decade," Telegraph, 11 May 2015. [Online]. Available: <http://www.telegraph.co.uk/finance/newsbysector/mediatechnologyandtelecoms/digital-media/11597743/Teenagers-spend-27-hours-a-week-online-how-internet-use-has-ballooned-in-the-last-decade.html>. [Accessed 09 August 2016].
- [133] Mathworks, "Evaluating Goodness of Fit," Mathworks.com, [Online]. Available: https://uk.mathworks.com/help/curvefit/evaluating-goodness-of-fit.html?s_tid=gn_loc_drop. [Accessed 11 August 2015].
- [134] L. S.-K. I. H. Mirko Suznjevic, "Statistical User Behavior Detection and QoE Evaluation for Think Client Services," *Computer Science and Information Systems*, vol. 12, no. 2, p. 587–605, 2014.
- [135] X. Z. Q. Z. W. X. L. H. Kan Zheng, "Quality of Experience Assessment and its Application to Video Services in LTE Networks," in *IEEE Wireless Communications*, United Kingdom, 2015.
- [136] The University of British Columbia, "Demographics - Age," 2016. [Online]. Available: <https://www.grad.ubc.ca/about-us/graduate-education-analysis-research/demographics-age>. [Accessed 10 February 2017].
- [137] UCAS, "Mature undergraduate students," [Online]. Available: <https://www.ucas.com/ucas/undergraduate/getting-started/mature-undergraduate-students>. [Accessed 25 September 2017].
- [138] Southeastern Louisiana University, "Undergraduate & Graduate Students by Age and Gender," 17 May 2017. [Online]. Available: <http://www2.southeastern.edu/Administration/Inst-Research/Student/data.cgi?stuage.txt>. [Accessed 27 September 2017].
- [139] Cogent Communications, "Network Services: Service Level Agreement Global," September 2016. [Online]. Available: https://www.cogentco.com/files/docs/network/performance/global_sla.pdf. [Accessed 10 September 2017].
- [140] A. B. a. P. K. Jaroslaw Sliwinski, "EmPath: Tool to Emulate Packet Transfer Characteristics in IP Network," in *Traffic Monitoring and Analysis, Second International Workshop*, Zurich, 2010.

- [141] Google, "System requirements: Movies, TV shows and live events requirements," YouTube, [Online]. Available: <https://support.google.com/youtube/answer/78358?hl=en-GB>. [Accessed 02 12 2016].
- [142] K. Mikoluk, "4 Problems with YouTube and How to Solve Them," UdeMy, 21 April 2014. [Online]. Available: <https://blog.udemy.com/problems-with-youtube/>. [Accessed 2 December 2016].
- [143] T. Middleton, "YouTube Help Forum: How do I fix youtube speed-up lag?," Google/YouTube, 13 March 2015. [Online]. Available: <https://productforums.google.com/forum/#!topic/youtube/wn3BmcFZ2pc>. [Accessed 3 December 2016].
- [144] Shyam, "Improve YouTube Buffering, Performance & Speed on Windows PC," The Windows Club, 5 November 2013. [Online]. Available: <http://www.thewindowsclub.com/improve-youtube-buffering-performance-speed>. [Accessed 3 December 2016].
- [145] R. S. E. B. a. L. P. Tobias Hoßfeld, "Internet Video Delivery in YouTube: From Traffic Measurements to Quality of Experience," *Data Traffic Monitoring and Analysis*, pp. 266-303, 2013.
- [146] S. J. Z.-L. Z. Vijay Kumar Adhikari, "Where Do You Tube, Uncovering YouTube Server Selection Strategy".
- [147] A. F. J. R. K. M. M. M. M. a. S. R. Ruben Torres, "Dissecting Video Server Selection Strategies in the YouTube CDN".
- [148] Google, "Google Data Centres: Data center locations," Google, [Online]. Available: <https://www.google.com/about/datacenters/inside/locations/index.html>. [Accessed 08 December 2016].
- [149] R. Miller, "Google Data Center FAQ," Data Center Knowledge, 15 May 2012. [Online]. Available: <http://www.datacenterknowledge.com/archives/2012/05/15/google-data-center-faq/>. [Accessed 15 December 2016].
- [150] S. J. Y. C. a. Z.-L. Z. Vijay Kumar Adhikari, "How Do You "Tube"?".
- [151] Google, "Video Quality," Google/YouTube, [Online]. Available: <https://support.google.com/youtube/answer/91449?hl=en-GB>. [Accessed 12 December 2016].
- [152] B. Derin, "Add-ons: YouTube High Definition 50.2," Mozilla, [Online]. Available: <https://addons.mozilla.org/en-GB/firefox/addon/youtube-high-definition/>. [Accessed 11 Decemeber 2016].

- [153] Google, "Supported YouTube file formats," Google, [Online]. Available: <https://support.google.com/youtube/troubleshooter/2888402?hl=en>. [Accessed 10 February 2017].
- [154] Google, "Recommended upload encoding settings," Google, [Online]. Available: <https://support.google.com/youtube/answer/1722171?hl=en>. [Accessed 10 February 2017].
- [155] StackExchange, "How does YouTube encode my uploads and what codec should I use to upload?," StackExchange, [Online]. Available: <https://video.stackexchange.com/questions/5318/how-does-youtube-encode-my-uploads-and-what-codec-should-i-use-to-upload/20895>. [Accessed 10 February 2017].
- [156] M. S. R. S. Tobias Hoßfeld, "Quantification of YouTube QoE via Crowdsourcing," in *2011 IEEE International Symposium on Multimedia*, Washington, 2011.
- [157] A. A. D. J. A. G. J. Z. V. S. I. S. H. Z. Florin Dobrian, "Understanding the Impact of Video Quality of User Engagement," in *SIGCOMM'11*, Toronto, 2011.
- [158] P. C. N. V. K. G. V. W. W. V. d. B. J. D. C. F. D. T. Nicolas Staelens, "On the impact of video stalling and video quality in the case of camera switching during adaptive streaming of sports content," Ghent University, Brussels, 2015.
- [159] D. Deeth, "Global Internet Phenomena Report: Africa, Middle East & North America," Sandvine Intelligent Broadband Networks, 7 December 2015. [Online]. Available: <http://www.internetphenomena.com/2015/12/global-internet-phenomena-report-africa-middle-east-north-america/>. [Accessed 22 November 2016].
- [160] J. E. Solsman, "Netflix and Other Streaming Services Hog 70 Percent of the Internet," Yahoo, 7 December 2015. [Online]. Available: <https://www.yahoo.com/entertainment/netflix-other-streaming-services-hog-70-percent-internet-145234018.html>. [Accessed 22 November 2016].
- [161] Statista, "Leading multimedia websites in the United States in November 2016, based on market share of visits," Statista, November 2016. [Online]. Available: <https://www.statista.com/statistics/266201/us-market-share-of-leading-internet-video-portals/>. [Accessed 26 October 2017].
- [162] New Zealand Telecommunications Forum, "IP Interconnection for Voice," 14 September 2012. [Online]. Available: <http://www.tcf.org.nz/industry/standards-compliance/infrastructure-connections/ip-interconnection/ip-interconnection-for-voice-technical-standards.pdf>. [Accessed 28 September 2017].
- [163] NTT Europe, "NTT Europe LTD Global IP Network Backbone Service Level Agreement ("SLA")," [Online]. Available: http://www.eu.ntt.com/content/dam/nttcom/eu/pdf/services/network/ip-network-transit/sla-of-global-ip-network/NTTE_GIN_SLA_250210.pdf. [Accessed 28 September 2017].

- [164] P. K. P. L. Z. S. K. T. Alisa Devlic, "QoE-aware optimization for video delivery and storage," in *International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Boston, 2015.
- [165] M. M. I. Bisio, "Analytical Expression and Performance Evaluation of TCP Packet Loss Probability Over Geostationary Satellite," *IEEE Communications Letters*, vol. 8, no. 4, pp. 232-234, 2004.
- [166] Y. C. C. S. G. S. H. Y. V. J. Neal Cardwell, "BBR: Congestion-Based Congestion Control," *ACMqueue*, vol. 14, no. 5, 2016.
- [167] A. M. Phillipa Sessini, "Observations on Round-Trip Times of TCP Connections," University of Calgary, Calgary.
- [168] I. K. N. M. Guido Appenzeller, "Sizing Router Buffers," [Online]. Available: <https://www.nanog.org/meetings/nanog32/presentations/appenzeller.ppt>. [Accessed 14 July 2017].
- [169] Juniper Networks, "Understanding T1 and E1 Interfaces," 5 November 2013. [Online]. Available: https://www.juniper.net/documentation/en_US/junos12.1x46/topics/concept/interface-security-t1-and-e1-understanding.html. [Accessed 12 July 2017].
- [170] Cisco, "Chapter: Configurable Queue Depth," [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_conmgt/configuration/xs-3s/qos-conmgt-xe-3s-book/qos-conmgt-qdepth.html#GUID-47CFCFBD-9E4D-4754-80DD-50CD6E296CF9. [Accessed 12 July 2017].
- [171] Accenture, "Network transformation: The quest for profitability, quality and all-IP networks," Accenture, [Online]. Available: <https://www.accenture.com/us-en/insight-highlights-cmt-article-network-transformation>. [Accessed 30 October 2015].
- [172] Executive Briefing Service, "Lag Kills! How App Latency Wrecks Customer Experience," STLpartners, November 2015. [Online]. Available: <https://stlpartners.com/research/lag-kills-how-app-latency-wrecks-customer-experience/>. [Accessed 25 February 2016].
- [173] J. Kenyatta, "Client/Server Architecture," Oracle, 1996. [Online]. Available: http://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch20.htm. [Accessed 21 January 2016].
- [174] C. M. Kozierok, "HTTP Operational Model and Client/Server Communication," TCP/IP Guide, 20 September 2005. [Online]. Available: http://www.tcpiptide.com/free/t_HTTPOperationalModelandClientServerCommunication.htm. [Accessed 27 January 2016].

- [175] L. Chung, "Client-Server Architecture," 23 March 2000. [Online]. Available: <https://www.utdallas.edu/~chung/SA/2client.pdf>. [Accessed 21 January 2016].
- [176] D. Scocco, "Example of Client-Server Program in C (Using Sockets and TCP)," Programming Logic, 22 April 2014. [Online]. Available: <http://www.programminglogic.com/example-of-client-server-program-in-c-using-sockets-and-tcp/>. [Accessed 9 February 2016].
- [177] TutorialsPoint, "Unix Socket - Server Examples," Tutorials Point, [Online]. Available: http://www.tutorialspoint.com/unix_sockets/socket_server_example.htm. [Accessed 9 February 2016].
- [178] H. Arora, "C Socket Programming for Linux with a Server and Client Example Code," TheGeekStuff.com, 19 December 2011. [Online]. Available: <http://www.thegeekstuff.com/2011/12/c-socket-programming/>. [Accessed 12 February 2016].
- [179] J. A. S. J M Pitts, Introduction to IP and ATM Design and Performance, Chichester: JOHN WILEY & SONS, LTD, 2000.
- [180] Cisco, "Quality of Service (QoS)," Cisco, [Online]. Available: <http://www.cisco.com/c/en/us/products/ios-nx-os-software/quality-of-service-qos/index.html>. [Accessed 8 December 2014].
- [181] Microsoft, "What is QoS?," Microsoft, 28 March 2003. [Online]. Available: <http://technet.microsoft.com/en-us/library/cc757120%28v=ws.10%29.aspx>. [Accessed 8 December 2014].
- [182] S. Floyd, "Sally Floyd - Information," ICIR, October 2013. [Online]. Available: <http://www.icir.org/floyd/>. [Accessed 9 December 2014].
- [183] Wikipedia, "Van Jacobson," Wikipedia, 8 August 2014. [Online]. Available: http://en.wikipedia.org/wiki/Van_Jacobson. [Accessed 9 Decemeber 2014].
- [184] Cisco, "Congestion Avoidance Overview," Cisco, [Online]. Available: http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfconav.html. [Accessed 9 December 2014].
- [185] R. L. Hill, "Random Early Detection (RED)," 2012. [Online]. Available: <http://www.cs.indiana.edu/classes/p438/triggeringTransmission.pdf>. [Accessed 9 Decemeber 2014].
- [186] C. B. S. F. Thomas Ziegler, "Stability Criteria of RED with TCP Traffic," in *9th IFIP Conference on Performance Modelling and Evaluation of ATM & IP Networks*, Budapest, 2001.
- [187] G. V. M. Shreedhar, "Efficient Fair Queuing Using Deficit Round-Robin," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, 1996.

- [188] Cisco, "Understand and Configure MDRR/WRED on the Cisco 12000 Series Internet Router," [Online]. Available: http://www.cisco.com/image/gif/paws/18841/mdrr_wred_18841.pdf. [Accessed 9 December 2014].
- [189] J. N. Chuck Semeria, "Supporting Differentiated Service Classess: Queue Scheduling Disciplines," 2001. [Online]. Available: <http://users.jyu.fi/~timoh/kurssit/verkot/scheduling.pdf>. [Accessed 19 February 2015].
- [190] M. C. Jae Chung, "Analysis of Active Queue Management," *Network Computing and Applications*, pp. 359 - 366 , 16 - 18 April 2003.
- [191] C. R. C. Q. Seungwan Ryu, "Advances in Active Queue Management (AQM) Based TCP Congestion Control," *Telecommunication Systems*, pp. 317 - 351, 25 March 2004.
- [192] V. J. Kathleen Nichols, "A modern AQM is just one piece of the solution to bufferbloat," *Controlling Queue Delay*, vol. 10, no. 5, 2012.
- [193] Lorema, "CoDel - The Controlled-Delay Active Queue Management algorithm," OpenWrt, 10 August 2013. [Online]. Available: http://wiki.openwrt.org/doc/howto/packet.scheduler/sch_codel. [Accessed 19 February 2015].
- [194] P. N. C. P. M. P. V. S. F. B. B. V. W. Pan, "PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem," pp. 148-155, 2013.
- [195] NCTA, "QoE: As Easy As PIE," National Cable & Telecommunications Association (NCTA), 2013. [Online]. Available: <http://www.nctatechnicalpapers.com/Paper/2013/2013-qoe-as-easy-as-pie>. [Accessed 18 February 2015].
- [196] P. N. C. P. M. P. V. S. F. B. B. S. R Pan, "PIE: A lightweight latency control to address the bufferbloat problem issue," 30 July 2013. [Online]. Available: <http://www.ietf.org/proceedings/87/slides/slides-87-aqm-4.pdf>. [Accessed 18 February 2015].
- [197] T. H. P. T.-G. Markus Fiedler, "A Generic Quantitative Relationship between Quality of Experience and Quality of Service," *IEEE Network*, vol. 24, no. 2, pp. 36-41, 2010.
- [198] G. G. J. L. N. M. G. A. Covington, "A Packet Generator on the NetFPGA Platform," in *Field Proramable Custom Computing Machines*, California, 2009.
- [199] N. M. G. W. G. G. P. H. J. N. R. R. J. L. J. Lockwood, "NetFPGA - An Open Platform for Gigabit-rate Network Switching and Routing," in *Microelectronic Systems Education*, California, 2007.
- [200] M. G. Y. G. M. L. J. S. G. Salmon, "NetFPGA-based Precise Traffic Generation," in *NetFPGA Developers Workshop*, Toronto, 2009.

- [201] S. P, "Ostinato Packet/Traffic Generator and Analyzer," [Online]. Available: <https://code.google.com/p/ostinato/>. [Accessed 8 February 2015].
- [202] H. O. Software, "Seagull: an Open Source Multi-protocol traffic generator," HP OpenCall Software, 26 February 2009. [Online]. Available: <http://gull.sourceforge.net/>. [Accessed 8 February 2015].
- [203] P. D. S. Ltd., "LanTraffic V2 - IP Traffic Generation Software," Packet Data Systems Ltd., 2010. [Online]. Available: http://www.pds-test.co.uk/products/ip_generation.html. [Accessed 8 February 2015].
- [204] NTOP, "Building a 10 Gbit Traffic Generator using PF_RING and Ostinato," NTOP, 18 August 2011. [Online]. Available: http://www.ntop.org/pf_ring/building-a-10-gbit-traffic-generator-using-pf_ring-and-ostinato/. [Accessed 9 February 2015].
- [205] F. Klassen, "Build Your Own 10GigE Wire-Rate NetFlow Traffic Generator Using Tcpreplay 4.0," AppNeta, 12 March 2014. [Online]. Available: <http://www.appneta.com/blog/wire-rate-netflow-traffic-generator/>. [Accessed 9 February 2015].
- [206] G. C. M. L. A. R. M. A. Marson, "Quality of Service in Multiservice IP Networks," in *QoS-IP*, Milano, 2003.
- [207] R. Aber, Cisco, 7 May 2004. [Online]. Available: <http://www.ciscopress.com/articles/article.asp?p=170742&seqNum=4>. [Accessed 18 February 2015].
- [208] Blogspot, "Role of The Router," April 2009. [Online]. Available: <http://ciscolearning.blogspot.co.uk/2009/04/role-of-router.html>. [Accessed 17 February 2015].
- [209] J. Aweya, "IP Router Architectures: An Overview," *Journal of Systems Architecture*, vol. 46, pp. 483-511, 1999.
- [210] Cisco, "Networkers," 1998. [Online]. Available: http://www.cisco.com/networkers/nw99_pres/601.pdf. [Accessed 15 February 2015].
- [211] J. D. Morris, "7.2 Finding a confidence interval for the sample proportion," 08 03 2007. [Online]. Available: <http://www.math.utah.edu/~morris/Courses/1070/notes/l14.pdf>. [Accessed 01 September 2015].
- [212] D. J. Amang Sukasih, "An Application of Confidence Interval Methods for Small Proportions in the Health Care Survey of DoD Beneficiaries," 08 06 2005. [Online]. Available: <http://econpapers.repec.org/paper/mprmprrres/9bc00765bb2645f7925da121d641cb05.htm>. [Accessed 01 September 2015].

- [213] E. Cameron, "On the Estimation of Confidence Intervals for Binomial Population Proportions in Astronomy: The Simplicity and Superiority of the Bayesian Approach," *Publications of the Astronomical Society of Australia*, vol. 28, pp. 128-139, 2011.
- [214] "Confidence Interval Calculator," TutorVista, [Online]. Available: <http://calculator.tutorvista.com/confidence-interval-calculator.html>. [Accessed 02 09 2015].
- [215] J. C. Pezzullo, "Exact Binomial and Poisson Confidence Intervals," StatPages.org, 25 5 2009. [Online]. Available: <http://statpages.org/confint.html>. [Accessed 02 September 2015].
- [216] Over-blog.com, "What Role Does Router Play in a Network," over-blog.com, 14 September 2011. [Online]. Available: <http://ciscorouterswitch.over-blog.com/article-what-role-does-router-play-in-a-network-84236006.html>. [Accessed 16 February 2015].
- [217] J. H. R. R. G. Cole, "Voice over IP Performance Monitoring," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 2, pp. 9-24, 2001.
- [218] L. X. a. J. Schormans, "Planning Simulation Run Length for Packet Networks," QMUL, London, 2013.

Appendix

Appendix A Challenges Faced in Research

Network Router Lab: The Challenges

Experimentation with a testbed that contains real network components such as routers, switches, and traffic generators can be very productive and a beneficial learning experience. However, working within such an environment is prone to failures potentially leading to a slow work progress and even setbacks.

In the subsequent sections, some similar challenges have been outlined that were faced during this research.

Issues with the Remote Access to the Testbed

The networks lab, a small-sized room without a window, that housed the testbed previously used for this research also had various other devices that were operating at the same time and used by other researchers. Considering the size of the lab and lack of ventilation as well as other noise of operational devices including the air-conditioning units, it was not ideal to work being physically present in the lab.

Therefore, for health and safety reasons and to focus on the experimental research itself in a productive and peaceful manner, it was advised to access the testbed remotely over a secure network and carry out the experimentations remotely.

To access the lab facilities remotely, the standard process required was a two-step procedure. The first step in establishing a remote connection to the testbed was connecting to the university network securely. This prevented unauthorised access to the lab from other networks outside of the university.



Figure A.1: A successful connection to the VPN

Having connected to the university VPN, the second step of the procedure was connecting to the testbed server (named RLPortal) to be able to operate the testbed remotely.

As the RLPortal was running Microsoft Windows Operating System, the remote connection was established using Remote Desktop Connection application from the host computer in use by the researcher. The IP address of the RLPortal and the user login credentials (set up beforehand) for the researcher were required to establish a connection with the server.

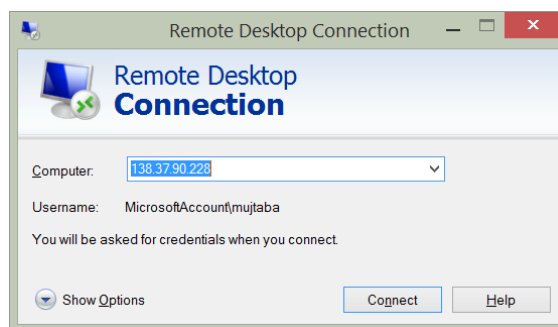


Figure A.2: Establishing a remote connection with the lab server (RLPortal).

Provided that the user was already connected to the VPN, and used their remote login credentials correctly, the remote connection to the RLPortal would be successful allowing the user to carry out experimentation on the testbed.

Technical Difficulties Experienced

In this two-step remote connection procedure, occasional failure was experienced in step 1, the VPN connectivity. Sometimes, the VPN server was down that did not allow the remote connection go past step 1.

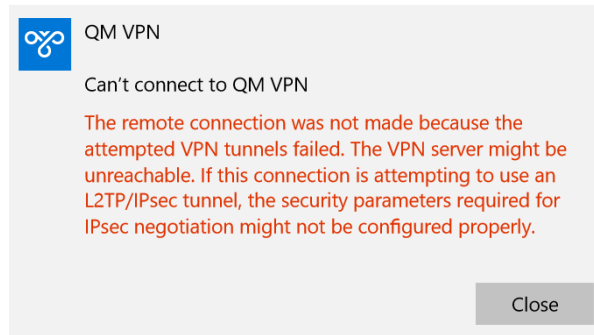


Figure A.3: Old VPN server fails to allow VPN connection.

As this old VPN server was not being monitored or maintained any more, it would take a long time to have it fixed and working again, which led to setbacks a few times even during official demonstrations to potential research partners.

Later the university VPN infrastructure was upgraded to the next generation OpenVPN based VPN servers. These new VPN servers were not only reliable but were maintained and monitored by the technical support for any technical issues.

Networks Lab Overheating

The lab got overheated several times leaving the testbed components to shut down, even with some of them burning out at one point.

NetEm Testbed: The Challenges

Designing the NetEm testbed, as it is in the current optimum state it is now, was not an easy task. Various obstacles were experienced along the way. However, the arising challenges were addressed chronologically.

Testbed Design Model

The initial challenge was to interpret the desired functionality into a model which could further outline the specifications of the design. At this stage, intensive research was carried out to study similar tools used in network research by other researchers. Various options with their pros and cons were considered. Some testbeds previously used consisted of a bigger proportion of software tools running on merely a single computer, i.e. simulations. However, this research was desired to be carried out on a set up that went beyond traditional network simulations. At the same time, one of the design objectives was not to complicate things unnecessarily. For instance, a traditional testbed approach would suggest to have three computers in the new testbed, the server to serve the content, a client to request and accept the content e.g. streaming, and a middleware computer working as a NetEmBox to add the required network degradations. But a simplified design could have only two computers, provided that merging the content server and NetEm on a single computer would not affect the operational performance and accuracy of results.

Having compared technical aspects of both approaches, it was concluded that the desired functionality could be achieved with the two-computer-testbed design as well. This simplified design would reduce the possibility of running into unnecessary issues as in the case of a more complex design, when the testbed

consisted of three computers instead. Hence, following the two-computer-testbed design approach, the third middleware computer was eliminated and the middleware NetEm functionality was incorporated in the server computer.

Server and Client Architecture

After the design model had been laid out, which included two computers in the testbed, the next challenge was to plan the infrastructure for data flow between the two computers. The data flow would be such that the client would request some data and the server would respond to this request. To model this data-driven infrastructure, being able to answer the following questions could simplify the architecture design:

- 1) What protocols would the client computer run?
- 2) To respond to the client computer, what protocols would the server computer run and support to fulfil requests from the client computer?
- 3) Scalability – what additional hardware/software resources would the server need in case more client devices required a connection?
- 4) What sort of traffic flow needed to be facilitated between the server and the client?
- 5) How much bandwidth would the link need to fulfil the client-server communication?
- 6) To decide whether most of the decisions will be made by the client or the server, how much processing control was to be assigned to each computer, e.g. the client and the server?

To be able to answer the above-mentioned questions, extensive research was conducted online and otherwise. Some of the resources that were referred to at this stage to assess the technical requirements were [173] [174] [175]. Then, from the implementation prospective of the client-server infrastructure many other resources were also accessed and [176] [177] [178] are some to mention.

With this continuous exploration, it was determined the streaming capabilities of the VLC media player could be utilised that would significantly streamline the traffic flow infrastructure by providing a out-of-the-box network streaming feature. Hence, to incorporate this functionality in the testbed, the VLC documentation was referred to which then lead to achieving a major milestone of the testbed design, for details on this please see Chapter 4 on page 48.

Replacing TFIFO with PFIFO in NetEm

The default queueing discipline in NetEm is TFIFO. TFIFO, without being given implicit instructions to reorder data packets, reordered the packets during transmission. Consequently, in some experiments conducted in the beginning, the experimental results were not found to match the analytical equivalent results, due to the packets going out of sequence. This was indeed a significant concern as the research could not be progressed further until this unexpected packet ordering in transmission had been controlled.

Having carried out some research, it was concluded that this unplanned packet order in NetEm was due to the default qdisc TFIFO, and this needed to be replaced with a PFIFO. Now, what made this challenge even more thought-provoking was the fact that nobody in network research had previously used the PFIFO in NetEm. Even if someone tried replacing the TFIFO qdisc with PFIFO in NetEm, there was no evidence of them being successful.

This posed a great challenge that was to be overcome. However, after several days reading up and attempting to implement PFIFO, the task was finally accomplished. This was an impressive achievement and a major milestone in testbed design as this was the solution to a well-known problem that NetEm throws at network researchers. Now, with this PFIFO implementation many researchers can take advantage of the powerful network emulation of NetEm in order to mimic real world network traffic, which could help these researchers take necessary design decisions from an ISP or manufacturer's prospective.

Testbed Advanced Interfacing and Connectivity

A whole set of challenges arose further down the line during the advancement of NetEmBox testbed capabilities for accommodating a wider range of experimentation facilities. This would be in addition to experimenting on VOD traffic on a local network. The planned outcomes of the design were to be able to experiment on real Internet traffic such as YouTube and Skype. Furthermore, adding a wireless connectivity feature to the testbed was another objective at this stage.

After conducting extensive research, trialling different approaches and investing many man-hours, eventually, all design outcomes were met successfully, and the posed challenges were resolved very methodologically. This innovative testbed will now be used by other network research students as well. This is a great contribution made to networks research at this university and in the field of networks in a wider view.

Appendix B Initial assessment of research related tools and skillset

The initial planning suggested that the following skills and tools were required to conduct this research competently.

Necessary Skills

A good understanding and operational knowledge of the following was required.

LabVIEW

As the main program used for the design of algorithms was initially LabVIEW, having a good working knowledge of LabVIEW was mandatory to conduct the research.

Internet Routers

As the research focused on the QoS, it was apparent that having a good understanding of the design and operational principles of routers was of vital importance to carry out this research.

CISCO Operating System

As the routers used in the lab environment were provided by Cisco, therefore having an understanding of the operational commands and troubleshooting techniques of these Cisco routers was essential.

Required Tools

- A computer with average specifications;
- LabVIEW software;
- Remote Desktop connectivity capability;
- Virtual Private Network (VPN) capability;
- Access to the QMUL VPN, and;
- Access to the Router testbed Lab.

Further skills and tools were acquired when the research progressed, for details please refer to Section 1.3 on page 19.

Appendix C A simple version of Instructions to operate NetEm with a PFIFO queue

Firstly, to see the default queue setup in NetEm, execute:

```
tc qdisc show dev eth0
```

This will then return the status of default qdisc as shown below:

```
qdisc PFIFO_fast 0: root refcnt 2 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1
```

Although the default settings show PFIFO_fast as the qdisc, when the delay or loss is added from NetEm, the PFIFO functionality disappears on the interface status. This is due to the default behaviour of NetEm. The simplest way of keeping NetEm operating with a PFIFO queue is executing the following command after all instructions for delay, loss, and jitter:

```
tc qdisc add dev eth0 root PFIFO limit 1000
```

The limit is a variable and does not have to be a 1000.

Checking the status of eth0 now will show the qdisc to be a PFIFO queue:

```
qdisc PFIFO 8004: root refcnt 2 limit 1000p
```

Although NetEm now shows the operating queue to be the PFIFO, any delay or packet loss added prior to executing the PFIFO command above will be discarded. To have network metrics (delay, jitter, loss) instructions to stay active at the same time as having the queue operating as PFIFO, one additional step needs to be taken, making use of handles mentioned earlier in Section 4.8 on page 52. As stated earlier, handles allow the use of more than one instructions (referred to as ‘filters’ in traffic routing terms) to be activated simultaneously.

Hence, one can add delay and packet loss on the Ethernet eth0 (the interface in this case), and also instruct NetEm to use PFIFO with the following set of instructions:

```
tc qdisc add dev eth0 root handle 1: NetEm delay 500ms 20ms
tc qdisc add dev eth0 parent 1:1 handle 2: NetEm loss 10%
tc qdisc add dev eth0 parent 2:1 PFIFO limit 1000
```

To confirm the simultaneous activation of all of the above instructions, the interface status can be checked with the following command:

```
tc qdisc show dev eth0
```

The output is shown in Figure C.1.

```
root@NetemBox:~# tc qdisc show dev eth0
qdisc netem 1: root refcnt 2 limit 1000 delay 500.0ms 20.0ms
qdisc netem 2: parent 1:1 limit 1000 loss 10%
qdisc pfifo 8003: parent 2:1 limit 1000p
```

Figure C.1: NetEm interface status – PFIFO queue has been activated along with packet delay and packet loss.

From the interface status shown in the screenshot in Figure C., it is evident that the queue is operating as a PFIFO now with a limit of 1000 packets, and the added delay and loss are also active. Further testing was carried out with ping probing to be absolutely certain that PFIFO queue was operating as intended, and the results verified the PFIFO queue to be operational instead of default queueing discipline TFIFO.

The `tc qdisc add dev eth0 parent 2:1 PFIFO limit 1000` command would require executing every time the network metrics are changed on NetEm. However, the use of bash scripts to automate this will simplify the process significantly.

Appendix D Server-side Settings for VOD Streaming

Server-side Network Settings

Before the VOD streaming can be done, the server needs to be prepared for establishing a connection to the client, which may require a new network connection. The process has been described with the help of screenshots below. Note that these screenshots have been taken on the Ubuntu 14.04 operating system.

1. Open/edit network connections:

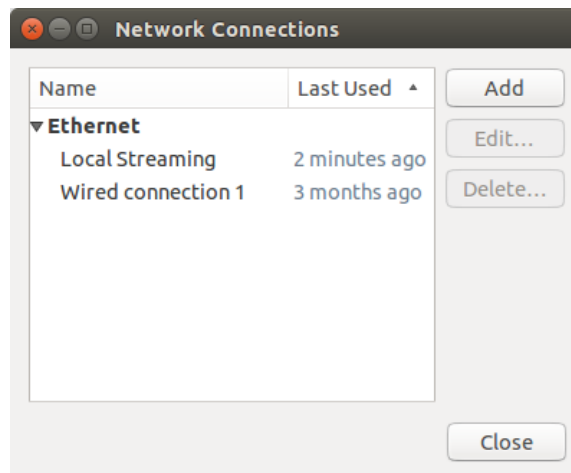


Figure D.1: Network connections graphical user interface

2. Click Add, and choose Ethernet as a connection type then click 'Create':



Figure D.2: Creating a new network connection

3. Give a connection name:

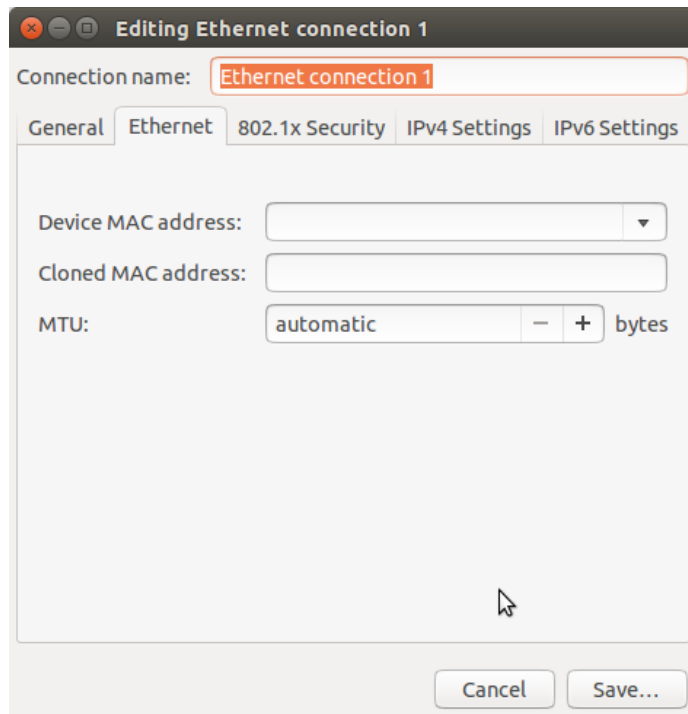


Figure D.3: Naming the new connection

4. Then in Ethernet tab, choose the Ethernet interface, e.g. eth1 in this case:

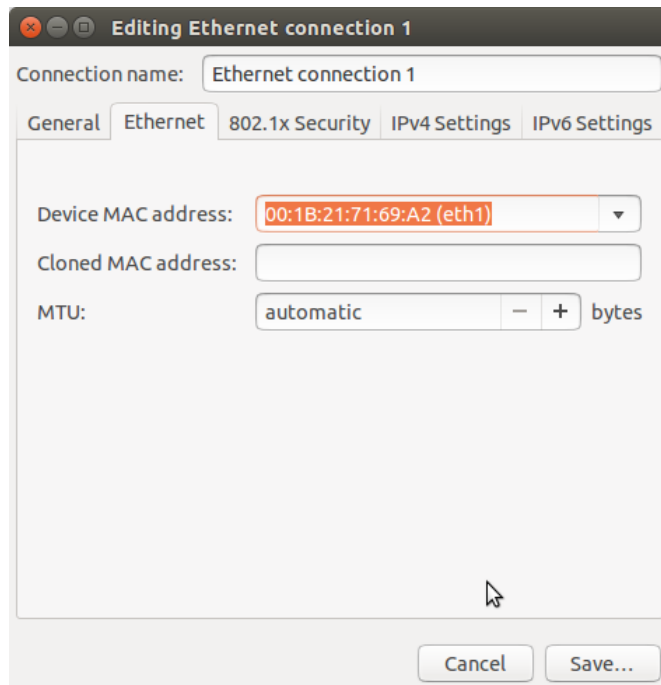


Figure D.4: Assigning an Ethernet interface to a new network connection

5. Then in IPv4 Settings tab, for Method, choose 'Shared to other computers':

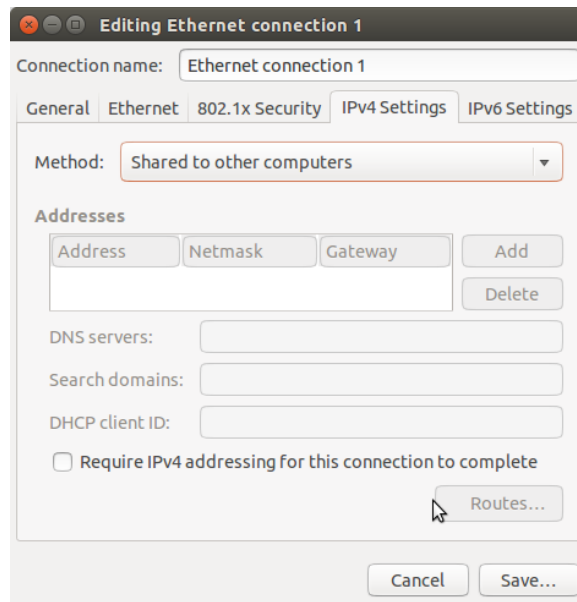


Figure D.5: Choosing the network configuration for this connection

6. Then in General tab, check the box for 'Automatically connect to this network when it is available'. Click 'Save' button:



Figure D.6: Automatic connectivity option

Now this newly created connection will be saved in the list of all available connections.

7. Open the list of connections by clicking the network icon in the top right corner of your desktop¹⁴, and click 'Edit Connections':

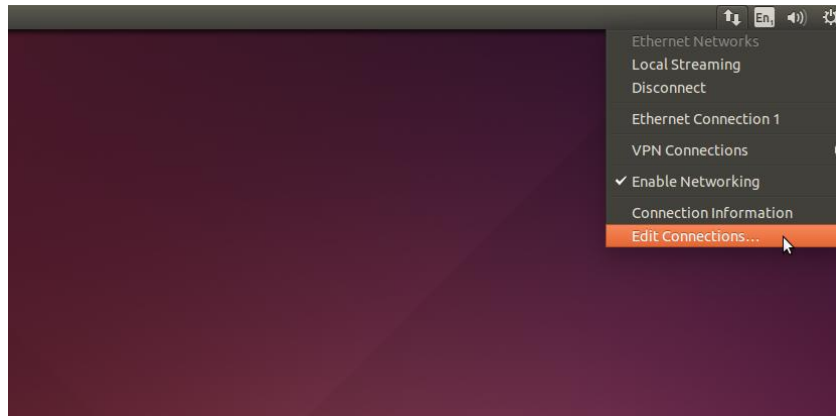


Figure D.7: Accessing all saved network connections

8. The newly created connection 'Internet Connection 1' will be present among all other connections:

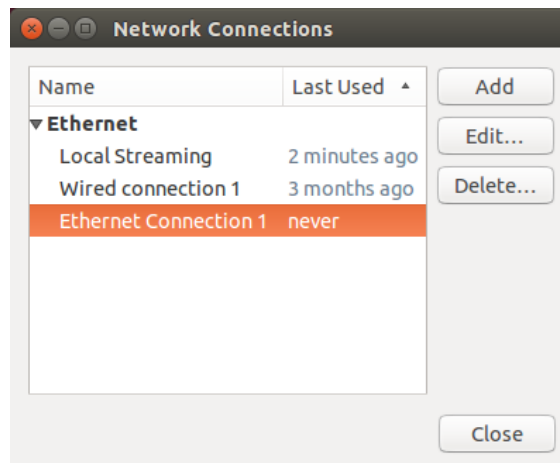


Figure D.8: Newly created connection seen in the connections' list

This confirms that this connection was successfully created.

9. To use this connection, go in the network connections icon (top right corner of Ubuntu Desktop) and click the connection name:

¹⁴ Specific to Ubuntu 14.04 operating system, the visuals may differ on other versions of Linux.

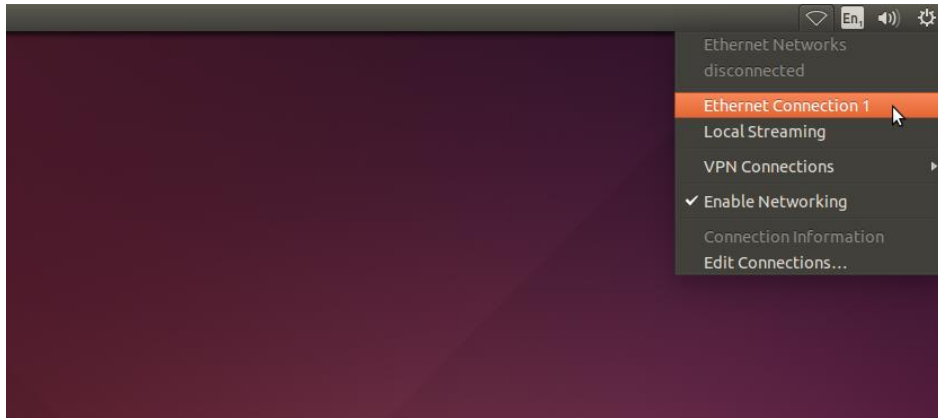


Figure D.9: Connecting via a network connection

10. The network connection to the client will be established, and a confirmation notification will be shown:

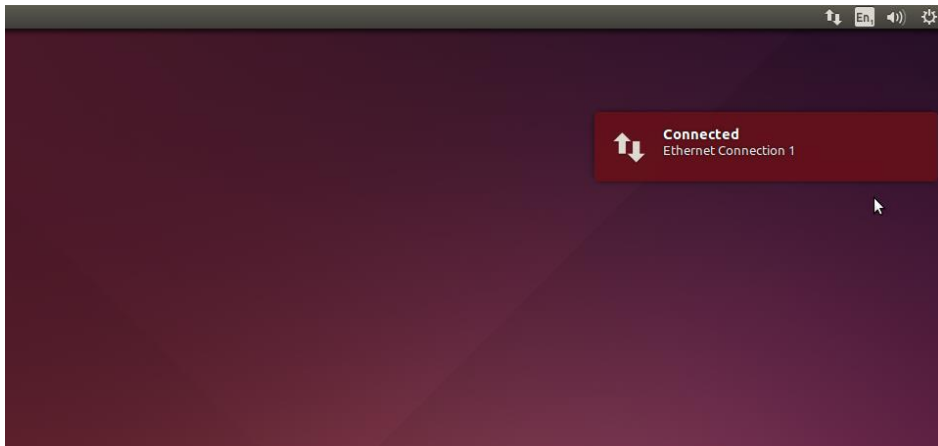


Figure D.10: Connection establishment confirmation

For a successful connection, the client should be ready to establish this connection (a set-up connection on the client-side also required, please see Appendix E on page 201).

Initiating Video Streaming from the Server

Streaming of the VOD has to be started from the server before it can be accessed from the client computers. This process has been described below in easy to follow steps along with screenshots.

1. Open the VLC player:

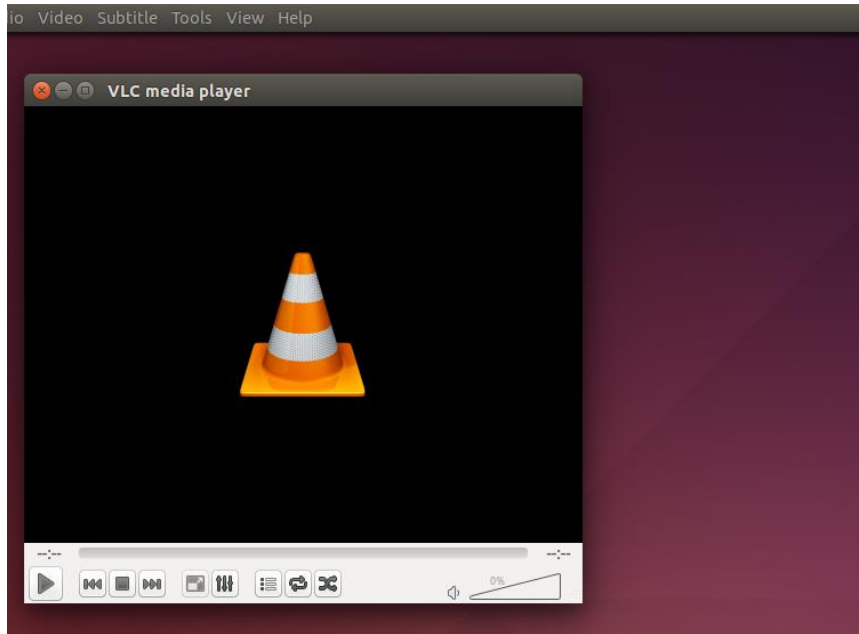


Figure D.11: VLC Player opened on the server computer on the NetEmBox testbed

2. On the top menu bar of VLC in Ubuntu, go in the 'Media' drop-down menu and choose 'Stream':

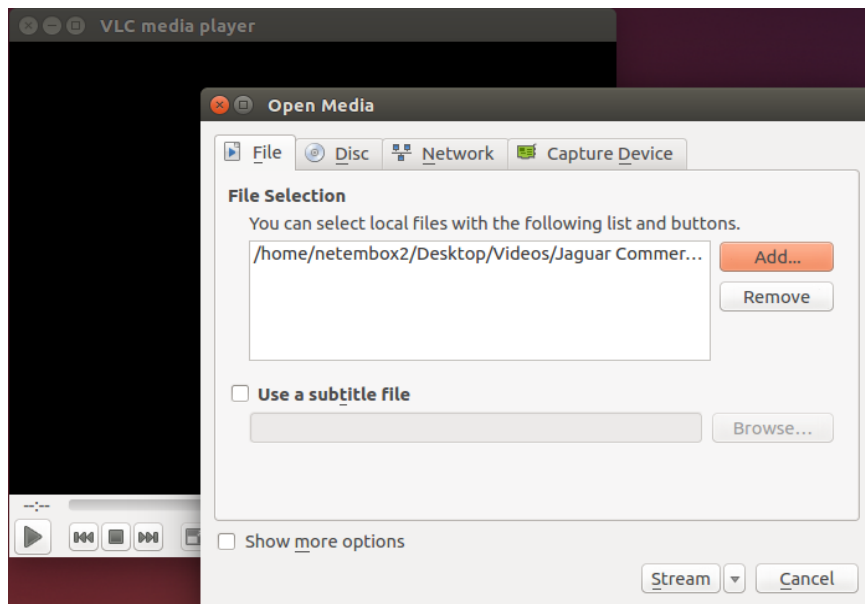


Figure D.12: Streaming option opens 'Open Media' window

In this 'Open Media' window, click 'Add' button and choose a video to stream. Once a video has been added to the list, click the 'Stream' button as shown in the screenshot in Figure D.12.

3. A 'Stream Output' window will then pop up showing the path of the video that is about to be streamed, click Next button:

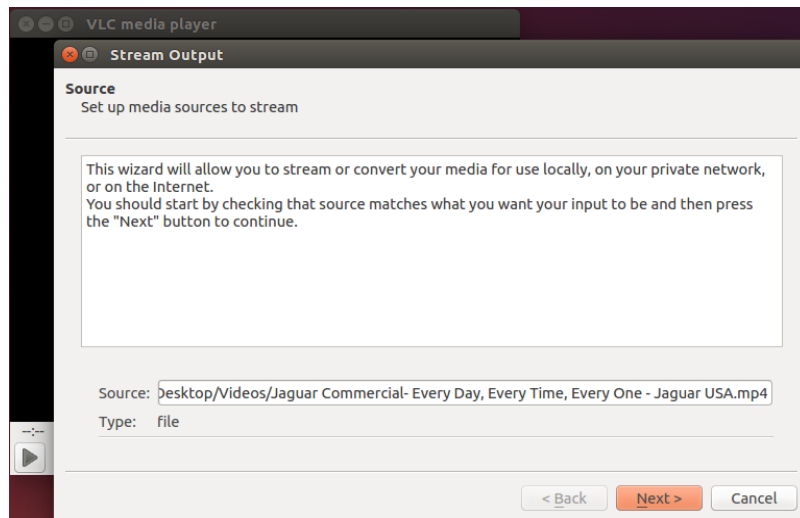


Figure D.13: Streaming output options

4. Now the 'Destination Setup' menu will open up, select your desired streaming protocol from the drop-down menu and click the 'Add' button next to it. In this case, the 'RTP / MPEG Transport Stream' has been selected. Now click Next:

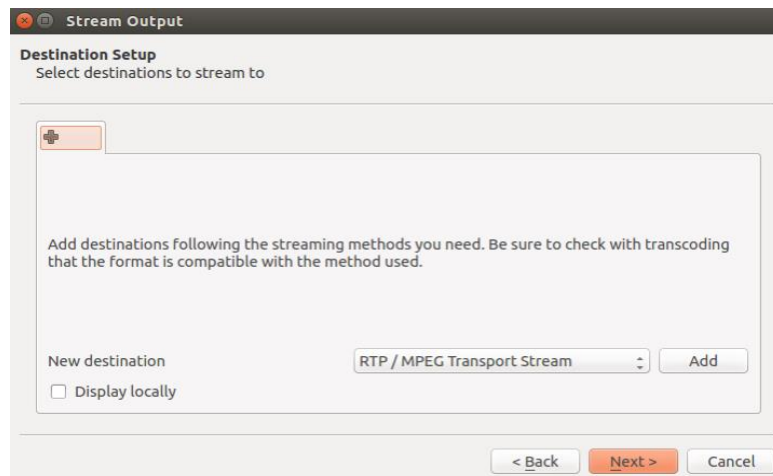


Figure D.14: Choosing the streaming protocol

5. On the next window, the selected streaming protocol will be shown in a tab, along with three input fields requiring the IP address of the device that you are streaming to, a port number (which may already be selected), and a stream name. Once all of these three fields have been filled, click Next.

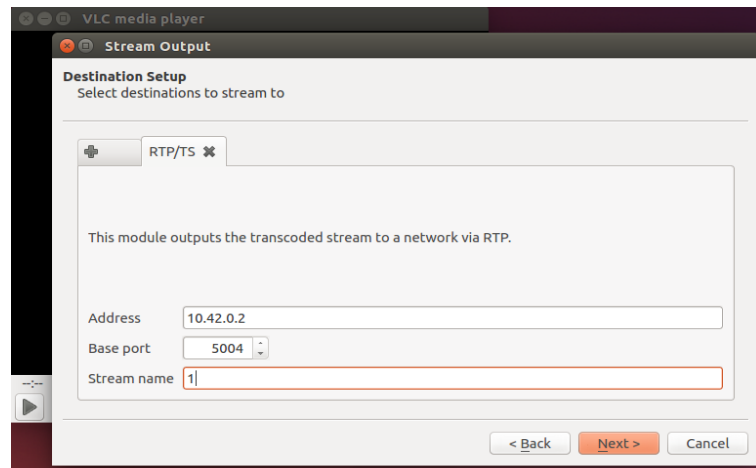


Figure D.15: Providing network details for the stream

6. Transcoding allows the use of a different format for the streaming media. If desired to do so, check the box 'Activate Transcoding' and choose the desired format for transcoding from the drop-down menu for 'Profiles'. The default profile can be used if no strict requirements are to be fulfilled.

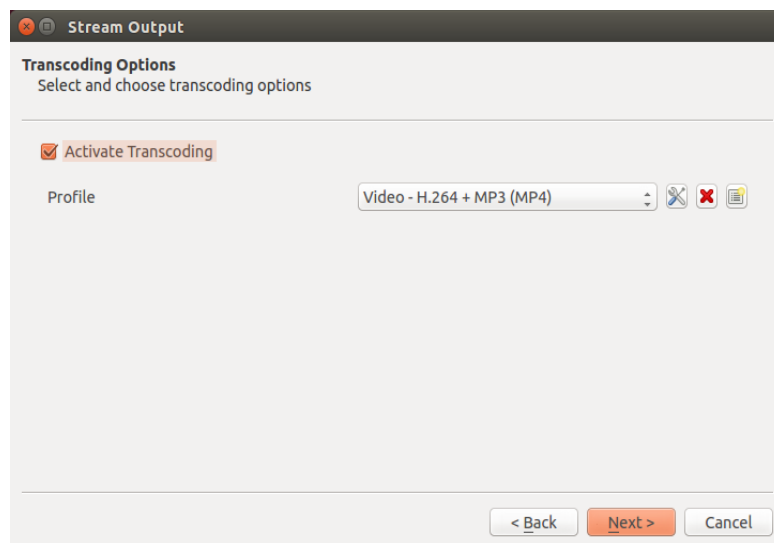


Figure D.16: Activating the transcoding

When ready, click Next.

7. Now check the box for 'Stream all elementary streams', and click the 'Stream' button:

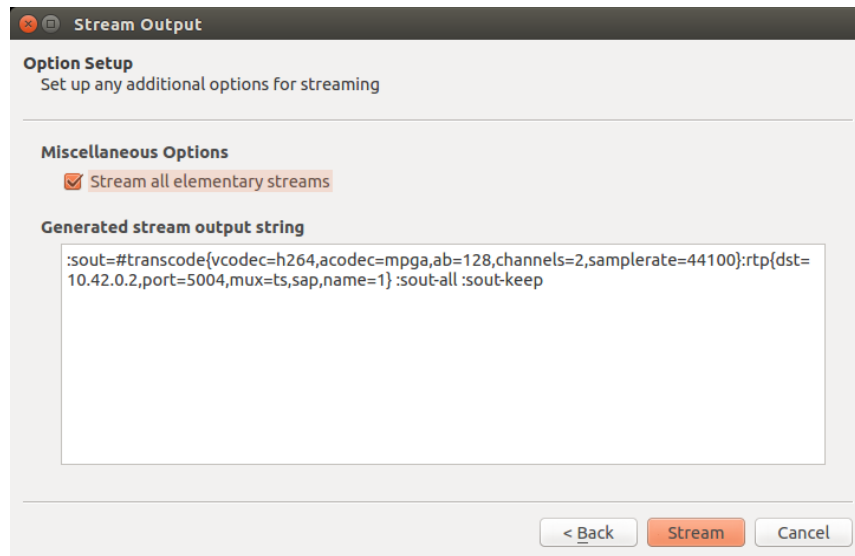


Figure D.17: Streaming all the added video streams

This will now start streaming the video(s), which can be opened from the client device from across the network.

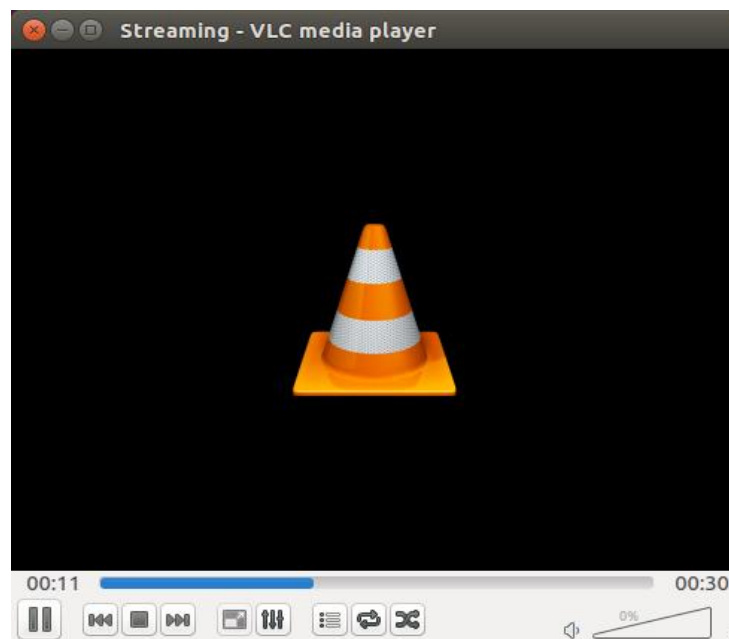


Figure D.18: Streaming starts

For TCP streaming, it may also be desirable to set the Network latency to lowest in VLC preferences. To do that open VLC Preferences, go in the 'Input / Codecs' tab on the left hand, and in Network Section (on the bottom of this opened window) from the 'Default caching policy' drop-down menu choose 'Lowest Latency':

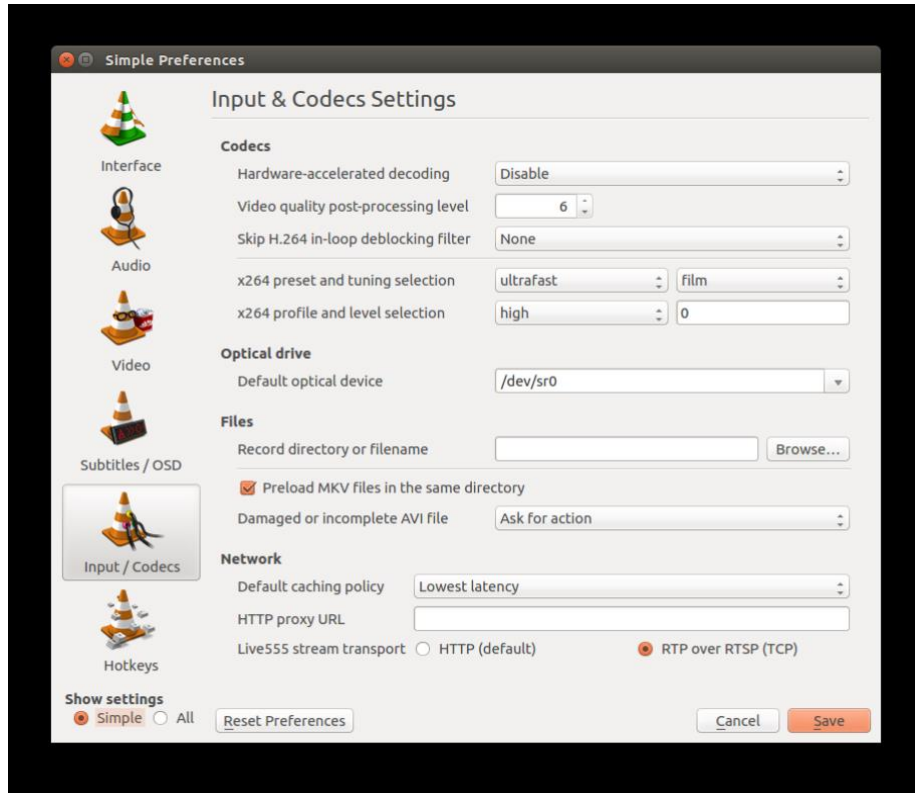


Figure D.19: The option to choose a lowest latency from VLC preferences

Appendix E Client-side Settings for VOD Streaming

Client-side Network settings

To create a new network connection on the client computer for playing-back a network streamed video, the following steps can be followed.

1. Edit network connections and choose the option to create a new connection:

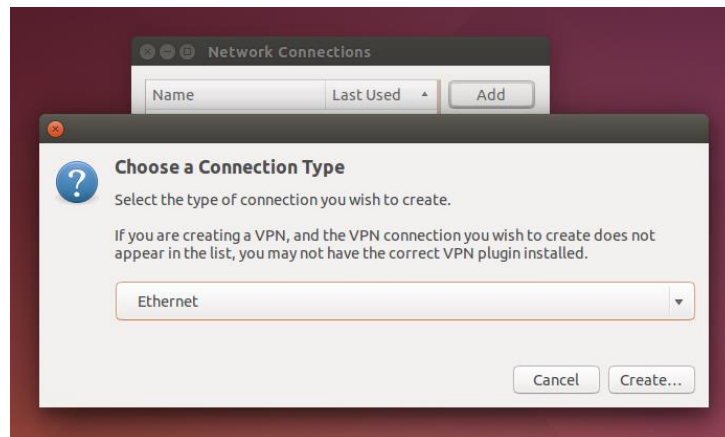


Figure E.1: New connection on the client computer

2. Give a name to the connection. For the testbed in this research, the name of the streaming server was given here, which was NetEmBox:

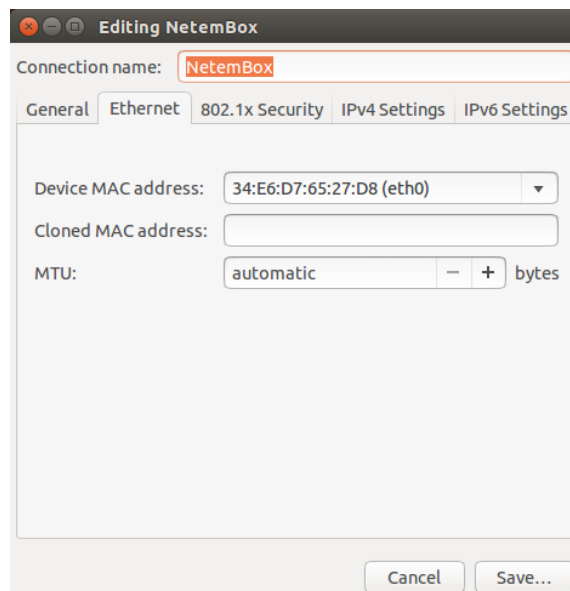


Figure E.2: A meaningful connection name for this connecting to the server

3. Assign an Ethernet interface to this connection, eth0 in this case:

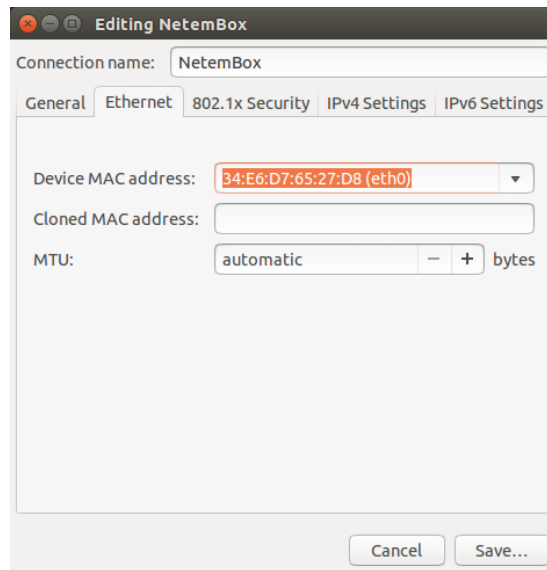


Figure E.3: The Ethernet interface that would be used for this connection

4. Choose the 'Automatic DHCP' in the method drop-down menu; this will allow the server to assign an IP address to the client automatically:

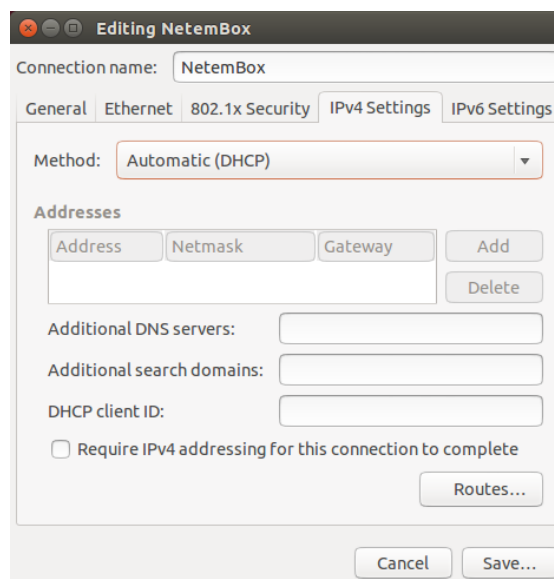


Figure E.4: Client IP address allocation method

5. Or if a particular IP address is required for this connection on the client computer, use 'Manual' method in the drop-down menu and assign the desired IP address:

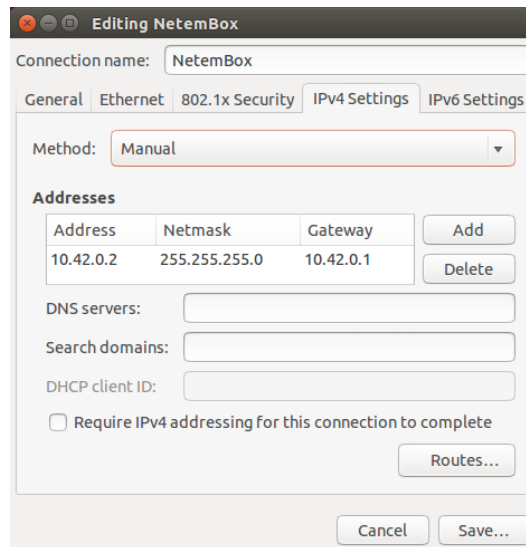


Figure E.5: Assigning a manual IP address to client computer for server connection

- Now in 'General' tab, check the box next to the option “Automatically connect to this network when it is available”:

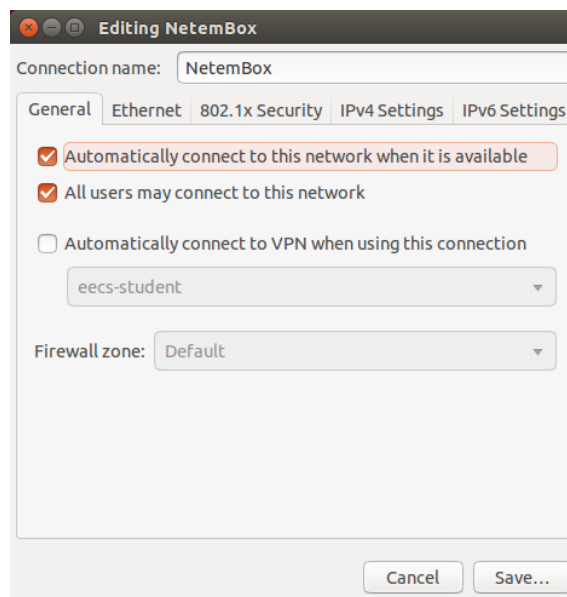


Figure E.6: The option for connecting to the server automatically whenever possible

- Click save, and this newly created connection on the client computer will be saved and listed in Network Connections (under the open/edit network connections menu):

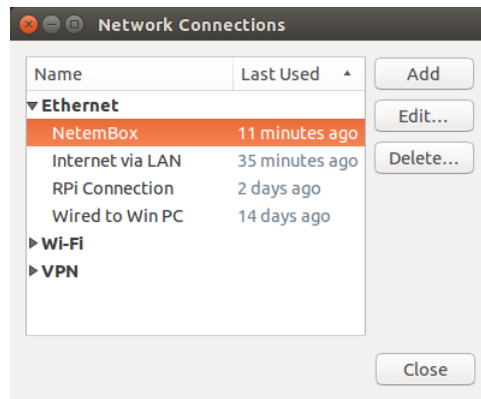


Figure E.7: Newly created connection, NetEmBox

Now every time the client-server connectivity is required through this connection, choose this in the network drop-down menu, click it, and the connection will be established:

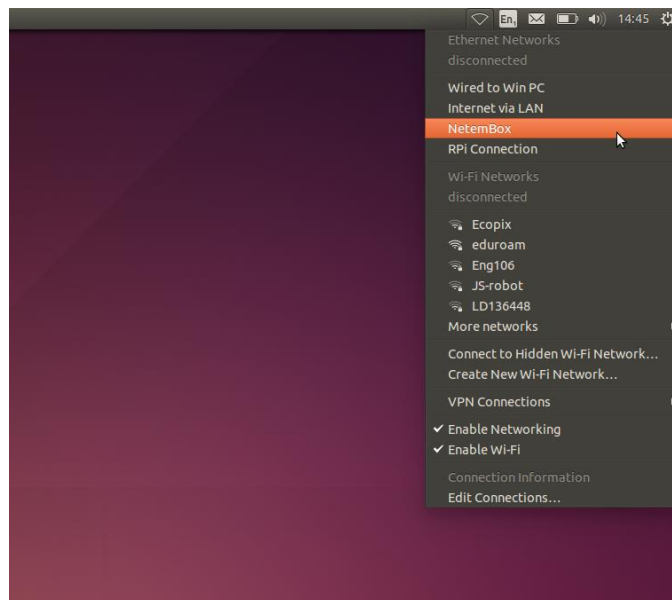


Figure E.8: To connect to the server

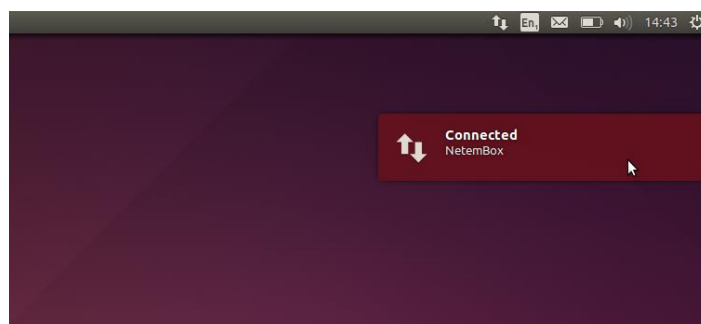


Figure E.9: Client successfully connected to the server

Opening a Network Streamed Video on a Client computer

The steps below show how to open a network video stream initiated from the server on the network.

1. On client computer, open VLC media player and on the menu bar, toggle 'Media' drop-down menu and choose 'Open Network Stream':

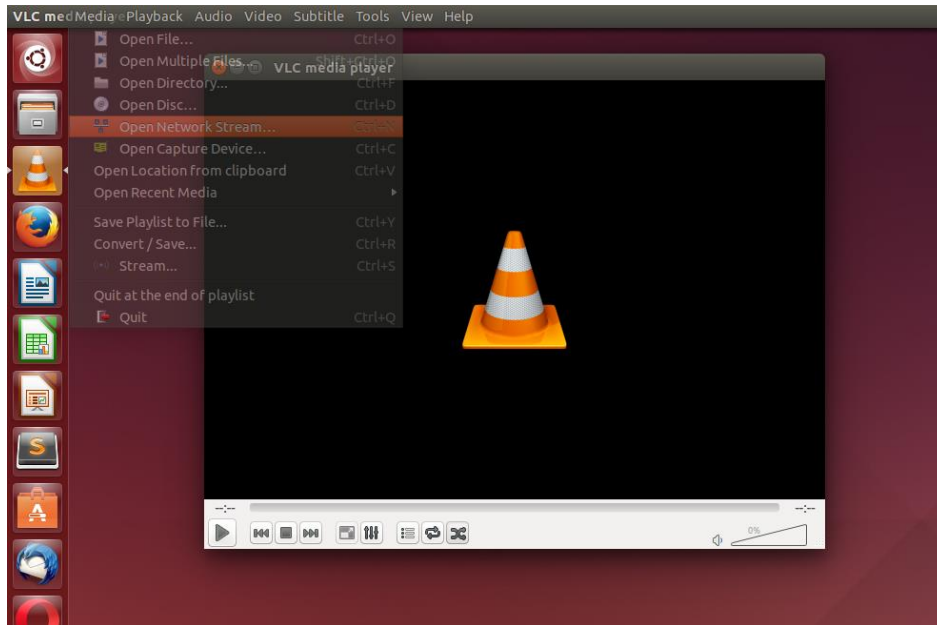


Figure E.10: Opening a network stream in VLC on the client computer

2. Now 'Open Media' window will open and ask for a network URL, enter the streaming protocol followed by the IP address to which the stream was sent from the server:

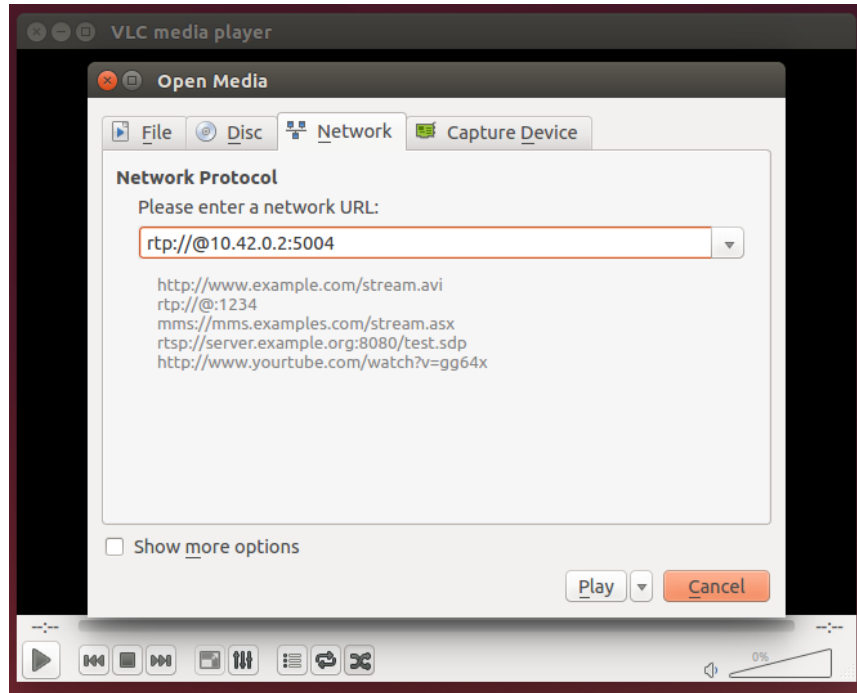


Figure E.11: Providing the steaming protocol and the IP address for VOD

3. Now click 'Play' button, and the video that is being streamed from the server will start playing on the client computer:



Figure E.12: The Client computer successfully opens the network streaming

If the video does not play, go back to the server and ensure the video is still being streamed and has not stopped in case of a video clip with a short duration.

Appendix F Matlab Programs used in the Research

Matlab code to create histogram from the RTT values

The code below creates a histogram from the RTT values extracted out of the raw ping data.

```
% The Round Trip Time (RTT) values extracted out of the ping data
% are imported into Matlab. A variable called 'varName1,2,3...'
% gets created that holds these values.

% This variable 'VarName1' can now be used in the script
% to refer to.

% WARNING: ALTHOUGH THIS IMPORTED VARIABLE MAY BE CALLED 'VarName1...', BUT
% THE VALUES STORED IN THIS GET OVERWRITTEN IF ONE DATASET IS IMPORTED
% AFTER THE OTHER! IF UNSURE, IMPORT THE REQUIRED PING VALUES AGAIN.

[n,x] = hist(VarName1) % creates a histogram of ping values

histogram(VarName1,'BinLimits',[min(VarName1) max(VarName1)])

maxValue = max(VarName1) % finds the maximum value in the dataset
minValue = min(VarName1) % finds the minimum value in the dataset

indexmin = find(min(x) == x); % locates the smallest value in the graph data
xmin = x(indexmin);
ymin = n(indexmin);

indexmax = find(max(x) == x); % locates the biggest value in the graph data
xmax = x(indexmax);
ymax = n(indexmax);

strmin = ['Minimum = ',num2str(minValue)]; % labels the smallest value in the graph data
text(xmin,ymin,strmin,'VerticalAlignment','bottom','HorizontalAlignment','center','Color','magenta','FontSize',12);

strmax = ['Maximum = ',num2str(maxValue)]; % labels the smallest value in the graph data
text(xmax,ymax,strmax,'VerticalAlignment','bottom','HorizontalAlignment','center','Color','blue','FontSize',12);

% title to be modified depending on the applied loss level
title('Applied Packet Loss = 0.1%; Packets Transmitted = 10,000')
xlabel('Delay in milliseconds (ms)') % x-axis label
ylabel('Number of packets') % y-axis label

% end of program
```

Matlab program to convert histograms into the PMF

```
edges = 0:15:150 % used to extend the x-axis with a certain number of bins.
[n,x] = hist(VarName1); % creates a histogram and stores in 'n'
pmf = n./sum(n); % converts histogram to the PMF
stem(x, pmf) % plots the PMF in stem graph
```

Matlab program for calculating standard deviation of the number of packets lost

```

% This program calculates standard deviation of the number of packets lost
x_i = [11 7 10 10 10 11 9 8 10 8]; % data values, to be filled in for each experiment

runs = length(x_i)      % number of runs
x_bar = 9.4;            % mean value of data

s = 0;                  % sum

for counter = 1:runs

    calc = ((x_i(counter) - x_bar).^2); % takes each value from the vector
                                           % and uses in this one by one
    s = s + calc;          % adds each calc value to the sum
end

std_dev = sqrt((s)/(runs-1))      % formula for standard deviation of packets lost

```

Matlab program to plot the error bar for loss values from all experiments

```

% Calculating the error bar for all loss value experiments

losses = [0.01 0.1 1 5 10]; % applied plp from NetEm

mean_loss = [9.4 10.9 9.8 11.5 9.6]; % the average packet loss

E = [1.1 1.7 1.5 3.19 2.22]; % standard deviation of number of packets lost

errorbar(losses, mean_loss, E, 'rx'); % plots the error bar

title('Error Bar Graph of Packet Loss')
xlabel('Expected PLP(%)')
ylabel('Number of Packets Lost')

```


Appendix G Queue Model for Jitter

The common NetEm offered features that have been used by the researchers in the past, have the same limitation as a traditional network lab, which is the inability of the NetEm delay jitter model to represent realistic network traffic models to reflect the behaviour of real world networks. The NetEm default method of inputting delay and jitter adds a fixed amount of delay on the outgoing traffic. NetEm also allows one to add variation to the added delay along with a correlation. However, the outputted delays are still generated from a small range of predefined values, and are hence not much like real Internet traffic that has a vast range of delay patterns.

NetEm pre-loaded distributions Normal and Pareto can also be used to apply delays and jitter to Network traffic. However, there are some limitations due to which the produced delays on the interface still do not represent realistic delay patterns. Normal distribution, for example, does not allow implementation of delays larger than mean delay added with four times the standard deviation. As a result, it is not possible to study the long tails seen in real networks. In Pareto distribution, shape parameter alpha that is a positive number, is very sensitive to standard deviation which means alpha value has to be inputted very precisely to have a desired delay. NetEm, however, puts a limit on this input value by allowing entry of only two decimal places of precision, which as a result, does not give the required accuracy when the delay is added in a way allowed in NetEm by default.

Creation of Bespoke Packet Delay Distribution

This model starts with the literature on queueing in packet buffers. Beyond simple classical queue models, like the M/D/1 (for fixed packet lengths), the next most widely used model is the N*on-off/D/1, which also features fixed packet lengths, and in which N identical on-off traffic source models generate packets for queueing in a fixed length (but generally very large) buffer [79] [179]. Each source has a packet generation rate (when it is on) of R packets per sec, and 0 when it is off. On and off periods are assumed exponentially distributed with mean T_{on} and T_{off} respectively. This is shown in Figure G.1.

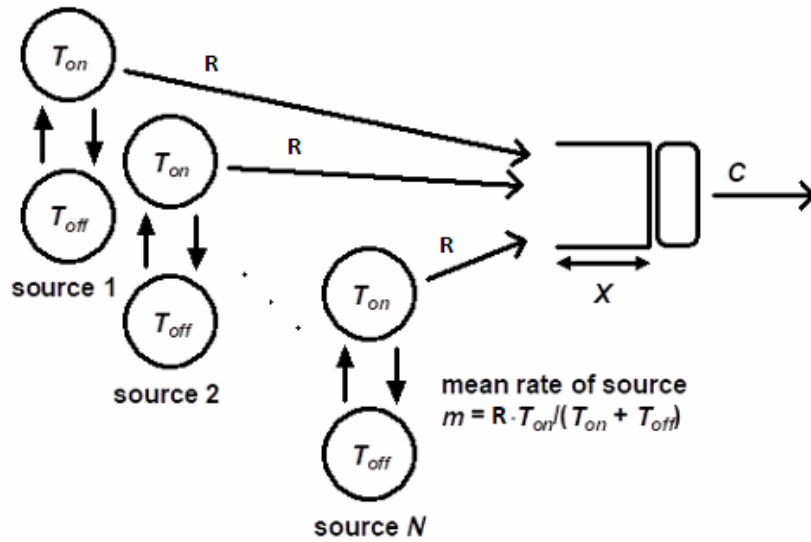


Figure G.1: Bespoke Packet Delay Distribution

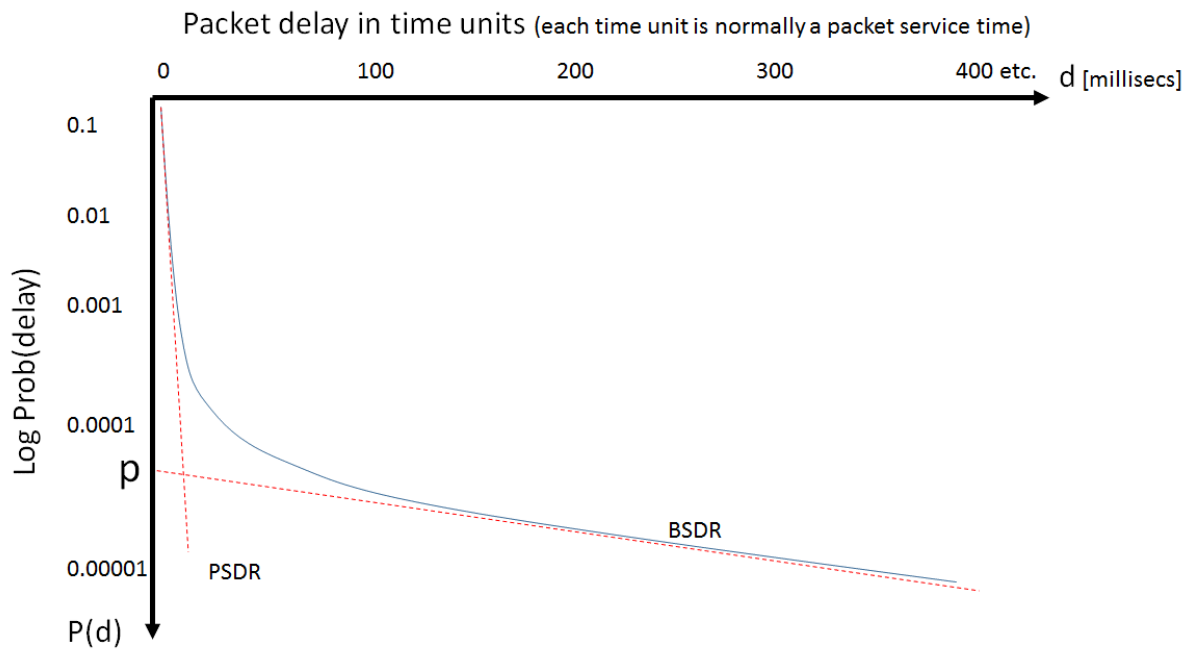


Figure G.2: Delay Distribution

The procedure required to get a NetEm delay distribution model out of this Burst Scale queueing model is as follows:

- Equate the Jitter to the standard deviation of the delay distribution shown in Figure G.2.
- Note that the distribution model shown in Figure G.2 is the weighted sum of 2 geometric distributions:

$$p(d) = p \cdot (pps(d)) + (1 - p) \cdot (pbs(d)) \tag{G.1}$$

Where:

d = delay

$p(d)$ = Probability of the delay

pps = Packets per second

- Note that for the Geometric distribution, the mean \approx standard deviation (in the limit, as the Gem becomes the Exponential, the mean is exactly equal to the standard deviation)
- Parameterize the distribution model shown in Figure G.2 through the mean such that

Chosen Jitter = overall mean of $P(d)$ distribution, which is:

$$p \cdot (\text{PS mean}) + (1 - p) \cdot (\text{BS mean}) \quad (\text{G.2})$$

Where:

PS = packet scale

BS = burst scale

- Choose service rate (bandwidth) of the link, and therefore set the packet service times too
- Implement the above model in MATLAB, such that it can generate a stream of random number from this distribution
- Use NetEm facility to turn these random numbers into a bespoke distribution, for details refer to Section 4.11 on page 59.

Appendix H Recording the MOS

Figure H.1 shows the survey sheet used for collecting MOS from research participants.

Mean Opinion Score (MOS)

Mujtaba Roshan, Networks Research Group

Version: May 2017

Instructions for the participants:

Thank you for taking part in this study. For research purposes, your opinion will be taken on the playback quality of a video. You may notice some distortion, or the video frames being delayed, in the video.

Once you have seen the video(s), please give your opinion score on the video quality you experienced. The table below will help you understand the scoring system.

Table 1: MOS Value Ratings

MOS	Quality	Impairment
5	Excellent	Imperceptible (you didn't even notice any imperfections)
4	Good	Perceptible but not annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Bad	Very annoying

Please record your opinion score in the table below.

Experiment / Run No.	Applied Delay (ms)	Applied Jitter (ms)	Delay Distribution	Recorded MOS	Participant's Age Group*
i					<i>Please tick the relevant box:</i> 10 – 18 years <input type="checkbox"/> 19 – 30 years <input type="checkbox"/> 31 – 45 years <input type="checkbox"/> 46 – 65 years <input type="checkbox"/> Over 65 years <input type="checkbox"/>
ii					
iii					
iv					
v					

* People in different age groups may perceive the video quality differently.

Date: _____

Figure H.1: Survey sheet used in this research.