Secure VoIP Performance Measurement

by

Amna Saad

A Doctoral Thesis

Submitted in partial fulfilment
of the requirements for the award of

Doctor of Philosophy
of
Loughborough University

30th April 2013

Loughborough University

Thesis Access Form

Copy No……..…….**Location**…….……...…

Author……..……..…….

Title……..

Status of access OPEN / RESTRICTED / CONFIDENTIAL

Moratorium Period:……years, ending……../……200…….

Conditions of access approved by (CAPITALS):……

Supervisor (Signature)……...……...

School of……...……

Author's Declaration: *I agree the following conditions:*

Open access work shall be made available (in the University and externally) and reproduced as necessary at the discretion of the University Librarian or Dean of School. It may also be digitised by the British Library and made freely available on the Internet to registered users of the EThOS service subject to the EThOS supply agreements.

*The statement itself shall apply to **ALL** copies including electronic copies:*

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Restricted/confidential work: All access and any photocopying shall be strictly subject to written permission from the University Dean of School and any external sponsor, if any.

Author's signature…….Date……..……...……...

users declaration: for signature during any Moratorium period (Not Open work): ***I undertake to uphold the above conditions:***			
Date	Name (CAPITALS)	Signature	Address

4.1e

Research Student Office, Academic Registry
Loughborough University, Leicestershire, LE11 3TU, UK
Switchboard: +44 (0)1509 263171 Fax: +44 (0)1509 223938

Loughborough University

Certificate of Originality

This is to certify that I am responsible for the work submitted in this thesis, that the original work is my own except as specified in acknowledgements or in footnotes, and that neither the thesis nor the original work contained therein has been submitted to this or any other institution for a higher degree.

. . . . . .

Amna Saad

1st May 2013

Abstract

This project presents a mechanism for instrumentation of secure VoIP calls. The experiments were run under different network conditions and security systems. VoIP services such as Google Talk, Express Talk and Skype were under test. The project allowed analysis of the voice quality of the VoIP services based on the Mean Opinion Score (MOS) values generated by Perceptual Evaluation of Speech Quality (PESQ). The quality of the audio streams produced were subjected to end-to-end delay, jitter, packet loss and extra processing in the networking hardware and end devices due to Internetworking Layer security or Transport Layer security implementations. The MOS values were mapped to Perceptual Evaluation of Speech Quality for wideband (PESQ-WB) scores. From these PESQ-WB scores, the graphs of the mean of 10 runs and box and whisker plots for each parameter were drawn. Analysis on the graphs was performed in order to deduce the quality of each VoIP service. The E-model was used to predict the network readiness and Common Vulnerability Scoring System (CVSS) was used to predict the network vulnerabilities. The project also provided the mechanism to measure the throughput for each test case. The overall performance of each VoIP service was determined by PESQ-WB scores, CVSS scores and the throughput. The experiment demonstrated the relationship among VoIP performance, VoIP security and VoIP service type. The experiment also suggested that, when compared to an unsecure IPIP tunnel, Internetworking Layer security like IPSec ESP or Transport Layer security like OpenVPN TLS would improve a VoIP security by reducing the vulnerabilities of the media part of the VoIP signal. Morever, adding a security layer has little impact on the VoIP voice quality.

KEYWORDS: VoIP, security, performance, E-Model, CVSS, PESQ, jitter, delay, packet loss rate, throughput.

Nomenclature

AAA Authentication, Authorization and Accounting

AES Advanced Encryption Standard

AH Authentication Header

API Application Program Interface

CA Certificate Authority

CBR Constant Bit Rate

CLI Command Line Interface

CODEC Coder/decoder

CoS Class of Service

CPAN Comprehensive Perl Archive Network

CPU Central Processing Unit

CVSS Common Vulnerability Scoring System

DDoS Distributed DoS

Diffserv Differentiated services

DNS Domain Name Service

DoS Denial of Service

DRDoS Distributed Reflector DoS

DS0 a basic digital signaling rate of 64 kbit/s for one voice channel

DTLS Datagram Transport Layer Security

EF Expediated Forwarding

ENUM E.164 NUmber Mapping

ESP Encapsulated Security Payload

ETSI European Telecommunications Standards Institute

H.225 An ITU-T call signalling protocol (part of the H.323 suite)

H.235 An ITU-T security protocol (part of the H.323 suite)

H.245 An ITU-T capability exchange protocol (part of H.323 suite)

H.248 An ITU-T signalling protocol (part of the H.323 suite)

H.323 An ITU-T standard protocol suite for real-time communications over a packet network

HMAC Hash-based Message Authentication Code

HTTP Hypertext Transfer Protocol

IAX Inter-Asterisk eXchange

ICMP Internet Control Message Protocol

ICT Information and communication technologies

IDS Intrusion Detection System

IEEE Institute of Electrical and Electronic Engineers

IETF Internet Engineering Task Force

iLBC internet Low Bitrate Codec

IM Instant Messaging

IP Internet Protocol

IPSec IP Security, a set of protocol developed by the IETF to support secure exchange of packets at the IP layer

IPS Intrusion Prevention System

ISAKMP Internet Security Association and Key Management Protocol

ISO International Organization for Standardization

ISP Internet Service Provider

ITU-R ITU Radiocommunication Sector

ITU-T ITU Telecommunication Standardization Sector

ITU International Telecommunications Union

KVM keyboard, video or visual display unit, mouse

LAMP LINUX Apache MySQL and PHP

LLQ Low Latency Queueing

M/M/1 An M/M/1 queue consists of a first-in-first-out buffer with packets arriving randomly according to a Poisson process, and a processor that retrieves packets from the buffer at a specified service rate.

M2E mouth-to-ear

MGCP Media Gateway Control Protocol

MitM Man-in-the-Middle

MNB Measuring Normalizing Blocks

MOS Mean Opinion Score, MOS provides a numerical indication (range 1 to 5, where 1 is the lowest) of perceived quality of received media after compression and/or transmission. Specified under ITU-T Recommendation P.800

MPLS Multiprotocol Label Switching

NAT Network Address Translation

NIST The National Institute of Standards and Technology

P2P Peer To Peer

PAMS Perceptual Analysis Measurement System

PAT Port Address Translation

PBX Private Branch eXchange, usually used on business premises to switch telephone calls

PESQ-WB PESQ-Wideband

PESQ Perceptual Evaluation of Speech Quality

PHB Per Hop Behaviour

PKI Public Key Infrastructure

PLC Packet Loss Concealment

PSQM Perceptual Speech Quality Measures

PSTN Public Switched Telephone Network

QoS Quality of Service

RED Random Early Detection

ROI Return On Investment

RTCP RTP Control Protocol

RTP Real-Time Transport Protocol

RTT Round-trip Time

SA Security Association

SASL Simple Authentication and Security Layer

SCCP Skinny Client Control Protocol

SCTP Stream Control Transmission Protocol

SDP Session Description Protocol

SIP Session Initiation Protocol

S/MIME Secure/Multipurpose Internet Mail Extensions

SOX SOund eXchange

SPI Security Parameter Index

SPIT Spam over IP Telephony

SPL Sound Pressure Level

SRTP Secure Real-Time Transport Protocol

TCP Transmission Control Protocol

TLS Transport Layer Security

UDP User Datagram Protocol

URI Uniform Resource Identifier

VAC Virtual Audio Cable

VAD Voice Activity Detection

VBR Variable Bit Rate

VoIP Voice over Internet Protocol

VPN Virtual Private Network

WAN Wide Area Network

WiMAX Worldwide Interoperability for Microwave Access

XML Extensible Markup language

XMPP Extensible Messaging and Presence Protocol

XORP eXtensible Open source Routing Platform

YMSG Yahoo Messenger

ZRTP Zimmermann Real-Time Transport Protocol

Acknowledgements

First and foremost I thank Allah for His blessing and for giving me the strength to complete this project and this thesis. I am grateful to my sponsor, Majlis Amanah Rakyat and my employer, Universiti Kuala Lumpur for giving me the opportunity to pursue my childhood dream to excel in my tertiary education.

I am heartily thankful to my main supervisor, Dr. Iain Phillips, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject. I owe my deepest gratitude to my second supervisor, Dr. Ana Salagean, for her unselfish and unfailing support especially during my thesis writing. I also like to thank Dr. Olaf Maennel for his encouragement and for believing that there is something special about my project.

I thank my husband, Roshidi Amran, my son, Umar Roshidi and my daughters, Nurul Roshidi and Nur Roshidi for being so patient with me. I just like to tell them that I love them very much.

I thank my mum, my in-laws, my sister, my brothers and other members of my extended family for their understanding and for being so patient with this temporary separation. I pray for Allah blessing for my late father, who passed away on the 12th of August 2007, 7 days after I left Malaysia for United Kingdom.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

I dedicate this thesis to my family and my friends. It has been a very long but a meaningful journey.

Amna Saad

List Of Publications

1. A. Saad, I.W. Phillips, and A.M. Salagean. A framework for monitoring the performance of secure VoIP. *In Proc. Proceedings of the Third International Conference on Internet Technologies and Applications (ITA 09), pages 495 – 505, 2009.*

Table of Contents

Abstract **1**

Nomenclature **2**

Acknowledgements **7**

List Of Publications **8**

Table of Contents **9**

List of Figures **15**

List of Tables **20**

1 Introduction **21**

1.1 Voice over Internet Protocol 21
1.2 Aim 22
1.3 Objective 22
1.4 Motivation 23
1.5 Contribution 24
1.6 Thesis Organization 25

2 Related Work **26**

2.1 Background 26
2.1.1 VoIP, Internet and PSTN 27
2.1.2 Major Concerns For VoIP 31
2.1.3 Summary 31
2.2 VoIP Protocols 32
2.2.1 SIP 34
2.2.2 RTP 38
2.2.3 H.323 39

2.2.4 MGCP 39
2.2.5 Megaco 39
2.2.6 XMPP 39
2.2.7 Skype 40
2.2.8 Skinny Client Control Protocol (SCCP) 41
2.2.9 YMSG 41
2.2.10 Inter-Asterisk eXchange version 2 (IAX2) 41
2.2.11 Summary 42
2.3 VoIP Security Issues 42
2.3.1 Social Threat 44
2.3.2 Eavesdropping 47
2.3.3 Interception and Modification 49
2.3.4 Intentional Interruption of Service 49
2.3.5 Service Abuse 53
2.3.6 Other Interruptions of service 53
2.3.7 Summary 53
2.4 VoIP QoS 54
2.4.1 VoIP Metrics 55
2.4.1.1 Bandwidth Sharing, Congestion Control and Time Sensitive Traffic 57
2.4.1.2 Data Reception 58
2.4.1.3 Multipoint or Single-point Measurement 60
2.4.2 Bandwidth Calculator 61
2.4.2.1 VoIP per call bandwidth 61
2.4.3 QoS Operations and Algorithms 63
2.5 Tools for Assessing Voice Quality 64
2.5.1 Mean Opinion Score 67
2.5.2 Perceptual Speech Quality Measure 67
2.5.3 Measuring Normalizing Blocks 68
2.5.4 Perceptual Analysis Measurement System 68
2.5.5 Perceptual Evaluation of Speech Quality 69
2.5.6 E-model 70
2.5.7 Objective Listening Quality Assessment 70
2.5.8 Summary 71
2.6 Previous Work 72
2.6.1 Quality of Service (QoS) 72
2.6.2 Security 73

2.6.3 QoS and Security 73
2.6.4 Summary 74

3 Methodology 75
3.1 Introduction 75
3.2 Instrumentation 77
3.2.1 Service Conditions 78
3.2.2 Scope 80
3.3 Testing Method 83
3.3.1 Test Scenarios 83
3.3.2 Test Parameters 84
3.3.3 Testing, Analysis and Verification 84
3.4 Summary 86

4 Project Implementation 87
4.1 First Stage: Network 88
4.2 Second Stage: Monitoring Tools and Packet Interceptor 92
4.3 Third Stage: VoIP clients 96
4.4 Fourth Stage: Script To Automate Experiment 97
4.5 Fifth Stage: Data Collection and Data Processing 99
4.5.1 Data Collection 101
4.5.2 Data Processing 101
4.6 Sixth Stage:Validation And Verification 103
4.6.1 Tools 103
4.6.2 Data Collection 108
4.7 Summary 110

5 Results and Analysis 111
5.1 Voice Quality:PESQ-WB Scores 113
5.1.1 Delay 113
5.1.2 Packet Loss Rate 117
5.1.3 Jitter 121
5.2 Security:Vulnerability Score 128
5.2.1 CVSS base score 128
5.3 Bandwidth Utilisation 131
5.4 Overall Scores 147
5.5 Summary 149

6 Knowledge Presentation And Distribution 150
6.1 Information Convergence 150
6.1.1 Algorithm 150
6.1.1.1 Four Coloured Zones 151
6.1.1.2 Network Impairments 151
6.1.2 CPU Power and Operating System Type 155
6.1.3 Source of Delays 156
6.1.4 Packet Loss Rate 157
6.1.5 Codec and Bandwidth Utilisation 158
6.2 Summary 158

7 Conclusion 159
7.1 Process Flow 159
7.2 Potential Research Works 160
7.3 Different Technologies under Test 161
7.4 Summary 162

References 163

A i
A.1 Network Configuration and Monitoring i
A.1.1 Routers' Scripts i
A.1.1.1 Source File: R3_config i
A.1.1.2 Source File: R2_config:IPIP tunnel iii
A.1.1.3 Source File: R2_config:IPSec tunnel v
A.1.1.4 Source File: R2_config:OpenVPN TLS viii
A.1.1.5 Source File: R1_config:IPIP Tunnel xi
A.1.1.6 Source File: R1_config:IPSec Tunnel xiii
A.1.1.7 Source File: R1_config:OpenVPN TLS xvii
A.1.1.8 Source File: TR_config: IPIP Tunnel xix
A.1.2 Switch's Script xxi
A.1.2.1 Source File: switch xxi
A.1.3 DummyNet's Scripts xxii
A.1.3.1 Bridging and Firewalling xxii
A.1.3.2 Delay xxiii
A.1.3.3 Jitter xxiv
A.1.3.4 Packet Loss Rate xxv

B **xxvi**
B.1 Semi-automated Scripts xxvi
B.1.1 Common Files xxvi
B.1.1.1 Source File: playsox.bat xxvi
B.1.1.2 Source File: recsox.bat xxvi
B.1.1.3 Source File: flush-ipfwrule-remotely.bat xxvii
B.1.1.4 Source File: setipfwrule-remotely.bat xxvii
B.1.1.5 Source File: flushout.sh xxvii
B.1.1.6 Source file: runtcpdump.sh xxviii
B.1.2 Skype xxviii
B.1.2.1 Source File: datacollection.bat xxviii
B.1.2.2 Source File: recandplay.bat xxviii
B.1.2.3 Source File: skypeClientAattach.pl xxix
B.1.3 SIP xxix
B.1.3.1 Source File: datacollection.bat xxix
B.1.3.2 Source File: recandplay.bat xxx
B.1.3.3 Source File: SIPRegisterNCall.bat xxx
B.1.3.4 Source File: clientAcallclientB.au3 xxx
B.1.3.5 Source File:hangupcall.au3 xxxi
B.1.4 Google Talk xxxi
B.1.4.1 Source File: datacollection.bat xxxi
B.1.4.2 Source File: recandplay.bat xxxi
B.1.4.3 Source File: clientAcallclientB.au3 xxxi
B.1.4.4 Source File: clientBacceptsClientA.au3 xxxii

C **xxxiii**
C.1 Software Tools xxxiii
C.1.1 PESQ xxxiii
C.1.1.1 Link To PESQ source codes xxxiii
C.1.1.2 Sample of PESQ run script xxxiii
C.1.1.3 Sample outout of PESQ run xxxiii
C.1.2 Pcap File Processing xxxiii
C.1.2.1 Sample of Ipsumdump script and rrdtool script xxxiii
C.1.2.2 Sample of Ipaggcreate script xxxv
C.1.3 ICMP Ping xxxv
C.1.3.1 Sample of ICMP ping command xxxv
C.1.4 E-model xxxv

C.1.5 CVSS xl
C.1.5.1 Link to CVSS calculator xl

D **xli**
D.1 Sample Report xli
D.1.1 Sample Report of PESQ.exe xli
D.1.2 ICMP log xliii
D.1.3 Sample Report of emodel.java lviii
D.1.4 Router log lx
D.1.4.1 TR lx
D.1.4.2 Router1 lxi
D.1.4.3 Router2 lxiii
D.1.5 CACTI report:Snapshots lxvii
D.1.6 Validation: Box and Whisker Graphs lxx

E **lxxxiv**
E.1 Framework lxxxiv

List of Figures

2.1 End-to-end VoIP Data Processing 28
2.2 PSTN versus Internet 29
2.3 Internet Challenges and Opportunities 30
2.4 VoIP Protocol Suite 33
2.5 Voice Data Processing of VoIP 33
2.6 VoIP Protocols 35
2.7 Open Source protocols 36
2.8 Proprietary protocols 37
2.9 VoIP security issues 45
2.10 Network attacks 46
2.11 Social threats 47
2.12 Eavesdropping 48
2.13 Interception and modification 49
2.14 Intentional interruption of services 50
2.15 Other service interruptions 53
2.16 VoIP QoS 56
2.17 QoS measurement points 60
2.18 QoS operations and algorithms 65
2.19 VoIP QoS Assessing tools 71

3.1 Methodology 76
3.2 Parameters that affect QoS 81
3.3 Experimental test parameters 85

4.1 Experimental Network 90
4.2 Laboratory Setup:Overall view 91
4.3 Specific devices:Front view 91
4.4 Scope for a semi-automated VoIP Test-kit 99
4.5 Call thread time line 100

4.6 Data processing process 102
4.7 CVSS Base Score report 106
4.8 NIST CVSS calculator version 2 107
4.9 Graph of Delay versus PESQ-WB 108
4.10 Graph of Delay versus PESQ-WB 109
4.11 Graph of Delay versus PESQ-WB 109

5.1 One way delay vs MOS 112
5.2 Packet Loss Probability vs MOS 112
5.3 One way delay vs R 112
5.4 Packet Loss Probability vs R 112
5.5 SIP protocol: The result of taking the mean of 10 runs 114
5.6 Skype protocol:The result of taking the mean of 10 runs 114
5.7 Google Talk protocol:The result of taking the mean of 10 runs 115
5.8 IPIP tunnel: The result of taking the mean of 10 runs 116
5.9 IPSec tunnel: The result of taking the mean of 10 runs 116
5.10 TLS tunnel: The result of taking the mean of 10 runs 117
5.11 SIP Protocol: The result of taking the mean of 10 runs 118
5.12 Skype Protocol: The result of taking the mean of 10 runs 118
5.13 Google Talk Protocol: The result of taking the mean of 10 runs 119
5.14 IPIP protocol:The result of taking the mean of 10 runs 120
5.15 IPSec protocol:The result of taking the mean of 10 runs 120
5.16 TLS protocol:The result of taking the mean of 10 runs 121
5.17 SIP IPIP Protocol: Box and whisker plot over 10 runs 122
5.18 SIP IPSec Protocol: Box and whisker plot over 10 runs 122
5.19 SIP TLS Protocol: Box and whisker plot over 10 runs 123
5.20 Skype IPIP Protocol: Box and whisker plot over 10 runs 124
5.21 Skype IPSec Protocol: Box and whisker plot over 10 runs 124
5.22 Skype TLS Protocol: Box and whisker plot over 10 runs 125
5.23 Gtalk IPIP Protocol: Box and whisker plot over 10 runs 126
5.24 Gtalk IPSec Protocol: Box and whisker plot over 10 runs 126
5.25 Gtalk TLS Protocol: Box and whisker plot over 10 runs 127
5.26 CVSS Base Score 129
5.27 Media security 130
5.28 Express Talk: Delay 0ms to 90ms 135
5.29 Express Talk: Delay 100ms to 190ms 135
5.30 Express Talk: Delay 200ms to 290ms 135

5.31 Express Talk: Delay 300ms to 400ms 135
5.32 Express Talk: PLR 0.0% to 0.9% 135
5.33 Express Talk: PLR 1.0% to 1.9% 135
5.34 Express Talk: PLR 2.0% to 3.0% 135
5.35 Express Talk: 9 Jitter profiles 135
5.36 Express Talk-IPSec: Delay 0ms to 90ms 136
5.37 Express Talk-IPSec: Delay 100ms to 190ms 136
5.38 Express Talk-IPSec: Delay 200ms to 290ms 136
5.39 Express Talk-IPSec: Delay 300ms to 400ms 136
5.40 Express Talk-IPSec: PLR 0.0% to 0.9% 136
5.41 Express Talk-IPSec: PLR 1.0% to 1.9% 136
5.42 Express Talk-IPSec: PLR 2.0% to 3.0% 136
5.43 Express Talk-IPSec: 9 Jitter profiles 136
5.44 Express Talk-TLS:Delay 0ms to 90ms 137
5.45 Express Talk-TLS:Delay 100ms to 190ms 137
5.46 Express Talk-TLS:Delay 200ms to 290ms 137
5.47 Express Talk-TLS:Delay 300ms to 400ms 137
5.48 Express Talk-TLS:PLR 0.0% to 0.9% 137
5.49 Express Talk-TLS:PLR 1.0% to 1.9% 137
5.50 Express Talk-TLS:PLR 2.0% to 3.0% 137
5.51 Express Talk-TLS:9 Jitter profiles 137
5.52 Skype: Delay 0ms to 90ms 141
5.53 Skype: Delay 100ms to 190ms 141
5.54 Skype: Delay 200ms to 290ms 141
5.55 Skype: Delay 300ms to 400ms 141
5.56 Skype: PLR 0.0% to 0.9% 141
5.57 Skype: PLR 1.0% to 1.9% 141
5.58 Skype: PLR 2.0% to 3.0% 141
5.59 Skype: 9 Jitter profiles 141
5.60 Skype-IPSec: Delay 0ms to 90ms 142
5.61 Skype-IPSec: Delay 100ms to 190ms 142
5.62 Skype-IPSec: Delay 200ms to 290ms 142
5.63 Skype-IPSec: Delay 300ms to 400ms 142
5.64 Skype-IPSec: PLR 0.0% to 0.9% 142
5.65 Skype-IPSec: PLR 1.0% to 1.9% 142
5.66 Skype-IPSec: PLR 2.0% to 3.0% 142
5.67 Skype-IPSec: 9 Jitter profiles 142

5.68 Skype-TLS: Delay 0ms to 90ms 143
5.69 Skype-TLS: Delay 100ms to 190ms 143
5.70 Skype-TLS: Delay 200ms to 290ms 143
5.71 Skype-TLS: Delay 300ms to 400ms 143
5.72 Skype-TLS: PLR 0.0% to 0.9% 143
5.73 Skype-TLS: PLR 1.0% to 1.9% 143
5.74 Skype-TLS: PLR 2.0% to 3.0% 143
5.75 Skype-TLS: 9 Jitter profiles 143
5.76 GTalk: Delay 0ms to 90ms 144
5.77 GTalk: Delay 100ms to 190ms 144
5.78 GTalk: Delay 200ms to 290ms 144
5.79 GTalk: Delay 300ms to 400ms 144
5.80 GTalk: PLR 0.0% to 0.9% 144
5.81 GTalk: PLR 1.0% to 1.9% 144
5.82 GTalk: PLR 2.0% to 3.0% 144
5.83 GTalk: 9 Jitter profiles 144
5.84 Gtalk-IPSec:Delay 0ms to 90ms 145
5.85 Gtalk-IPSec:Delay 100ms to 190ms 145
5.86 Gtalk-IPSec:Delay 200ms to 290ms 145
5.87 Gtalk-IPSec:Delay 300ms to 400ms 145
5.88 Gtalk-IPSec:PLR 0.0% to 0.9% 145
5.89 Gtalk-IPSec:PLR 1.0% to 1.9% 145
5.90 Gtalk-IPSec:PLR 2.0% to 3.0% 145
5.91 Gtalk-IPSec:9 Jitter profiles 145
5.92 Gtalk-TLS:Delay 0ms to 90ms 146
5.93 Gtalk-TLS:Delay 100ms to 190ms 146
5.94 Gtalk-TLS:Delay 200ms to 290ms 146
5.95 Gtalk-TLS:Delay 300ms to 400ms 146
5.96 Gtalk-TLS:PLR 0.0% to 0.9% 146
5.97 Gtalk-TLS:PLR 1.0% to 1.9% 146
5.98 Gtalk-TLS:PLR 2.0% to 3.0% 146
5.99 Gtalk-TLS:9 Jitter profiles 146

6.1 CVSS to PESQ-WD chart for Delay between 0ms and 150ms 153
6.2 CVSS to PESQ-WD chart for Delay between 150ms and 400ms 153
6.3 CVSS to PESQ-WD chart for PLR between 0% and 1% 154
6.4 CVSS to PESQ-WD chart for PLR betweeen 1% and 2% 154

6.5 CVSS to PESQ-WD chart for PLR betweeen 2% and 3% 155

D.1 List of devices on CACTI lxvii
D.2 Snapshot of TR memory usage and CPU usage lxviii
D.3 Snapshot of Router1 memory usage and CPU usage lxix
D.4 Snapshot of Router2 memory usage and CPU usage lxix
D.5 Graph of Jitter versus PESQ-WB lxxi
D.6 Graph of Jitter versus PESQ-WB lxxi
D.7 Graph of Jitter versus PESQ-WB lxxii
D.8 Graph of Packet Loss Rate versus PESQ-WB lxxii
D.9 Graph of Packet Loss Rate versus PESQ-WB lxxiii
D.10 Graph of Packet Loss Rate versus PESQ-WB lxxiii
D.11 Graph of Jitter versus PESQ-WB lxxiv
D.12 Graph of Jitter versus PESQ-WB lxxiv
D.13 Graph of Jitter versus PESQ-WB lxxv
D.14 Graph of Packet Loss Rate versus PESQ-WB lxxv
D.15 Graph of Packet Loss Rate versus PESQ-WB lxxvi
D.16 Graph of Packet Loss Rate versus PESQ-WB lxxvi
D.17 Graph of Delay versus PESQ-WB lxxvii
D.18 Graph of Delay versus PESQ-WB lxxvii
D.19 Graph of Delay versus PESQ-WB lxxviii
D.20 Graph of Jitter versus PESQ-WB lxxix
D.21 Graph of Jitter versus PESQ-WB lxxix
D.22 Graph of Jitter versus PESQ-WB lxxx
D.23 Graph of Packet Loss Rate versus PESQ-WB lxxx
D.24 Graph of Packet Loss Rate versus PESQ-WB lxxxi
D.25 Graph of Packet Loss Rate versus PESQ-WB lxxxi
D.26 Graph of Delay versus PESQ-WB lxxxii
D.27 Graph of Delay versus PESQ-WB lxxxii
D.28 Graph of Delay versus PESQ-WB lxxxiii

E.1 Testbed Architecture lxxxv

List of Tables

2.1 Denial Of Services Attacks [127] 51
2.2 VoIP per call bandwidth calculation 63

3.1 Different Types of VoIP Services 82

4.1 Device Specification 93
4.2 Project IP Addresses 94
4.3 IPIP tunnel 94
4.4 Site-to-site IPSec tunnel 94
4.5 Site-to-site OpenVPN TLS 95
4.6 Base Metrics 105
4.7 Several output from NIST CVSS calculator version 2 107

5.1 Jitter Profiles 123
5.2 VoIP per call bandwidth per CODEC type 132
5.3 Overall Performance 148

Chapter 1

Introduction

"It does not matter how slowly you go so long as you do not stop."

Confucius

1.1 Voice over Internet Protocol

Systems employing the Internet and Voice over Internet Protocol (VoIP) are alternative to the legacy landline telephony system. The services offered by these systems allow users to communicate with their family members, friends, banks and business partners whenever they are online. Particularly, the VoIP service is popular with the Internet users because charges are bound into usually fixed access costs making the price of the long distance calls themselves appear economical and even free. However, VoIP services have several disadvantageous:

i) Voice quality, although improving, is still not as good as the landline telephony [157, 159, 161]. This is because IP networks generally do not guarantee transmission quality. In which case inadequate bandwidth or other network resources and excessive end-to-end delay, jitter and packet loss rate may reduce voice quality.

ii) Reliability is related to the Internet reliability, which is less than the typical 99.999% network availability for the traditional telephony [164]. It would not be much a problem in urban areas whereby the Internet connectivity is relatively good but it creates more concern to remote areas.

iii) A VoIP service is exposed to Internet security vulnerabilities, threats and attacks [166, 47]. There are security loopholes in the Internet which inevitable presence in the VoIP system.

iv) A VoIP service uses resources such as CPU, memory and buffers of both end systems and intermediate devices [90].

v) A VoIP service consumes bandwidth thus reducing available bandwidth for other applications on the Internet [99, 88]. PSTN dedicates 64kbps bandwidth for a call whereas a VoIP service shares available bandwidth with other applications.

1.2 Aim

The aim of this research is to provide information regarding the performance of a secure VoIP to VoIP users, companies IT managers and VoIP service providers. This information can be used by the relevant parties for decision making and creating IT policy within the companies. As for the users they are able to choose which VoIP or Instant Messaging (IM) are most suitable for communicating with their friends, family, business partners and banks.

1.3 Objective

The objectives of this research are three fold: First, to compare and contrast the performance of VoIP services in the interest of the Internet users under different security configurations. The service performance is reflected by the voice quality and security features that are deployed. Voice quality is influenced by traffic performance such as end-to-end delay, jitter and packet loss as well as network resources such as available bandwidth, CPU, host memory and buffer size.

The second objective is to monitor the VoIP network performance from the perspective of a company IT manager or network operator as VoIP services run side-by-side with data packets sharing Internet resources and security vulnerabilities. The trend and the result can be used to derive the company IT policy and for other decision making.

The third objective is to implement a test rig that is able to control experiments conducted on several VoIP services. The test rig can be extended to cater for other VoIP services. This is essential since most researchers estimated the performance of VoIP services instrumented on software simulators or standalone test rigs. In this project, a live testbed is deployed in order to produce more realistic results that are closer to real world problems. This objective supports the first and the second objectives whereby without the test rig we could not measure and analyse the performance of the VoIP calls. The next section describes the reasons that motivate the research behind this work.

1.4 Motivation

Reasonable, good quality and secure VoIP services are beneficial to the public in the long run. However, since VoIP applications are time-sensitive, they often get higher priority over other applications on the Internet. Any attacks on VoIP applications will affect time-insensitive Internet applications. As a result the Internet security problem is magnified with the implementation of VoIP [47]. This situation warrants a further investigation on the behaviour of VoIP services over existing available resources.

VoIP research areas are not limited to the research on QoS, security or both QoS and security, however these areas are the concerns to the end users because they related directly to their conversational experiences on the application. Other areas such as system integration and interoperability issues or integrating VoIP services into IPv4 and IPv6 for an ubiquitous access to IP telephony services based on SIP protocol services, as discussed in [65] and [95] respectively, are important but beyond the scope of our project. It is believed that security in general contributes to the low voice quality in any VoIP services because of the added delay it incurs due to extra security processing. For example, a secure network imposes less security risk but increases the processing time in filtering and forwarding voice packets in the network. Hence a secure network introduces extra delay. This delay might or might not reduce the overall voice quality. If the balance between VoIP security and QoS could be achieved, visually impaired persons would gain benefits from the embedded technology, as more applications that are voice activated can be programmed and transmitted over the Internet. As of now the relationship between VoIP security and QoS is scarcely documented.

A large number of measurements on various VoIP end-points, mouth-to-ear (M2E) delay of commercial IP phones like Cisco, 3Com and PingTel and software based clients like Microsoft Messenger and NetMeeting and others had been performed [81]. The research focused on the effects of packet loss concealment (PLC), silence suppression, clock skew, hangover time and jitter on these clients. They measured M2E delay by recording both the original and output audio in a two-channel (stereo) mode. However, some testing could not be done due to pairwise combinations of end-points. For example, Net2Phone only talks to other Net2Phone clients. To automate delay estimation, they used their self-developed Adelay software.

VoIP QoS can also be used for capacity planning and to gauge the readiness of an access network to handle VoIP traffic. For example, Mehmood et al. assessed VoIP quality over access networks in Pakistan using a delay jitter measurement methodology for evaluating the perceptual quality of voice calls using the ITU-T G.107 speech quality E-model [109]. Passive measurements for voice calls in the presence of background Internet data traffic for G.723.1 and G.729a CODECs were carried out using a non-intrusive parametric model. The R-factor and resultant Mean Opinion Scores (MOS) were calculated at different link loads and conges-

tion hot spots were identified. The result in 2005 showed the inadequacy of access networks for handling VoIP traffic in Pakistan. A recommendation was made to alleviate congestion by increasing the bandwidth capacity in Pakistan access networks.

Other researchers studied the relationship of VoIP security and QoS by understanding the signalling process and how to secure them and measuring QoS and security of VoIP system on OPNET and ns-2 [134, 25]. The results could be used by network designers to design reliable, fault tolerant, scalable and secure VoIP networks. With the underpinning architecture in place, more reliable telephony services could be offered to the general public. This project builds upon this research particularly on measuring the overall performance of a secure VoIP service on the perceptual speech quality and users' experience.

1.5 Contribution

Since the Internet offers a best-effort service in an open and scalable environment, it is important for one to understand how a time-sensitive service would behave under different types of security restrictions. This project is a stepping stone for better understanding on how different types of VoIP services work within the Internet environment.

Particularly, the research provides information pertaining to VoIP performances for three different types of VoIP services. The VoIP services are based on different VoIP protocols. For example: Skype is built on a proprietary protocol, Google Talk is based on the XMPP protocol and Express Talk is a SIP based application. The information gained is useful for the relevant parties to enhance the security of their current networks to prevent the invasion of users privacy. The information gives them chances to choose which VoIP services provide reasonable voice qualities even after additional securities are deployed.

1.6 Thesis Organization

The next five chapters are organised as follows:

Chapter 2	Background, VoIP Protocols, VoIP Security Issues, VoIP QoS and previous work by other researchers are discussed.
Chapter 3	The potential research gaps that relate to the VoIP quality of service and VoIP security are highlighted. The methodology and project framework are presented. The project scope is discussed before the actual project implementation that is described later in Chapter 4.
Chapter 4	The project implementation stages are discussed. The project framework that was discussed in Chapter 3 are refined further and subdivided into six implementation stages that include setting up network, monitoring tools and packet interceptor, installing VoIP clients, writing scripts to automate data collection, data collection and data processing, and last but not least, validation and verification.
Chapter 5	The result of the experiments that have been specified in Chapter 4 are compared. From the comparison, conclusions are made. Any problems with the results are also highlighted.
Chapter 6	This chapter discusses the contribution to knowledge and the lessons learned on the subject.
Chapter 7	This chapter concludes the thesis. It highlights the overall thesis process, the research limitations and further studies on the subject.

Chapter 2

Related Work

"Study the past if you would define the future."

Confucius

This chapter compares the network architectures, users' experience, security and quality of service between VoIP system and PSTN system. In addition, we take a look at the existing VoIP protocols, security issues, and QoS. We also look at the available tools for assessing voice quality. Then we study work by other researchers. Section 2.1 discusses the background of the VoIP system and the PSTN system. Section 2.2 discusses the different types of the VoIP protocols. Section 2.3 discusses the VoIP security issues. Section 2.4 discusses the VoIP QoS. Section 2.5 highlights the tools for assessing voice qualities. Finally in Section 2.6 we look at the research performed by previous researchers.

2.1 Background

This section highlights the differences and similarities between VoIP to the Public Switched Telephone Network (PSTN) pertaining the network architectures, users' experience, security and quality of service. It is important to make comparison especially on the overall performance of a new system to a well known legacy system. In this context the overall performance of a new system includes the quality of the audio signal, the system integrity, reliability and scalability. However, since VoIP is treated as another application in the Internet, it is inevitably important to discuss the Internet along with VoIP and PSTN. Generally, the new system is expected to perform as well as, or better, than the legacy system in order to replace or work side-by-side the legacy system. To date the VoIP service qualities have improved but still not on par with the qualities of PSTN service. However, its tangible and intangible benefits exceed its incapability [71]. One of the tangible benefits of a VoIP service is reduced charges on

long distance calls because the international toll charges induced by PSTN is not applicable to Internet services. In addition, the intangible benefits include immediate support on VoIP services once the IP network connectivity is available and lessening the burden of a network administrator in monitoring multiple networks.

2.1.1 VoIP, Internet and PSTN

VoIP is a fast growing emerging technology. In term of services, it manages to position itself side-by-side with its counterpart: the PSTN. It attracts different parties, ranging from hobbyists that use Internet to get free phone calls on a peer-to-peer basis and enterprise and companies that want to reduce their operational cost. On the other hand, service providers need to rollout new revenue generating services quickly by leveraging existing IP networks to host VoIP services. At the same time they plan to deploy full scale VoIP network infrastructures in order to replace the PSTN network to reduce operational and administrative overhead. By doing so their system administrators would only have to consider one system that hosted voice, data and other media.

It is relatively cheap to make a long distance international call through a VoIP service rather than a PSTN service. This is because network resources such as bandwidth, router CPU and memory are shared between applications in the Internet. Hence, a VoIP service charge would not include a relatively expensive trunk call charge that is normally associated with a PSTN service. In the PSTN, a dedicated end-to-end link is provided to a call and other calls would not be able to use the resources while the link is engaged. As a result of shared resources, the perceived voice quality received at each end of a VoIP service is reduced. This, however, does not prevent users from using VoIP services. To date there are approximately two-billions internet users. The majority of these users are from Asia. The Internet is still growing at a good rate, but the growth rate is not the same all over the world. The growth rate will not increase again until broadband is further developed, and its price rates reduced further [158]. VoIP users are also growing although most of them are hobbyist and IT-savvy. Most users are taking advantage of low cost long-distance calls using PC-to-PC connectivity. There are companies that have migrated their telephony services entirely to VoIP in order to optimise the utilisation of their internal packet networks.

To use the services, VoIP users need IP phones or soft phones and also subscriptions to VoIP service provider of their choice. They prefer using VoIP services to PSTN because companies such as Skype offer free calls or impose relatively small charges to selected destinations. Except for a minor adjustment on telephone types, users can still make land line calls from their analogue phones, IP phones or softphones over existing IP-Based network. Users have similar experiences that they are used to, as though as they are making calls using their

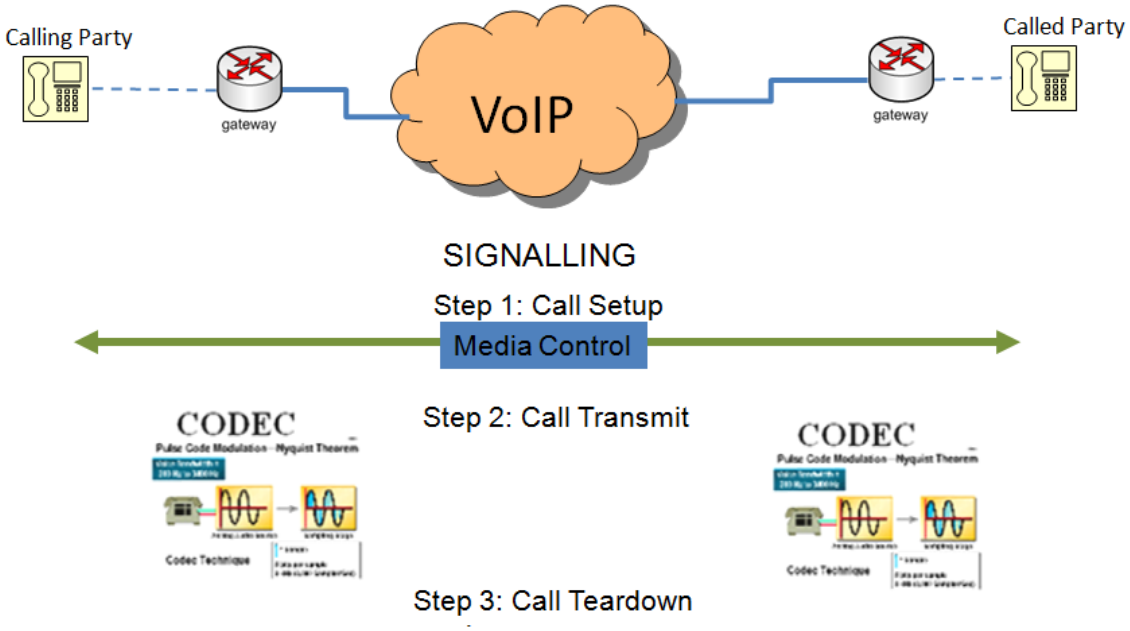


Figure 2.1: End-to-end VoIP Data Processing

landline phones. In fact, VoIP behaves almost identically to PSTN when initiating, managing, and terminating call sessions. The only different is that for a peer-to-peer VoIP, the end devices are logically connected. When someone wants to make a call to a friend, he refers to his buddy (i.e. contact) list for his friend account address. He also checks his friend connection status. He will make the call if his friend is online. On the other hand, a PSTN service is between two designated end devices. So even if the call is successful, it might be left unattended at the other end of the line.

Figure 2.1 on page 28 shows a simplified end-to-end VoIP system process. It shows how a signalling protocol establishes and manages call sessions. The process starts when a user dials a destination number. A signalling protocol tries to establish a communication with the destination phone. Once the call is setup, then a media control protocol negotiates on media processing capabilities such as an audio or a video codec to be used for each media type between two terminals. Then both users start their conversation as usual. Transparent to them, their analogue voices are digitized then are transported over VoIP network and later are decoded at the other end. The process of encoding, transporting and decoding continues, until one of the users hangs up or there is a network problem that suddenly cuts off their conversation. The main different between a VoIP call and a PSTN call is that it is carried over IP-based network whereas a PSTN call is carried over a circuit-switched network. In PSTN, a dedicated communications path is established for the duration of a conversation. No new call can be made using the same channel as long as the path is still established. On the other hand, the Internet is a best-effort public network without central administration that was

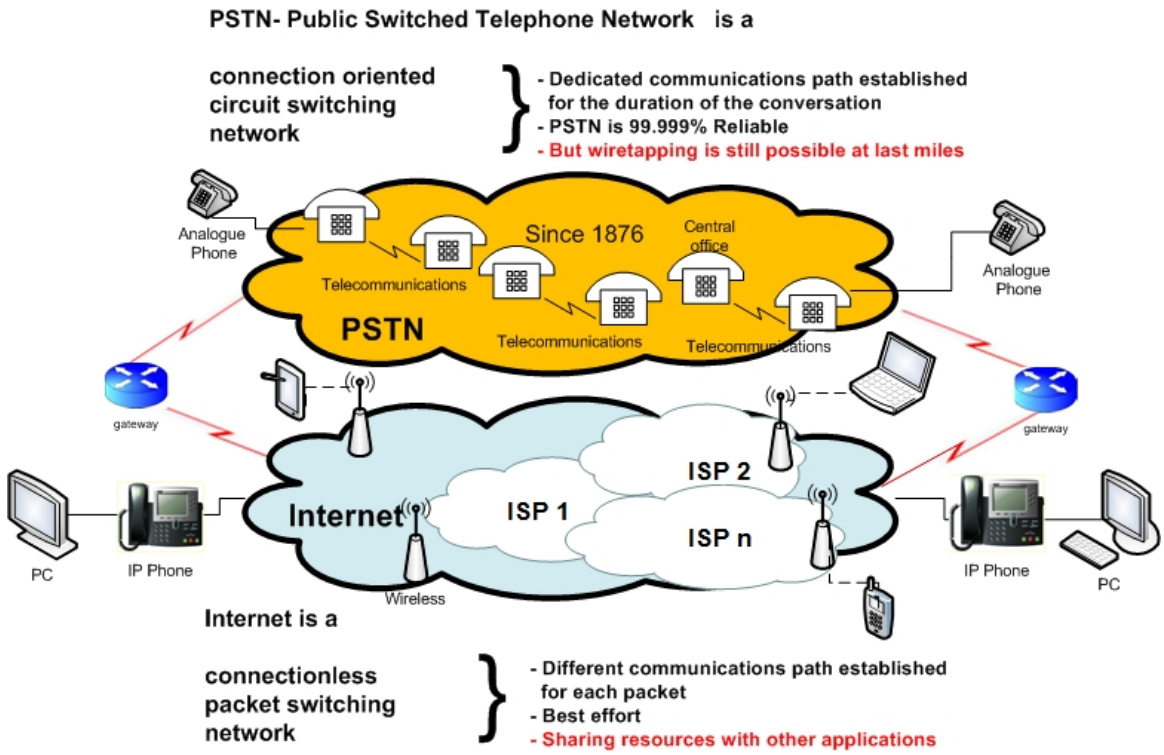


Figure 2.2: PSTN versus Internet

originally designed for openness and scalability, refer to Figure 2.3 on page 30. The design allows the network to grow rapidly. Resources are shared with several applications on the Internet whereby a different communication path is established for each packet transmits over the Internet. Hence, its applications are subjected to various kind of threats and attacks. VoIP calls are treated as another applications on the Internet. Unless given higher priority, VoIP calls must compete for network resources just like any other applications. The very different architecture of VoIP to the traditional circuit-based telephony has resulted in significant security issues. Keromytis suggested that VoIP system represent a higher complexity in terms of its architecture, protocols and implementation which undoubtly would increase the potential for misuse [87]. The PSTN was first introduced by Alexander Graham Bell in 1876 and its services have been improved over the years. It is now a mature technology, with 99.999% service availability [164]. Its equipment has been fine-tuned to provide good voice quality over narrowband frequency of 300Hz to 3400Hz. Once a call is setup, the PSTN provides a dedicated link between two parties. It seems secure, however, wiretapping is still possible at the last mile.

Immature VoIP technology both introduces security risks and opens the door to more exciting applications such as interactive shopping, streaming audio, electronic white-boarding and CD-quality conference calls in stereo though it is believed that the quality of these ser-

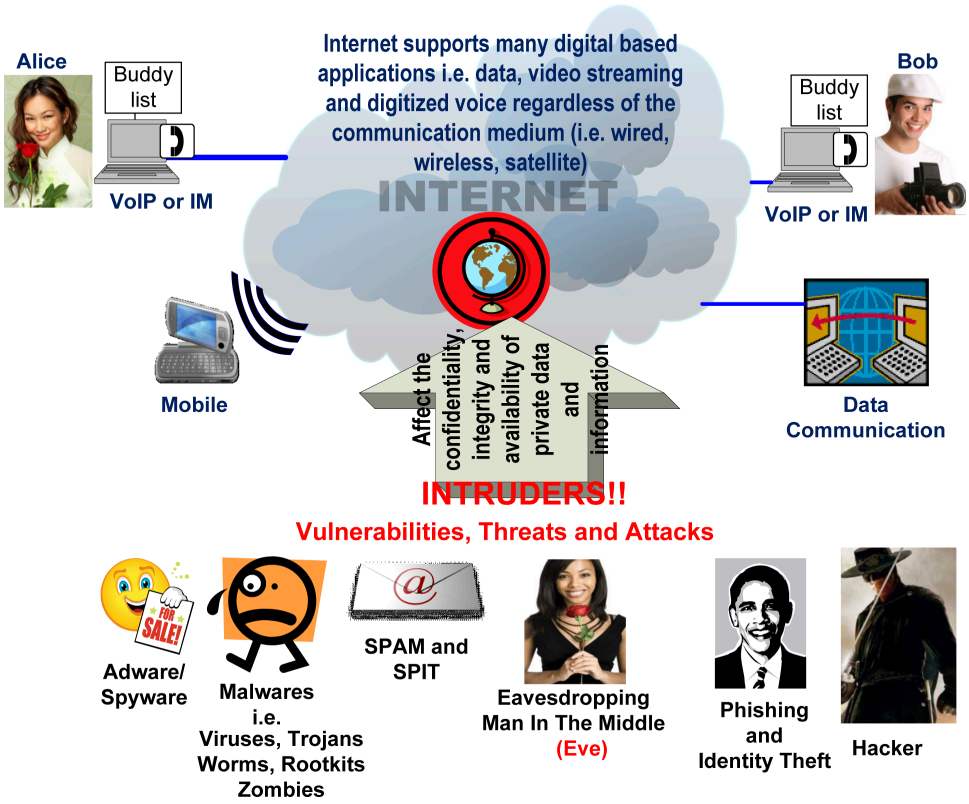


Figure 2.3: Internet Challenges and Opportunities

vices still needs improvement [12]. Although packet-based and unmanaged networks like the Internet are ideal for delivering data such as e-mail or static Web traffic, the public Internet's best-effort service is less suitable for delivering interactive applications and time-sensitive data such as a voice or a video streaming [110], similarly the Internet security needs are intensified with the introduction of VoIP services in order to protect the two invaluable assets: data and voice [91].

Figure 2.2 on page 29 shows the possibility of connecting both networks through gateways. A gateway is a network device used to connect networks of different platforms (i.e. architectures, infrastructures, topologies, technologies, codec types and environments). It is required as an intermediate device to perform path selection, packet switching and codec negotiation and transcoding if required. A gateway with a firewall is used to filter unwanted packets or data. The packets would be queueing at the gateway to be processed. In which case there would be a fixed delay due to the processing and variable delay that is caused by the queueing. The source of delays and their effects to the VoIP quality are explained in Subsection 6.1.3. For example, variable delays may cause voice packets to reach their destinations at unpredictable time. This may result in conversational difficulty.

2.1.2 Major Concerns For VoIP

Quality of service (QoS) and security are the two major concerns for the VoIP community. It is believed that increasing security mechanism would result in poor performance of VoIP services due to the additional processing of the security mechanism that would increase the overall one-way delay [107, 91, 70]. On the other hand, without security mechanism in proper places, VoIP services would be vulnerable and open to threats and attacks. VoIP architectures are sensitive to network level attacks, protocol level attacks as well as management[1] level attacks. An effective security mechanism is necessary for a secured VoIP environment. There was no major incidents like a Man-in-The-Middle (MiTM) attack or Denial of service (DoS) attack that could jeorpadise VoIP services [1]. Nevertheless, DoS attacks are still one of the main concern in VoIP. Mitigating DoS attacks on VoIP and other Internet applications are essential since most proxies and network equipments are reachable on the Internet [9]. DoS attacks especially DDoS are difficult to mitigate [34, 131, 84]. Several researchers have come up with solutions. For example, in order to minimize the DoS risk for servers, proxies or any service equipment, Battistello et al. have designed a security protocol (i.e. Denial-of-service Resistant Call Establishment Protocol) [9]. The protocol handles authentication and key agreement and aims at guaranteeing secure VoIP call establishment between interconnection proxies of different domains. Another type of attacks is the Spam over Internet Telephony (SPIT). Experts predict that VoIP operators will face problem with SPIT, a variant of an email spam attack [118, 114]. SPIT is more intrusive since it is generated by automated softwares. Unlike spam, the content of SPIT is hidden until the recipient of the call answers the call and the disruption is done [118]. Since 2009, there are several solutions suggested to mitigate DoS and SPIT attacks, especially on SIP based VoIP systems [83, 131, 114].

Securing and providing good quality services is nontrivial because VoIP calls are treated just like any other applications on the best-effort Internet even though VoIP is a time sensitive application [174]. The perceived voice quality is compromised, due to either lost packets or excessive delay or jitter or combinations of these factors. There are many different types of VoIP and IM to choose from. Currently the performance and behaviour of each service over secure VoIP networks are not well documented. However, there are several efforts on understanding the behaviour of Skype [45, 88, 44] and SIP [15, 149, 186].

2.1.3 Summary

The section gives the background of VoIP technology and its benefits to VoIP users, companies IT administrators and VoIP service providers. Tangible benefits like reduction in long-distance

[1] All the activities, methods, procedures, and tools that pertain to the operation, administration, maintenance, and provisioning of network resources.

calls has attracted its users. On top of that, the overall reduction in network maintenance cost might be the reason for a company to implement VoIP. Other benefits include intangible benefits like ease of maintenance and fast deployment over existing company network. There are many VoIP clients in the market. However, the performance and the behaviour of some of these clients over secure VoIP networks are yet to be tested.

2.2 VoIP Protocols

VoIP turns the sender's analogue speech into a digital representation and then transports the signal into IP packets to the recipient. In order to be able to do that a set of communication protocols is required to ensure a reliable interchange of data over a best-effort communication channel. In a best-effort delivery system, there is no guarantee that the data is delivered to its destination within a specified time limit. There is a possibility that data are lost during the transaction. The performance demands of VoIP mean that an ordinary data network software and hardware must be improved to support VoIP services. Many security issues are associated with still-evolving VoIP technology, so it is difficult to develop a complete picture of what a mature worldwide VoIP network will look like one day. Nevertheless, there are currently many different architectures and protocols to choose from and eventually a true standard will emerge. One of the current protocols is through TCP/IP protocols [91].

At the moment, most VoIP related protocols are assigned by the IETF for Internet communications, and the IEEE and ISO, for other types. The ITU-T handles telecommunications protocols and formats for PSTN. The ITU-R handles protocols and formats for radio communications like WiMAX [33]. The different sets of standards are also being driven towards technological convergence for the PSTN, radio systems, and Internet. These technologies are inseparable and the services they offer complement one another. Ideally, communication protocols are specified in such a way, that engineers, designers and software developers can implement them without difficulty. One way to establish this objective is through protocol layering [128]. Layering enables a mix-and-match of protocols that permit familiar protocols to be adapted to support new services. Figure 2.4 shows a typical VoIP protocol suite. The functions of the signalling and media control protocols are separated. In general, a signalling protocol controls a call setup, establishment and tear down. A media control protocol controls on how audio signals are being transmited, received and processed at the end points. It may also negotiate on the codec type and on other resources like bandwidth, CPU and memory of intermediate systems to carry out the task.

Figure 2.5 on page 33 shows how a digitized voice is carried over UDP/IP protocol. First, the voice data are divided into small chunks. Then each chunk is encapsulated with a RTP header before it leaves Application Layer to Transport Layer to form a voice data segment.

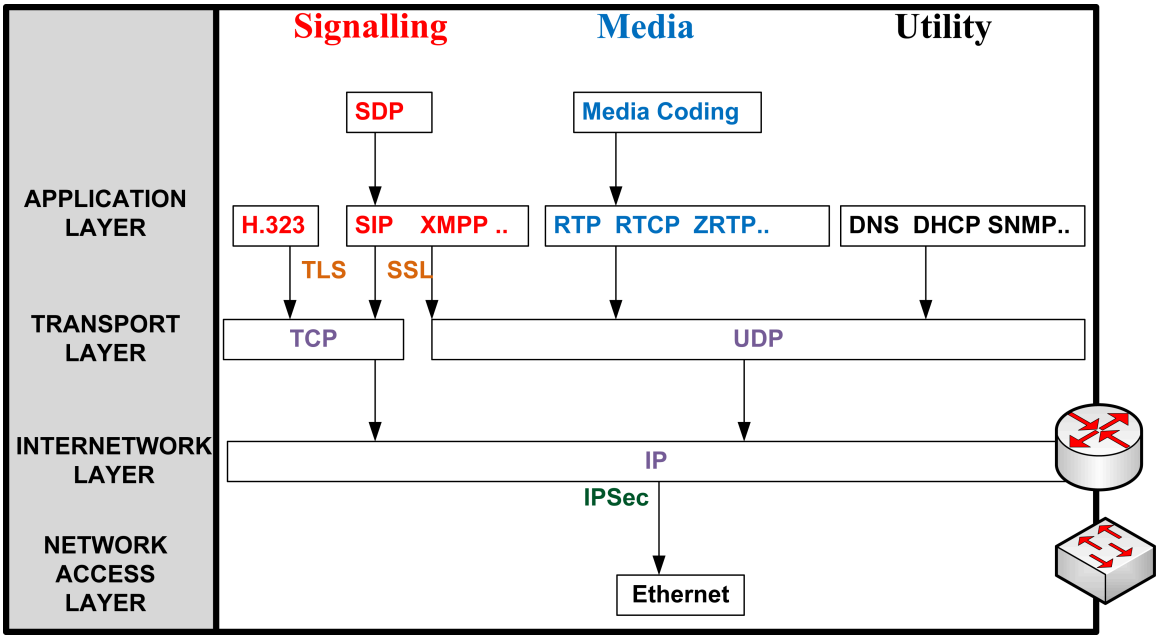


Figure 2.4: VoIP Protocol Suite

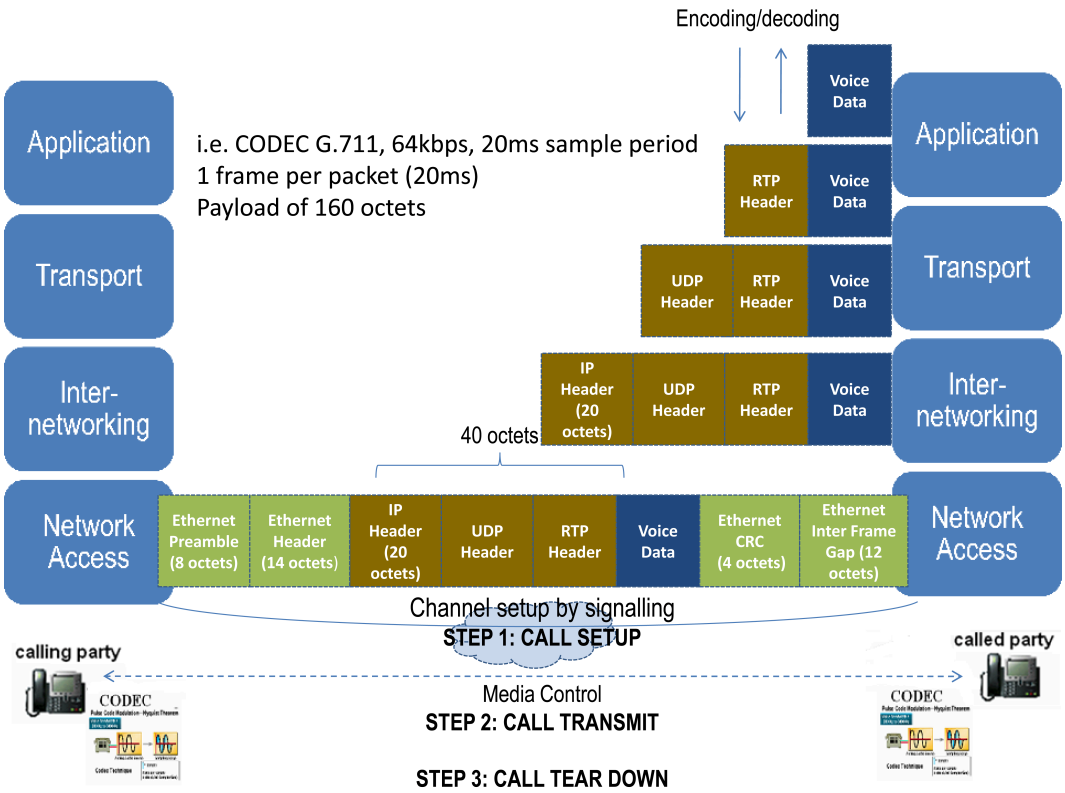


Figure 2.5: Voice Data Processing of VoIP

Then the segment is encapsulated with a UDP header before it leaves Transport Layer to Internetworking Layer to form an IP packet. Next the IP packet is encapsulated with an IP header before it leaves Internetworking Layer to Network Access Layer to form a data frame. The data frame is changed into bits of 0 and 1 pattern and is sent through Physical Layer and over several other IP-Based clouds to a destination. At the other end, the process is reversed whereby bits of 0 and 1 pattern is changed back to form a frame then to a packet then to a segment. The process is repeated for several other segments. Then the voice segments are assembled to form a complete stream if there is no error or no packet lost along the way. As described above a voice data is carried by a RTP protocol over UDP which is connectionless and best-effort. Therefore there is no retransmission for any voice data lost. During conversation difficulty, a user repeats his sentence to be understood or he may terminate the call and then redial to establish a new connection. There are several factors that affect the making of a high-quality VoIP call. These factors include the speech codec, packetization, packet loss rate, delay, delay variation, and the network architecture [55]. This section focuses on the primary factors that are neccessary in making a successful VoIP call which include the call setup signalling protocol, call admission control, security, and the ability to traverse NAT and firewall. Consequently several VoIP protocols have been chosen for this study. They are clasified into Open Standard and Proprietary protocols. SIP and H.248 are examples of signalling protocol available in Open Standard. XMPP is a protocol for Instant Messaging and RTP and RTCP are protocols for call transfers that are also categorized in the Open Standard [10, 150]. Skype P2P, Cisco Skinny, Yahoo Messenger (YMSG) and IAX open source protocols are examples for proprietary protocols. The important features and characteristics of each protocol are highlighted as depicted in Figure 2.6 on page 35. These features are extracted from RFC documents and H.Series documents for those under IETF and ITU-T standards respectively. For the proprietary protocols, books or other resources including online documents were referred to. These characteristics are elaborated in Figure 2.7 on page 36 and Figure 2.8 on page 37.

2.2.1 SIP

SIP is a standard under the IETF [143]. Since the beginning, Internet Community prefers SIP to H.323 protocol because its architecture is less complex and due to the fact that it has been adopted by various standardisation organisations as the protocol for both wireline and wireless world in the Next Generation Networks era [52]. It is a signalling protocol between user agents and one or more proxy servers. However, SIP does not provide the service rather it provides primitives that can be used to implement different services. The main purpose of SIP is to initiate, modify and terminate sessions between two (or more) Internet end entities. There

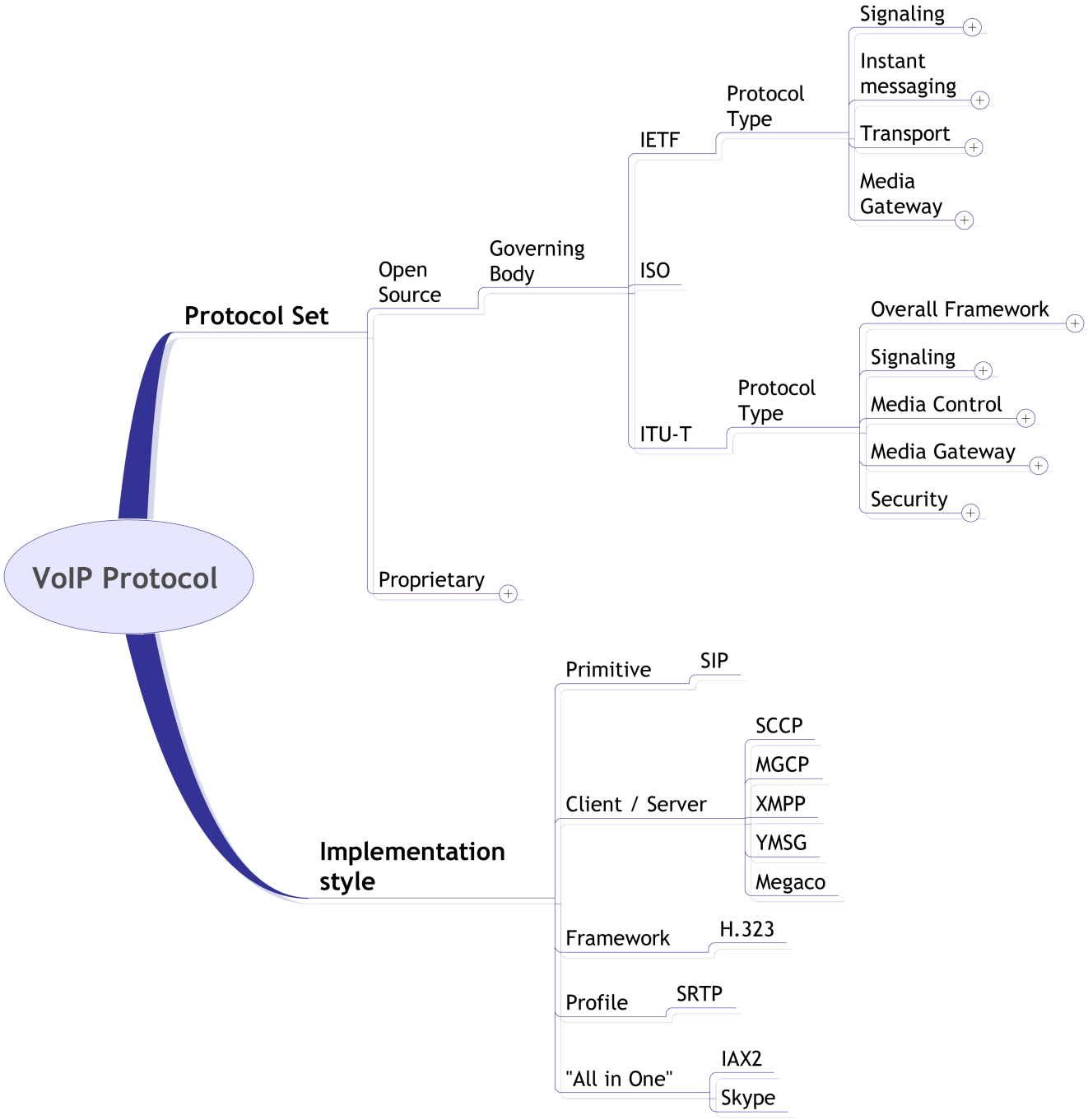


Figure 2.6: VoIP Protocols

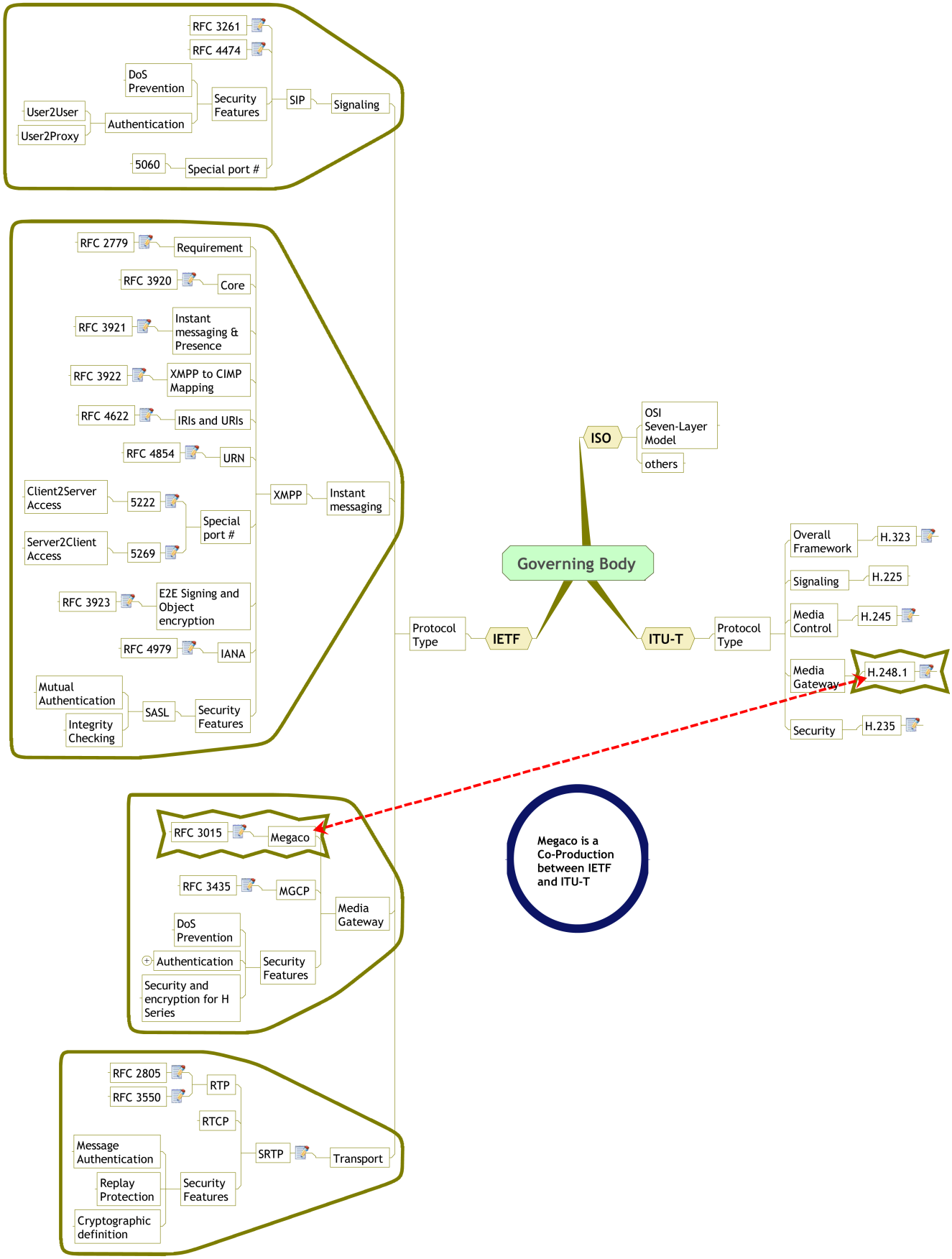


Figure 2.7: Open Source protocols

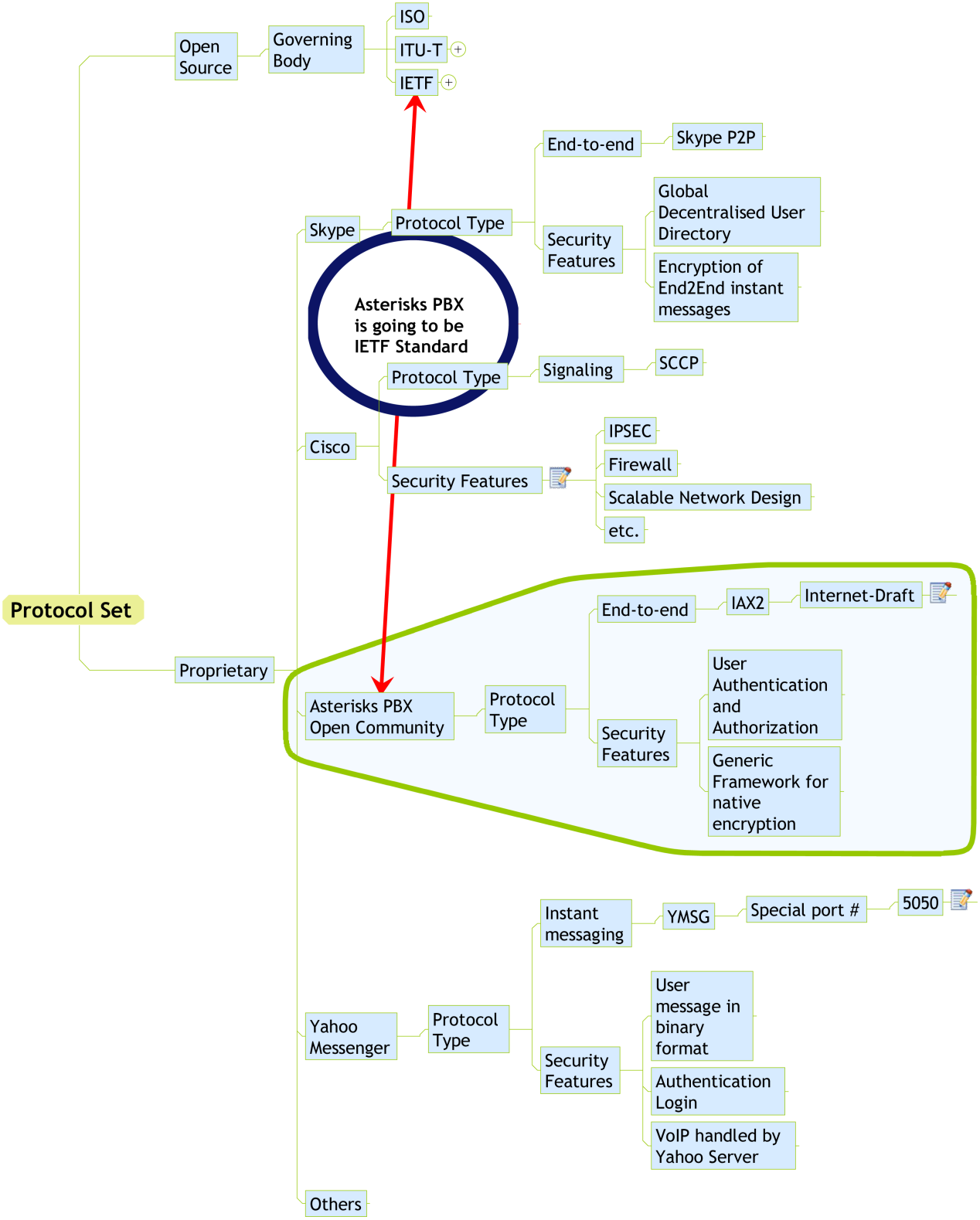


Figure 2.8: Proprietary protocols

are several types of entities that are defined in SIP: user agents, proxy servers, redirect servers, and registrar servers. The proxy server is an intermediary entity that acts as both a server and a client for making requests on behalf of other clients. The registrar is a particular server that accepts user registration requests. The redirect server accepts requests and replies to the client and provide a contact address for the called user [147]. SIP itself is independent of the type or characteristics of the session and handles the session description as an opaque body. The actual session description is handled by a companion protocol, SDP [64]. SDP stands for Session Description Protocol. It is used to describe multimedia sessions in a format understood by the participants over a network. For example, depending on this description a party decides whether to join a conference or when or how to join a conference. SIP is based on the use of textual messages, that are very similar to HTTP messages, aimed at session management and parameters negotiation among SIP clients through SDP [157]. Nevertheless, SIP messages may contain information that a user or a server wishes to keep private. However, the open and distributed nature of the IP telephony architectures turn the establishment of a secure environment for SIP into an extremely difficult task [52]. Notwithstanding, SIP provides its own suite of security which includes Denial of Service prevention and authentication for User-to-User and User-to-Proxy at both ends. SIP supports both IPv4 and IPv6 network addresses.

2.2.2 RTP

RTP is a transport protocol for real-time applications. RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. RTP does not address resource reservation and does not guarantee quality of service for real-time services. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner scalable to large multicast networks, and to provide minimal control and identification functionality. RTP and RTCP are designed to be independent of the underlying transport and network layers.

SRTP is a profile of RTP and RTCP. It provides confidentiality, message authentication and replay protection to the RTP traffic and to RTCP to control RTP traffic. SRTP defines a set of default cryptographic transforms and it allows a new cryptographic transformation to be introduced in the future. It is suitable for a heterogeneous environment, hence, SRTP works well with different types of signalling protocols. RTP and SRTP are under IETF standardization [10, 150].

2.2.3 H.323

H.323 is a framework for various pieces of protocols that fit together. H.225 defines call signalling between endpoints and gateway. RTP/RTCP is used for transmit real time data over IP networks. H.248.1 handles gateways and gatekeepers. H.245 controls establishment and closure of media channels within the context of a call and to perform conference control. H.235 is a recommendation for Security and encryption for H Series. H.323 is a standard under ITU-T. It was the first Voice over IP standard [75].

Dalgic and Fang compared the functionality, QoS, scalability, flexibility, interoperability and ease of implementation of H.323 and SIP [36]. H.323 version 2 and SIP are very similar in term of functionalities and services that they can support. H.323 has fewer interoperability issues since the supplementary services in H.323 are more rigorously defined. H.323 has taken more steps to ensure compatibility among its different versions, and to interoperate with PSTN. The two protocols are comparable in their QoS support for example similar call setup delays, no support for resource reservation or Class of Service (CoS) setting, however, H.323 version 3 allows signalling of a requested CoS.

SIP's main advantages are flexibility to add new features, and relative ease of implementation and debugging. H.323 and SIP are improving themselves by learning from each other, and the differences between them are narrowing with each new version [36].

2.2.4 MGCP

MGCP is a media gateway protocol. It is a master/slave protocol where the gateways are expected to execute commands that are sent by Call Agents within a distributed system. It is a standard under IETF [6].

2.2.5 Megaco

Megaco is another protocol similar to MGCP. However, it was a co-production between IETF and ITU-T [60, 35]. Both protocols abide to the guidelines of the API Media Gateway Control Protocol Architecture and Requirements[2] [59].

2.2.6 XMPP

XMPP provides a generalized extensible framework for exchanging XML data. It is used mainly for building instant messaging and presence applications that meet the requirements of Instant Messaging / Presence Protocol Requirements [37]. To date it has been implemented

[2] RFC2805

via a client-server architecture whereby a client utilizing XMPP accesses a server over a TCP connection via port 5222 and servers also communicate with each other over TCP connections via port 5269. XMPP includes a method for authentication of a stream by means of an XMPP-specific profile Simple Authentication and Security Layer (SASL) protocol. XMPP must support high security that is to provide mutual authentication and integrity-checking. XMPP is a standard under the IETF.

Currently XMPP stanzas such as Jingle negotiation messages and service discovery exchanges are not encrypted or signed. As a result, it is possible for an attacker to intercept these stanzas and modify them, thus convincing one party that the other party does not support XTLS and therefore denying the parties an opportunity to use XTLS. This is a more general problem with XMPP technologies and needs to be addressed at the core XMPP layer.

2.2.7 Skype

Skype is a proprietary P2P telephony network. Skype employed special techniques to deliver state-of-the-art-IP-based telephony; Firewall and NAT traversal; Global decentralized user directory; Intelligent Routing; Encryption of end-to-end instant messages; Supernode and Simple User Interface [146].

As a proprietary protocol, any events that happen behind the scenes are beyond control of the user or application deployer. Therefore Skype users are advised to apply security patches regularly. Skype as a host program scans ports and IP addresses to identify if it is behind a firewall or NAT devices. This action is believed to deliberately weaken the security of any corporate firewalls because pinholes are created to allow communication through. Skype also depends on the whole of Internet connectivity for efficient use of the Internet's bandwidth and processing power. It uses a network of supernodes, that can act as proxy servers for hosts that are behind firewalls. Supernodes are selected among peers with large computational power and good connectivity in terms of bandwidth, uptime and absence of firewalls [13]. Skype can promote any of its clients to supernode status without the user knowledge. As a result, a voice call can go through an unexpected path and through a risky and uncontrolled proxy server. On top of that an enterprise that does not deploy any firewalls, could find that its Skype clients are promoted to supernodes [30].

Bonfiglio et al. discovered that Skype adopted different types of voice codecs [13]. Some are Constant Bitrate (CBR), while others are Variable Bitrate (VBR) codecs. Skype reacted differently and changed its behaviours according to network conditions. The algorithm used by Skype to perform selection between different codecs was unknown. All codecs were standard except for ISAC, a proprietary solution of GlobalIPSound. ISAC was the preferred codec for end-to-end calls, while the G.729 codec was preferred for Skypeout (i.e. any call involving

a Skype peer and a PSTN terminal) calls. Skype detected the change of network conditions based on several parameters. For example, Codec Rate (i.e. the bitrate used by the source) and Redundancy Factor (i.e. the number of past blocks that Skype retransmits, independently from the adopted codec, along with the current block) were the preferred indicators used by Skype before responding to the network conditions change. Other than that, Skype message framing time (i.e. time elapsed between two subsequent Skype frames) was frequently modified as well.

2.2.8 Skinny Client Control Protocol (SCCP)

SCCP is a Cisco proprietary protocol for communication between Cisco IP phones and Call-Manager Server. In this model, all the intelligence is built into the server, which is a Call Manager in the Cisco IP Telephony solution. The client, which is a Cisco IP phone, has minimal intelligence. In other words, the Cisco IP phones have to do less work, thus requiring less memory and processing power. The Call Manager, being the intelligent server, learns client capabilities, controls call establishment, clears calls, sends notify signals e.g. message waiting indication (MWI), reacts to signals from the client after the user presses the directory button on the phone [132].

2.2.9 YMSG

YMSG is a proprietary network protocol used by the Yahoo! Messenger instant messaging client. Communication is between client application and a server over TCP/IP on a default port 5050 or others. User messages are sent in binary format in which text portions of the data are transmitted in plain view. Hence, in this case, part of the message is exposed and unprotected. However, authentication is on the login details. Different protocols are used to transfer different data types on the Internet. File transfer, JPEG are via HTTP. Chat room categories, rooms and lobbies are retrieved using HTTP as XML documents. VoIP is handled indirectly by Yahoo server to avoid use by unauthorized users [183]. YMSG also supports PC-to-PC voice and conference calls between friends. YMSG cannot run behind corporate proxy servers and firewalls. However, YMSG works behind most PAT and NAT.

2.2.10 Inter-Asterisk eXchange version 2 (IAX2)

IAX2 is a P2P application-layer control and media protocol for creating, modifying and terminating multimedia sessions over IP networks. Unlike Skype it was developed by the open source community for the Asterisk PBX for VoIP call control. It is proprietary, but open [40]. It is extremely flexible for other type of streaming media. It is an all in one protocol

that combines both control and media services in the same protocol. IAX2 supports security features by allowing multiple methods of user authentication and authorization during peer registration. It also supply generic framework for native encryption [156].

2.2.11 Summary

The VoIP protocol types, implementation styles, protocol stack that they are in, security features and protocols that are related of the selected protocols are observed. The survival of each protocol could not be predicted, however, the trend is on the standard recommended by the Internet community, IETF. For building instant messaging and presence applications, XMPP might be the winner especially as it is used by Google in Google Talk Service. For signalling, SIP is preferred over H.323 by the IETF community. H.323, however, was predominance in the 90s. Skype P2P is believed to be much superior in term of its architecture, however, enterprises requiring total control and administration of its network might find it is a less desirable choice. MGCP, Megaco, H.248.1 may be used as standards for media gateway. For transport, TCP and UDP are the two most obvious choices.

IAX2 is an alternative end-to-end connection especially for Asterisk PBX community. It has overcome the problem of intermediary NAT and firewalls by using the same stream for data and control. In addition IAX2 may soon become a standard under IETF. An Internet-Draft April 2007 was sent for IETF review.

So far the existing protocols seem sufficient to handle various part of end-to-end VoIP communications. Nevertheless, since the technology is relatively new, the protocols are still evolving and one standard that govern both IETF and ITU-T is still far away. In which case it brings opportunities (i.e. more advanced features than PSTN) and security risk at the same time. Unfortunately, until then, VoIP vendors will have to cater for both standard within their hardwares and softwares. This give researchers an opportunity to explore further on how to improve the security of VoIP network and to improve VoIP quality of service.

2.3 VoIP Security Issues

David Endler, chairman of the VoIP Security Alliance aptly argues that the VoIP security issues affect not only carriers, service providers and enterprises but also consumers. Currently, technologies like instant messaging and web services also started to feel its detrimental effects. The problems though should be addressed by all parties involved. They are too complicated to be solved by one party alone because every party is only responsible for a part of the problem [58]. To appreciate the vulnerabilities and threats that are awaiting to emerge, one needs to understand the threats and vulnerabilities of computer systems and networking. Most

computers are connected with the rest of the world through the Internet and are therefore vulnerable to attack.

Though it is rare, it is not impossible for someone to steal one's identity and perform transactions on one's behalf. Friends, business partners or even banks might think that they are communicating with an authorised individual. Whatever white collar crimes that can happen in the real world, they will give bigger impact in the virtual world since Internet has no boundary. Somebody from Singapore for example might be breaking a law in China during his visit in Malaysia. Computer Security was introduced in a later stage of the World Wide Web development during the late 80s and early 90s. Identification, Authentication, Authorization and Accountability are the parameters to ensure that only authorized personnel can get access to authorized resources.

Vulnerabilities and threats that relate to a computer network soon will be threatening VoIP services, after all VoIP is just another application on IP Network. This is consistently expressed in [166, 82, 47]. According to Lehtinen, vulnerabilities to computer networks include physical and natural disaster, on hardware and software, media, emanation[3], communication and human [98]. Natural and physical disaster, unintentional, and intentional attacks are threats to computer networks. Foreign intelligence agents, terrorists, criminals, corporate raiders and crackers are few examples of outsiders that might try to steal unauthorized information from the computer networks.

In addition, the computer networks are exposed to the threats from the company employees, former employees or contractors. For example, disgruntled employees impose high threats, because they are authorised to access the company information through legitimate means [133]. Their plans to sabotage the network could be executed undetected until the damages have been done. The statistics for the security research done between 1980s to 2000 for 13 incidents correlated with traditional IT world at the time whereby 70% of security breaches were carried out by insiders [19]. Another study done between 2001 and 2003 showed that 70% of all events came from external, indicating a significant change in threat source since then due to the increase in global connectivity at this point in time and onwards. For example, the Internet users are growing at a good rate, however, the growth rate is not the same all over the world. There are approximately two-billions Internet users whereby the majority of these users are from Asia [158]. Further study on "point of entry" on 14 internal incidents and 25 external incidents showed that majority (i.e.~36%) of the external security incidents are from Internet and majority (i.e.~43%) of the internal security incidents are from business networks. This shows that there is high possibility for the vulnerabilities and threats from internal and external sources to interfere with the VoIP services because they are closely linked

[3]Emits electrical and electronic radiation

to the Internet and business networks.

Fortunately there are several ways to mitigate these vulnerabilities and threats although some of these methods may put the quality of service and performance of the system at risk [120]. Computer Security, Communication Security and Physical Security are a few counter measures to mitigate these vulnerabilities and threats. Everybody in an organization is accountable to the well-being of the organization computer network and IT Assets. For example, Computer Security is the responsibility of the company System Administrator. They should be given appropriate authority to grant the right access controls to the right personnel at the right time based on their roles in the company. On top of that the company top management supports are important in order to ensure that every employee abide to the company IT policies. There are several access control models available. Role-Based Access Control can be configured to enforce Mandatory and Discretionary Access Control policies. Discretionary Access Control and Mandatory Access Control are more traditional models. For Mandatory Access Control only one administrative role is assumed. For Discretionary Access Control, a more complex set of administrative roles is required. Role-based Access Control mechanisms are general enough to simulate both traditional methods [123]. On the other hand, Communication Security is the responsibility of the service provider. Physical Security is the responsibility of the building security and asset department of a company. Other employees must ensure that their IT Assets are well looked after. Voice over IP Security Alliance Public Release 1.0.24, October 2005 listed out VoIP Security and Privacy Threats, [166]. The threats are Social Threats, Eavesdropping, Call Pattern Tracking, Traffic Captures, Interception and Modification, Service Abuse, Intentional Interruption of Service, Physical Intrusion and other Interruption like power failure. Figure 2.9 on page 45 shows the relationship between the Internet Attacks and the VoIP Attacks.

Although there are many areas of concern, the thesis addresses the network layer security and the media transport security only but not on the social engineering attacks, the physical intrusion, the session security and other types of threats and vulnerabilities as depicted in Figure 2.9.

2.3.1 Social Threat

Every individual, enterprise, goverment and country has the right for the protection against social threats. Social threats consist of misrepresentations of identity, authority, right and contents, theft of service and receiving unwanted contacts and contents like spam of subjective or offensive contents, refer to Figure 2.11 on page 47.

Security and privacy are vital to social needs [166]. Planners should balance between Return on Investment (ROI) and convenience of service when design IP network. Several

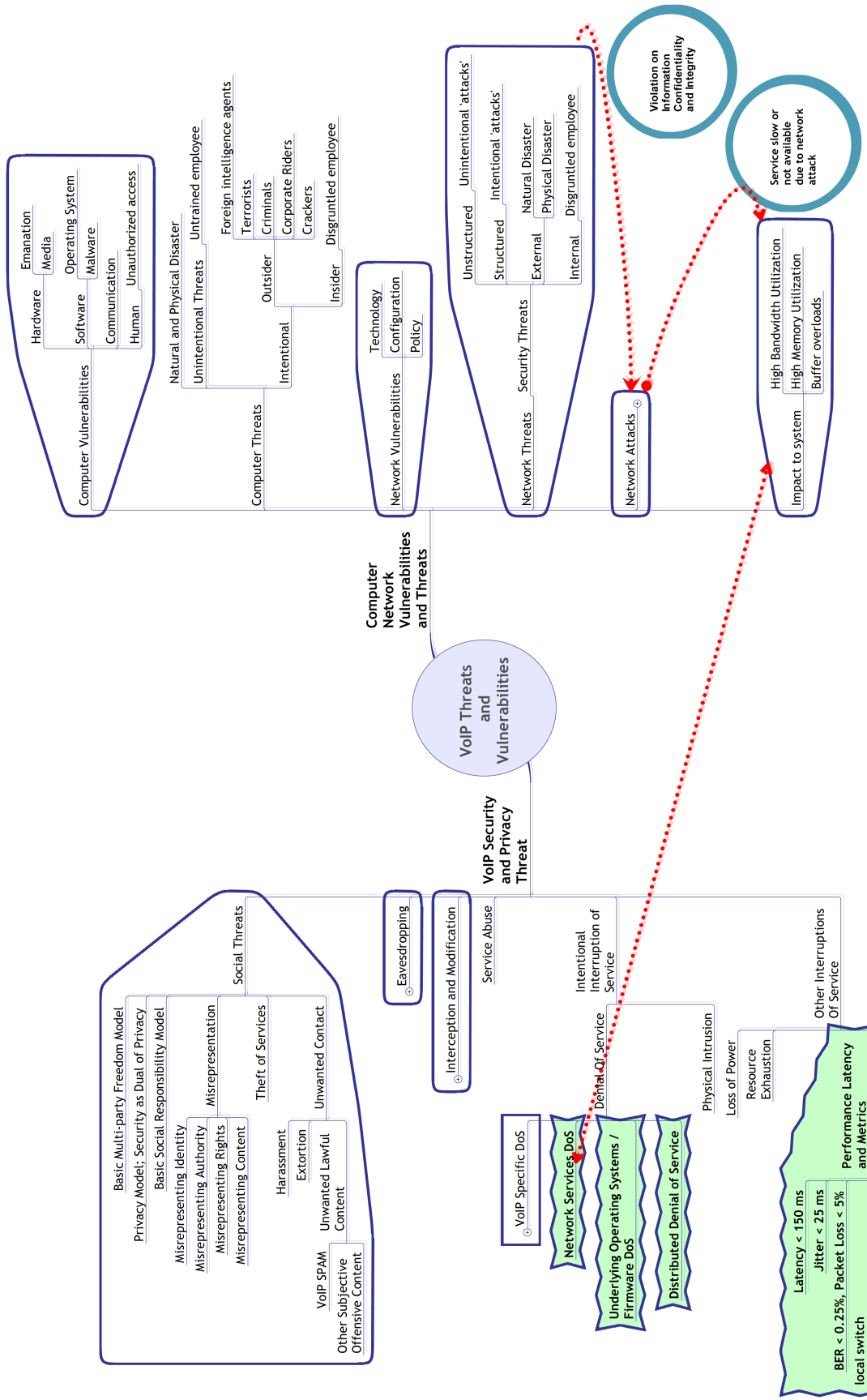


Figure 2.9: VoIP security issues

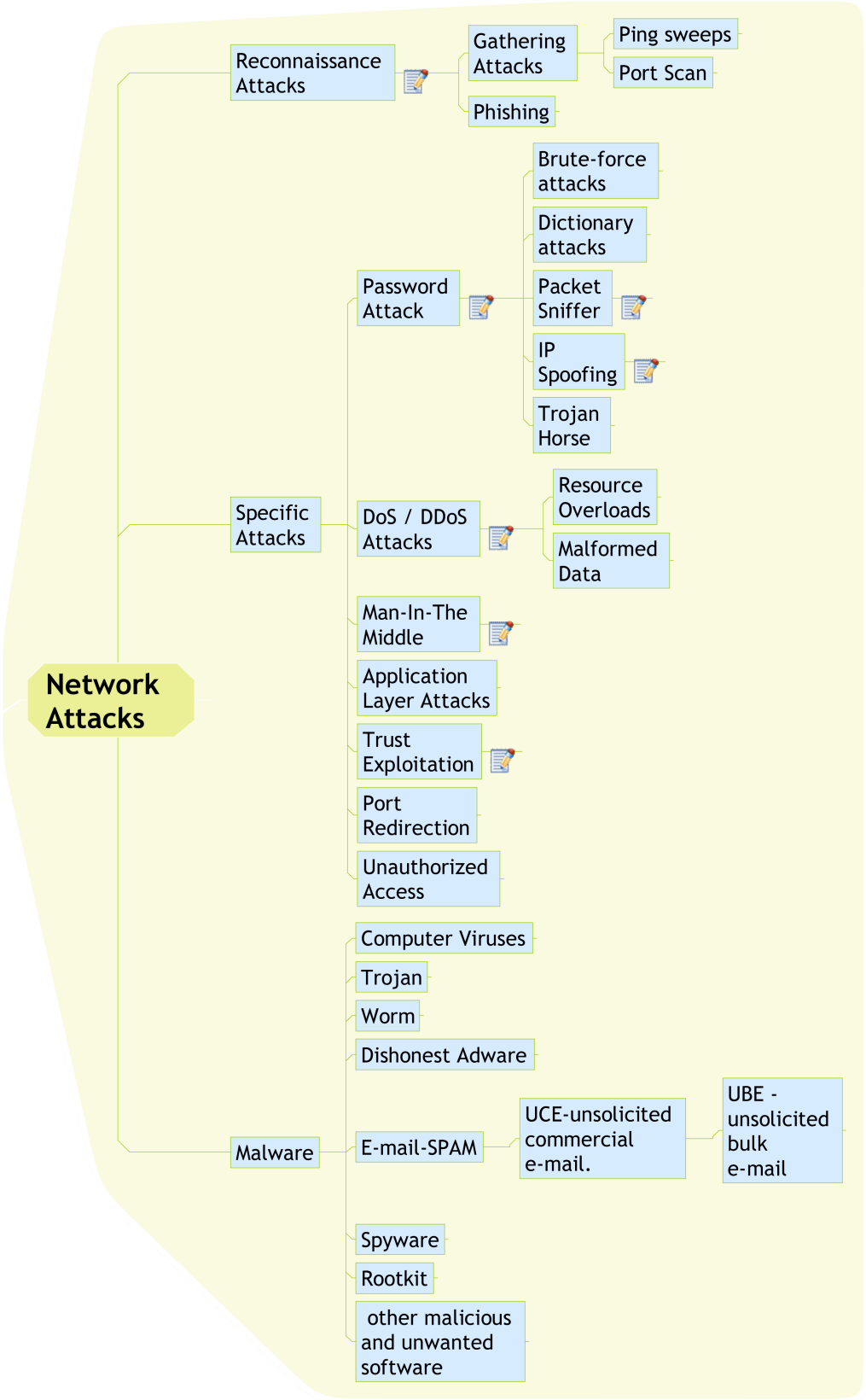


Figure 2.10: Network attacks

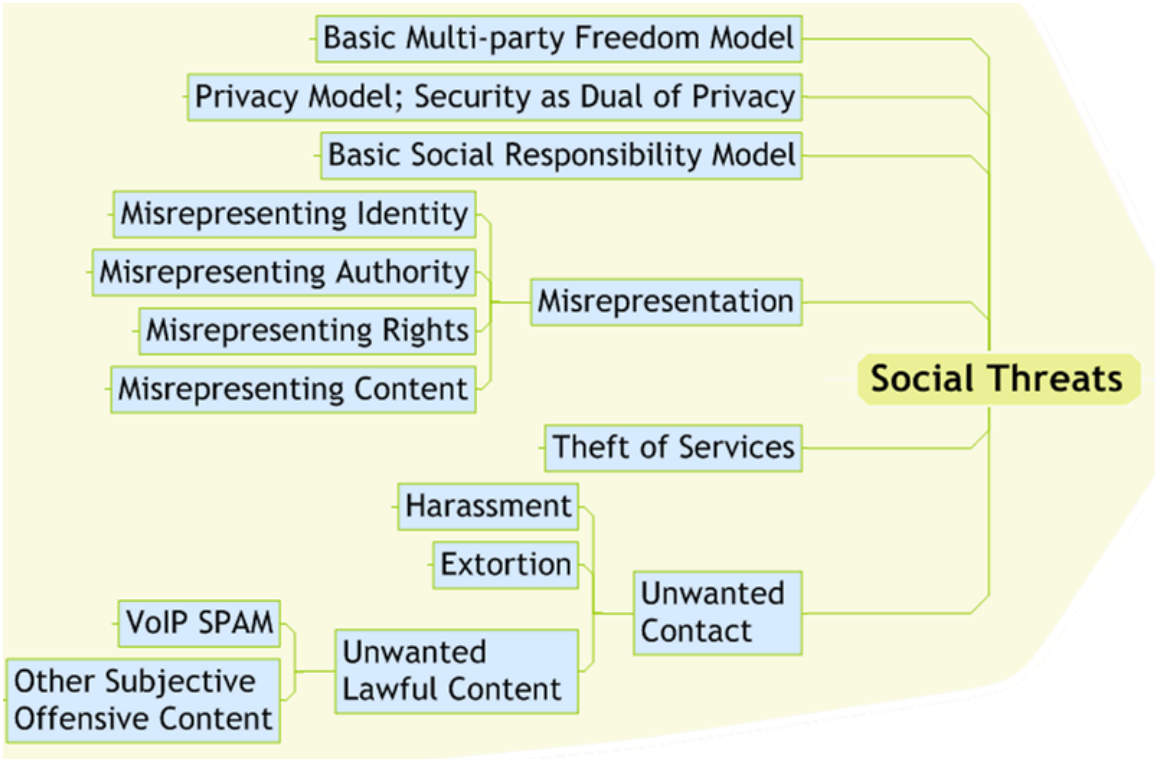


Figure 2.11: Social threats

models have been adopted in order to protect the privacy of any communications made within the ICT community. The Basic Multi-party Freedom Model, for example, allows role shifting between two or more parties that are involved in any interactive communications. In the Privacy Model, everybody has the privilege over his communication system, protection from unauthorized access, interruption, delay or modification. Security is used as a medium to protect the right to have privacy. The Basic Social Responsibility Model determines the social responsibility by examined both the intention and the impact of a person's conduct before any consideration being made on what a system should do either to deny, tolerate or permit. Sometimes intentional and external control or interruption is justified because it adheres to other more pressing social requirements, such as intercepting a communication system to assist in a rescue or prevent a catastrophe.

2.3.2 Eavesdropping

An eavesdropping attack is defined as a method by which an attacker is able to capture the entire signalling and/or data stream between several VoIP endpoints, however, the attacker cannot or does not change the data itself [166]. The attacker, however, can reconstruct new sentence from the data that he has gained earlier. In general, eavesdropping attacks are possible in shared media such as Ethernet and wireless networks. The attacker configures the respective

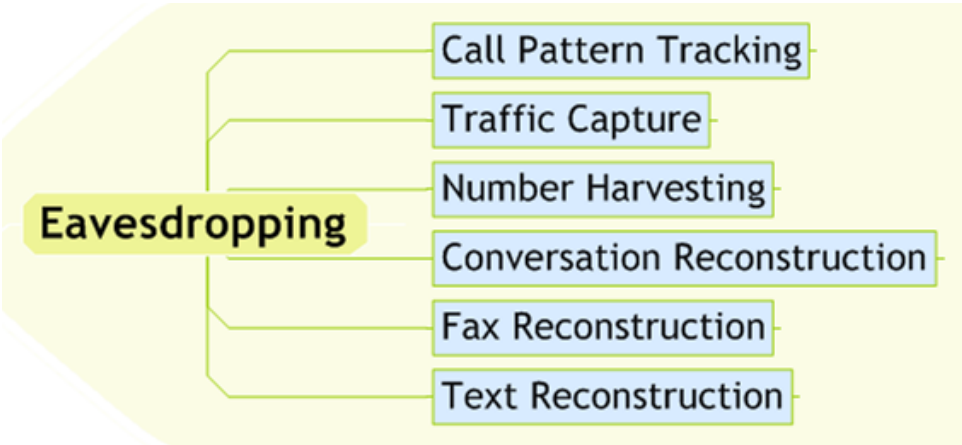


Figure 2.12: Eavesdropping

network interface in promiscuous mode. In this mode, the attacker's computer graps any packets sent on the network. If packets are unencrypted, the attacker can read the credentials data including password [180]. Eavesdropping tools are easily available, for example tcpdump and Wireshark [148]. In the context of VoIP, eavesdropping can be divided into call pattern tracking, traffic capture, number harvesting and conversation, voice mail, fax, video or text reconstructions, refer to Figure 2.12 on page 48. Wireshark and tcpdump can still be used to sniff the VoIP packets. On top of that a microphone and an audio recorder might be used to record any conversation between the two communicating channels.

In networks that do not use shared media or where packets are encrypted, an attacker may be able to use a MitM attack to intercept communication between a client and a server. MitM is a form of active eavesdropping where the entire conversation is controlled by the attacker. By impersonating the server or an intermediary system, the attacker may be able to fool the client into connecting with the attacker rather than the server. The attacker can then capture the client's credentials including the client's identity and password. The attacker uses those credentials to connect to the server, impersonating the client. In order to make the communication appear normal, the attacker relays packets between the client and the server. Nonetheless the attacker can read, modify, inject, or drop any packet, even if the client and the server authenticate and encrypt all packets [181]. MitM attacks compromise the integrity of the data sent between any two communicating channels [92]. In general, most of the SIP messages exchanged during a call-setup phase are not authenticated, and usually the communicating parties do not apply source control [157]. On the same account, H.323 can also suffer from MitM attacks.

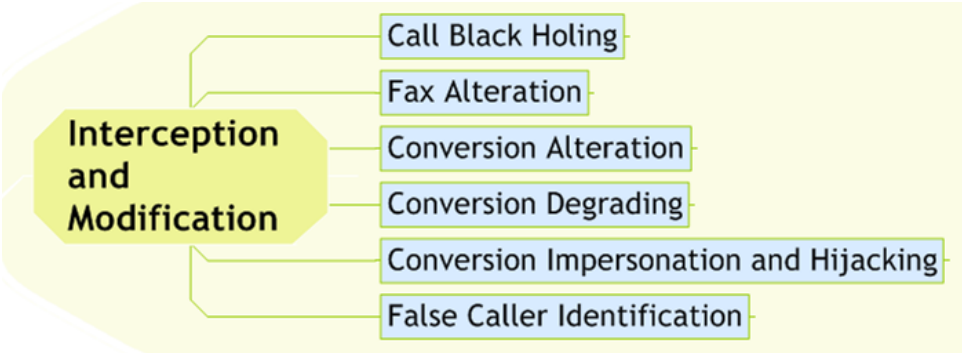


Figure 2.13: Interception and modification

2.3.3 Interception and Modification

In these class of attacks, an attacker can see the whole signalling and data stream between two endpoints just like eavesdropping. However, the attacker can also modify the traffic as an intermediary in the conversation [166]. Interception and modification consist of call black holing[4], call rerouting, fax or conversation alterations, conversation degrading, conversation impersonation and hijacking, and false caller identification, refer to Figure 2.13 on page 49.

2.3.4 Intentional Interruption of Service

Intentional interruption of a service includes Denial of Services (DoS) and physical intrusion. In particular, the VoIP Security Alliance categorizes denial of services into VoIP specific DoS, network services DoS, DoS attack on operating system or firmware and Distributed Denial of Service (DDoS) [166], refer to Figure 2.14 on page 50. Unlike social threats, eavesdropping, interception and modification attacks, this attack can disrupt the Internet system which affects all related Internet services.

There exist several taxonomies of DoS attacks. For example, Farraposo et al. give a classification of DoS attacks that are associated with the TCP/IP suite and distinquish this classification with some other taxonomies in order to assist them in understanding the similarities and differences in DoS attacks and the scope of the DoS problem [48]. Most of the attacks use illegitimate packets, however, some techniques allow attackers to launch their attacks with legitimate packets. Most DoS attacks on TCP/IP suite exploit the weaknesses that existed in the TCP/IP protocol and architecture. For example, a flooding attack is launch by creating a packet storm against the victim and an exploitation attack exploits TCP vulnerabilities during connect phase by manipulating TCP/IP flags (SYN, ACK, RST and FIN) and timeouts.

In a similar interest, Peng et al. categorize denial of services into four different types of bandwidth attacks [127]. They are DoS, DDoS, Distributed Reflector DoS (DRDoS) and

[4]Unauthorized dropping

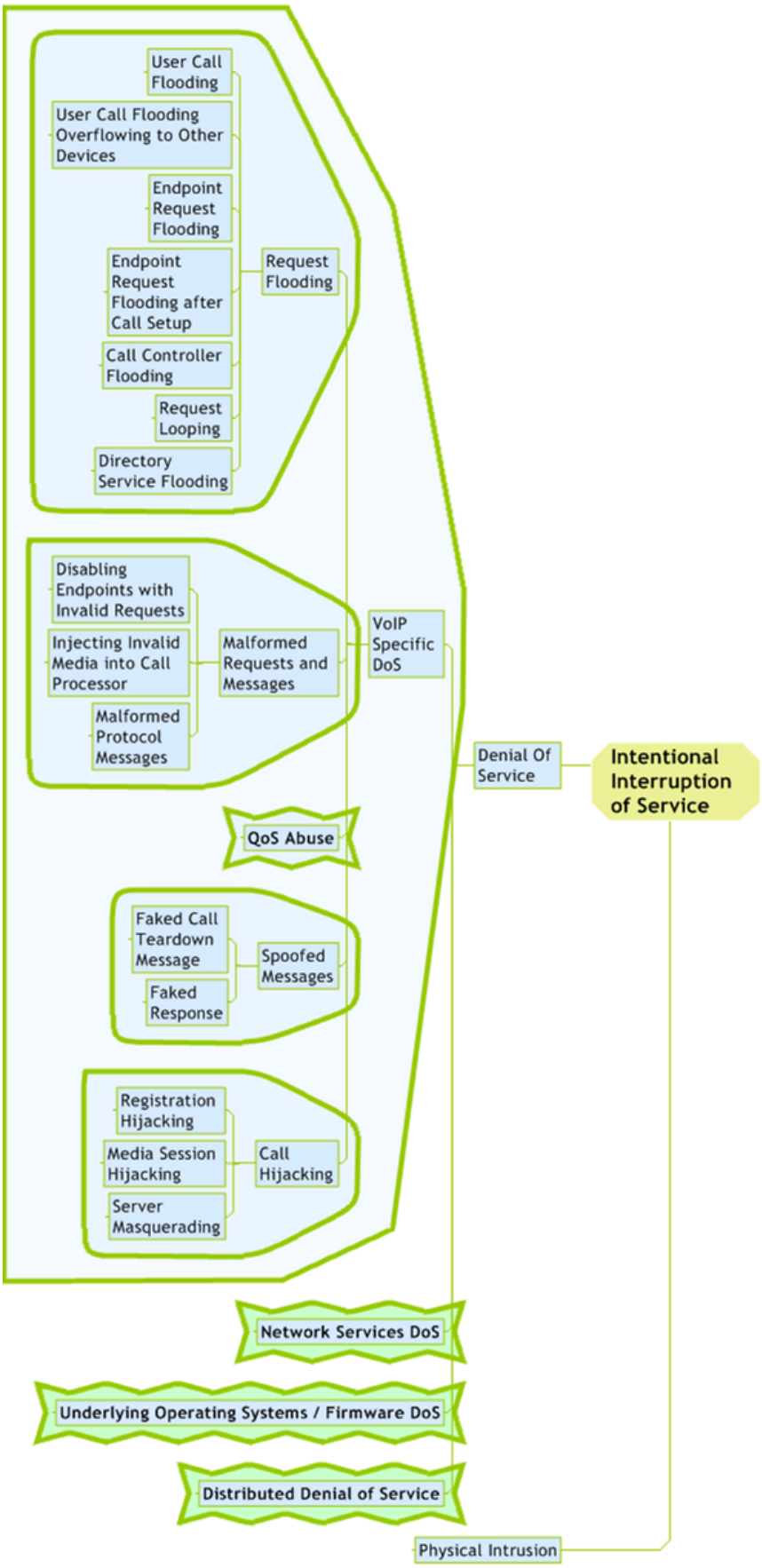


Figure 2.14: Intentional interruption of services

Table 2.1: Denial Of Services Attacks [127]

Type of Attack	Description	Methods Of Attack	Aims	Possible Target
Denial of Services (DoS)	DoS attacks generally achieve their goal by sending large volumes of packets arbitrarily similar to legitimate traffic. that occupy a significant proportion of available bandwidth. *Launch from a single or few hosts (i.e. less than 10)*	Protocol-Based Bandwidth Attacks: • SYN Flood • ICMP Flood	Consume critical resources in a network service to prevent legitimate users from accessing the service	*CPU and memory capacity in a server, stack space in network protocol software, or Internet link capacity*
Distributed DoS (DDoS)	Similar to DoS but attackers usually control *many hosts* to generate the attack traffics. DDoS is the most common Bandwidth attack. Typically contains two stages: Stage 1: Compromise vulnerable systems that are available in the Internet and install attack tools in these compromised systems. (i.e. turning computers into zombies) Stage 2: Attacker sends an attack command to the zombies through secure channel to launch a bandwidth attack against targeted victims.	Application-Based Bandwidth Attacks: • HTTP Flood • SIP Flood	Similar to DoS	Two important Internet Applications, namely World Wide Web and VoIP.
Distributed Reflector DoS (DRDoS) Attacks	DRDoS is considered to be a potent and increasingly prevalent Internet Attacks. DRDoS attacks have the ability to amplify the attack traffics. DRDoS attack in three stages: Stage 1: Compromise vulnerable systems that are available in the Internet and install attack tools in these compromised systems. (i.e. turning computers into zombies) Stage 2: Attackers instruct zombies to send any third parties spoofed traffic with the victim's IP address as the source IP address. Stage 3: The third parties will then send the reply traffic to the victim, which constitutes a DDoS attack.	Amplification Attack: • Domain Name Service (DNS) Amplification Attacks.	Similar to DDoS but obscure the sources of attack traffic by using third parties (routers, web servers) to relay attack to victims.	Web sites, DNS servers, Computers
Infrastructure Attacks	It is potentially catastrophic as the whole Internet may be affected.	Example: • Attack on DNS root servers. • Attack on core routers.	Disable the services critical components of the Internet.	All services that depended on Internet Infrastructure.

Infrastructure attacks, refer to table 2.1 on page 51. DDoS is the most common bandwidth attack. An attacker usually controls many hosts to generate the attack traffics. Once the target systems have been compromised, an attacker sends an attack command through secure channel to launch a bandwidth attack against targeted victims. In which case not only VoIP is affected, but data network as well. Fortunately these bandwidth attacks can be mitigated with the right filtering techniques like ingress/egress filtering, router-based packet filtering and source address validity enforcement protocol.

VoIP specific DoS includes request flooding, malformed requests and messages, QoS abuse, spoofed messages and call hijacking [166]. For example, refer to Table 2.1 on page 51, DDoS main targets are the World Wide Web and VoIP in particular the http and SIP protocols. Once affected, a computer becomes a zombie which then would launch another bandwidth attack to another victim. Request floodings are DoS attacks that involve overwhelming the target with a number of legitimate and/or invalid requests. For legitimate request flooding, the attacker must have a legitimate account with a SIP server. After logging in, the attacker may continuously request for a SIP service from a target. The requests would be processed normally but the target may appear busy to other users. This creates a DoS attack. However, it can be easily traced to the attacker. In invalid request flooding, attacker is not required to be properly authenticated by SIP server. An attacker overwhelms a SIP server by sending fake requests or messages. In this kind of DoS attack the attacker identity is hidden [26].

Implementation flaws of SIP systems also create opportunities for malformed requests and messages attacks. The vulnerabilities may be fixed through software patches as soon as they are found. These kind of DoS attacks include disabling endpoints with invalid requests, injecting invalid media into call processor, and malformed protocol messages. Spoofed messages cause the call processing system to be disrupted in a number of ways. For example, a faked call tear down distrupts services by causing a session to end prematurely, thus denying service to the users. A faked response like a line BUSY may deny a victim from receiving any incoming call. QoS abuse involves an attacker violating the QoS negotiated at setup, for example codec type. This would have the effect of introduced latency which affects voice quality during a call. Call hijacking includes registration hijacking, media session hijacking, and server masquerading. Hijacking occurs when some transactions of a VoIP Service are taken over by an attacker. The hijacked transactions may be signalling, media or both. For example, attacker spoofs a SIP Response redirecting the caller to a rogue SIP address and intercept the call. The attack leads to the service interruptions, thus denying service to the users.

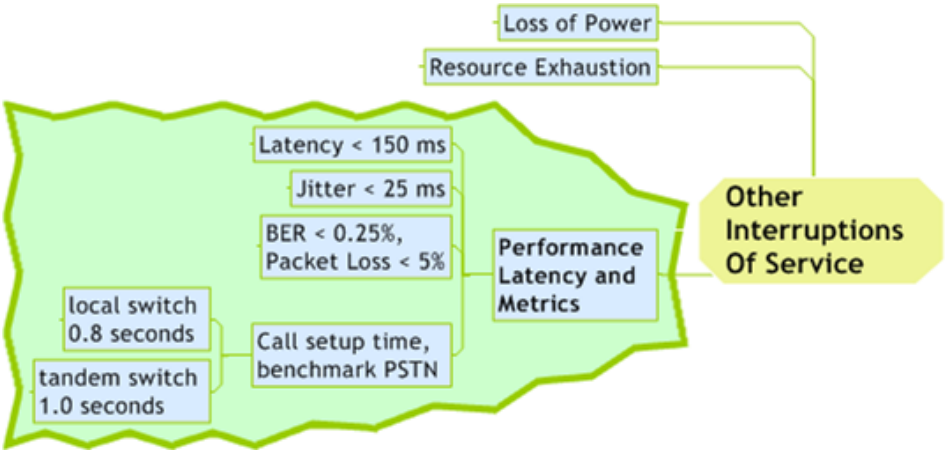


Figure 2.15: Other service interruptions

2.3.5 Service Abuse

Service abuse could be from a customer or an employee of an ISP or a third party. For example Premium Rate Service Fraud is a method whereby traffic is artificially increased for the purpose of maximise billing. Improper Bypass or Adjustment to Billing are method to avoid authorized service charges or for concealing other fraud by adjusting billing records. Other improper access to services include various forms of identity theft where legitimate credentials obtained without consent are exploited without concession of their rightful owner, manipulating internal access into authentication systems like VoIP gateways, switches and active directories and various method of concealing fraud by spreading access across numerous accounts to avoid exposure by fraud analytical analysis and reporting software [166].

2.3.6 Other Interruptions of service

Other than the intentional interruption of service, VoIP is also subjected to other types of service interruption. Among a few are: loss of power supply, resource exhaustion like CPU and stack overflow and packet delay that is caused by traffic congestion or processing delay at a gateway, see Figure 2.15 on page 53. Since a VoIP service depends on the availability of these resources, the interruptions have a profound effect on the overall VoIP performance.

2.3.7 Summary

There are six different categories of VoIP related security threats and attacks. Among others are security threats that affect individuals privacy and attacks that affect the service quality. This indicates that VoIP services can lead to more threats to Internet users. For example, eavesdropping and MiTM attacks compromise one's privacy and DoS attacks cause Internet

and VoIP system to be out of service. Researchers should find ways to protect VoIP services before it is too late.

2.4 VoIP QoS

There are several meaning of QoS. One is on the operation and the other is pertaining to the measurement of service quality. Not to be mistaken, the term QoS sometimes is used to indicate quality of a particular service. A QoS operation ensures that an application is able to transmit data in an acceptable way, in an acceptable time frame so that the transmission is not delayed, distorted, or lost. On the other hand, QoS measurement is related to end-to-end service performance. It is a measurement on how good or bad a service quality is at any point of data transmission over any selected network. The service quality can be determined by measuring the service availability and the data throughput. The service is good if the service availability is high and the data throughput is also high. One way to measure the service availability is to calculate the service downtime for a given period and then calculate the percentage of service availability for the given period. The data throughput could be determined by measuring the amount of successful data transfer in a given time.

The main purpose of QoS operation is to differentiate between different types of traffic and types of services. In which case the different types of service and traffic can be treated differently. Figure 2.16 on page 56 emphasizes several relevant themes as regards to VoIP QoS characteristics. QoS is important for time-sensitive data and interactive application like voice and video streaming over best effort network. Customer satisfaction is important to service providers therefore it is important for a serviced-based organization to maintain a specific level of quality. Naturally, one would compare the quality of service provided by a new system over the legacy system. So, what is the benchmark of this comparison?

Horacio de Oliveira from Telecommagazine.com in 2002 suggested that the quality of telecommunication services depends on a combination of numerous factors such as the quality of call setup, service availability and call quality [68]. Among the three, call quality was most emphasized. Several methods and processes to measure the QoS have been derived. The standard methods and processes were documented in QoS standard like E-Model, Perceptual Speech Quality Measures (PSQM) and Perceptual Evaluation of Speech Quality (PESQ) to name few of them [76, 159, 124, 139]. Other methods like Measuring Normalizing Blocks (MNB) and Perceptual Analysis Measurement System (PAMS) are derived from PSQM and Mean Opinion Score (MOS) respectively [169, 171, 136, 137, 140]. The overall objective of measuring QoS is to find out whether the system operate within the tolerable response time, loudness levels, frequency response, interrupts and noise level. In addition, it minimises cross-talk and any noticeable echoes.

One must understand why QoS is more important to VoIP services rather than PSTN services. As mentioned before PSTN is a dedicated circuit-switched network. QoS was as important then as it is now. A circuit-switched network is not immune to propagation delay, network noise and other errors. That was why sometimes users experience crosstalk over PSTN network. However, the error or noise is still acceptable to the human ear. On top of that, once a call is established, the line is dedicated to one traffic channel until it is terminated. Hence, there was no network resource competition even though the call traffic is passing through several discrete networks.

2.4.1 VoIP Metrics

VoIP is carried over a packet-switched network. The network is shared by variety of applications. No dedicated line is reserved to time-sensitive data and interactive applications such as voice and video streaming. That is why Salah and Almashari encourage that, before implementing VoIP services on a new or existing network, a corporate user should assess its readiness [144]. They suggested that VoIP is bounded by two important metrics, first, available bandwidth and second, end-to-end delay. Subsection 2.4.2 on page 61 describes the method in calculating the required bandwidth in order to ensure that the VoIP service is good and its user experience little or no disruption. This is consistent with the parameters suggested in the E-Model [76].

The QoS operation cannot create bandwidth, it can only efficiently partition bandwidth based on different parameters. Generally, VoIP requires an adequate amount of uninterrupted bandwidth for transmission to be successful, and therefore can strain existing network resources [119]. The QoS operations and algorithms is discussed in Subsection 2.4.3 on page 63. When there is a heavy traffic, the overall performance degrades and results in the traffic degradation due to a one-way delay, jitter and packet loss. A delay is the elapsed of time observed for a complete or an on-going task. The delay influences customer satisfaction because it may cause conversational difficulty. The flow of any telephony conversations is bidirectional. If the delay for the ongoing and the incoming traffics are consistent and within tolerance values, then the calling and called parties would not experience any conversational difficulty. However, when the one-way delay due to the propagation, transmission and queueing delays exceeds 150ms, then both parties would have problem understanding one another [43]. The situation is deteriorated with the jitter and packet loss because in most cases, there is no retransmission for any packet loss and jitter that causes an inconsistent delay. Packet loss concealment (PLC) technique is used to minimise the overall effect of packet loss. For example, to compensate for a single lost packet, a VoIP waveform CODECs, such as G.711, an end device typically plays the previous packet. In case of a VoIP non-waveform CODECs, such as G.723.1, an end

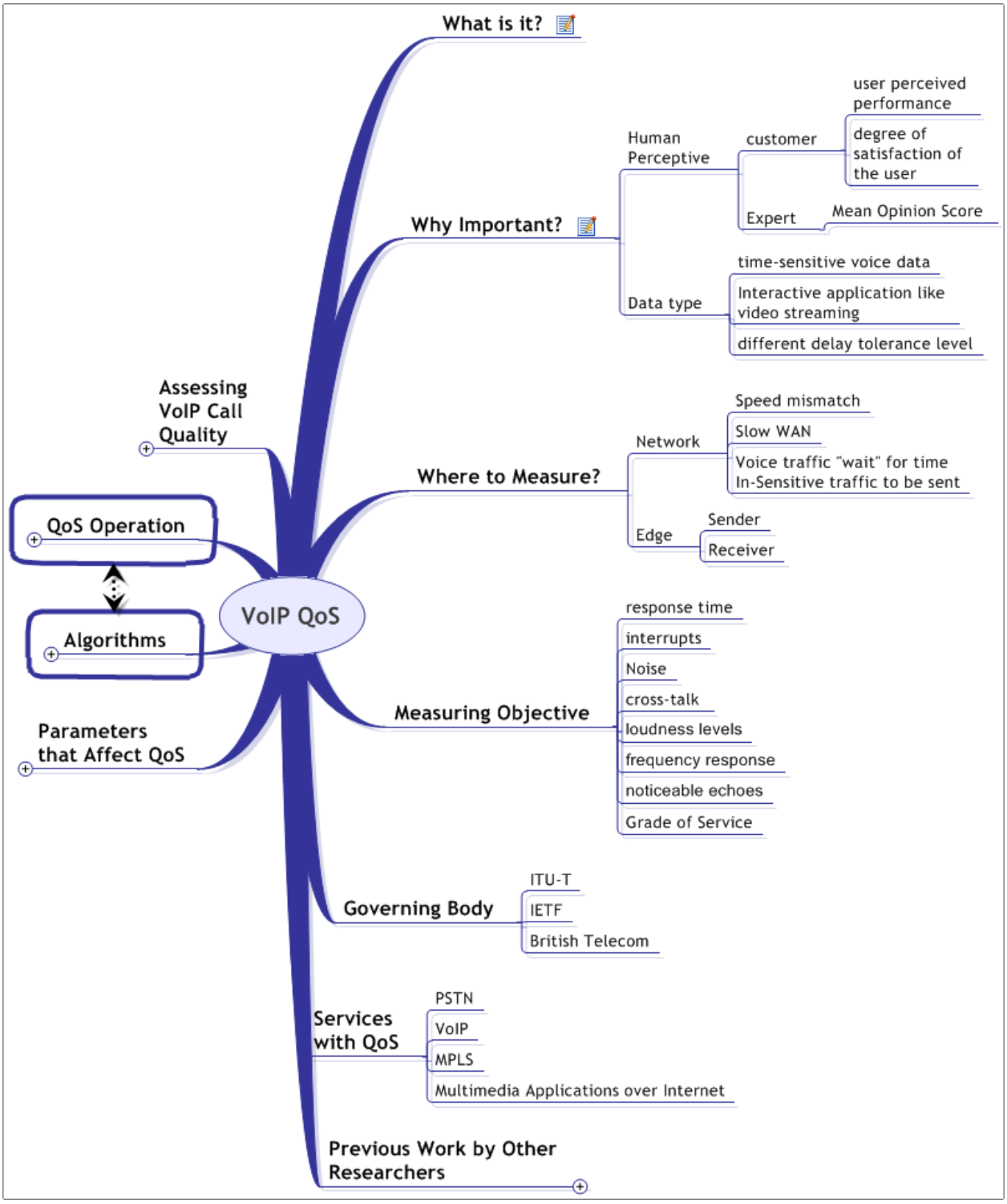


Figure 2.16: VoIP QoS

device typically uses forward error correction (FEC) to compensate for lost packets [3].

PSTN provides a dedicated mouth-to-ear transmission once a call is established. It is a connection-oriented network where calls are assigned a constant bandwidth. The PSTN rejects any new calls when there is an excessive traffic transmitted over its network. However, this is not the case with IP network. The bandwidth is shared between several applications. Unfortunately, there is no guarantee that VoIP services will have enough bandwidth for any single new mouth-to-ear transmission. The problem with the end-to-end delay, jitter, packet reordering and frame erasures in WAN is discussed by Kos et al. [90]. They also explained the emerging techniques for solving those problems. However, other important issues for VoIP such as codec types, VoIP signalling, protocols, securities and its compatibility with PSTN network were not discussed.

It is believed that part of the poor VoIP performance is due to the TCP/IP design. TCP/IP is designed to maintain network reliability and stability and equal bandwidth sharing among applications within the IP network. The design is fine if its objective is to ensure that all packets eventually reach their destinations. Nevertheless, it does no justice to time-sensitive applications like VoIP and video streaming. Any delay, jitter or packet loss affect the quality of such applications.

2.4.1.1 Bandwidth Sharing, Congestion Control and Time Sensitive Traffic

In order to maintain stability and reliability within a shared network, TCP control the transmission made from a sender to a receiver. If TCP granted every users to transfer data whenever they wished, bandwidth would not be shared equally and network throughput is reduced in response to the heavy traffic. The TCP protocol uses a variety of algorithms which basically decelerate the sending application when congestion is discovered. It is designed to spread the deceleration fairly evenly across every contending links, thus giving each link an equal share of the available network bandwidth [5].

In order to measure the congestion, TCP uses two different mechanisms (i.e. Round-trip Time (RTT) and packet loss). TCP cuts its transmission rate by half every time it detects loss. This can contribute to another additional latency after the initial loss. Even small amounts of loss can drastically reduce the TCP throughput. Designers of time-sensitive applications often opposed this equal bandwidth sharing. They want better control over how network bandwidth is assigned to applications so that highly time-critical data can get through quickly, at the cost of less time-sensitive data. Many designers choose a UDP-based messaging system to deliver their time-sensitive data because UDP does not automatically reduce the data transfer rate in the event of congestion. Unfortunately, as many designers have discovered, UDP-based messaging can cause instability in a congested network, sometimes leading to congestion onset

being cleared prematurely. Since a network bandwidth is a finite resource, some measures need to be taken to maintain stability when there are more time-sensitive messages to be sent than the network can satisfied.

VoIP routers and switches require enough bandwidth capacity to support VoIP services. Network vendors have developed mechanisms to calculate bandwidth for different type of codecs [168]. For example, a VoIP over Ethernet with G.711 codec without compression, the bandwidth is 95.2 kbps percall. However, VoIP over Ethernet with G.729A codec requires 39.2 kbps of bandwidth [119]. Compression reduces the amount of required bandwidth but may induce additional latency due to coder/decoder processes at both ends. In addition, any packet losses within compressed packets can seriously degrade the quality of VoIP services.

2.4.1.2 Data Reception

TCP only supports in-order delivery model. This means that a TCP receiver must add extra delay to cater for queueing delay of a de-jitter buffer (also known as a playout buffer) whenever data arrives out of order so as to put the data back in order. TCP also often unnecessarily retransmits data that was already successfully received after out-of-order data is received. There are two main causes of out-of-order data reception. The most frequent cause is packet losses, either at the physical or network-layer. Another, less frequent cause of out-of-order data reception is that packets can take different paths through the network whereby one path might have more latency than the other.

There are several attempts to make TCP friendly towards time-sensitive application [29]. For example, popular applications such as Skype use TCP since UDP packets cannot pass through restrictive network address translators (NATs) and firewalls [16]. TCP has traditionally been considered inappropriate for real-time applications. This is because the congestion control algorithm of TCP leads to a varying throughput for a real-time application and may cause packets retransmission to arrive too late for playback and therefore the packets become useless [179, 129]. Due to the characteristic of TCP that provides variability in the data throughput, most applications use a de-jitter buffer at the receiver side to compensate for any jitter at a trade-off of delays and packet losses. This method is used to improve the Quality of Experience (QoE) of a VoIP user [184]. As with packet switched networks the packet queuing delays are hard to predict. In order to mitigate the problem, a de-jitter buffer holds the VoIP packets temporarily until their scheduled playout time is due [179]. Though the packets experience slightly longer network delays, they can still be used as long as they arrive at the listeners node ahead of their respective scheduled playout time. However, the challenging issue of a de-jitter buffer is to determine the right buffer size for the current network conditions whereby a larger buffer size leads to a better sound quality, but it reduces any conversational interactiv-

ity [179]. In which case, the buffer size adjustment can be treated as an optimization problem. This buffer dimensioning could be incorporated into a VoIP application as only a weighted sum operation is needed to compute the optimal buffer size. One solution is to use elastic buffers. For example, these buffers may be constructed according to the prescribed rules from fixed sets of basic building blocks [103]. The basic building blocks could be a component that store values and a component that divide and recombine stream of packets, including a method to construct arbitarry systems from the set of these basic components. Tree buffers that consist of fan-out and fan-in trees are more flexible than linear buffers that only operate at a maximum throughput at a single occupancy level. One research, the empirical evaluation of a VoIP de-jitter buffer on Skype, Google Talk and MSN Messenger, shows that Skype does not adjust its de-jitter buffer at all while MSN Messenger performs the best in terms of buffer dimensioning even though both Google Talk and MSN do not adjust their buffer sizes appropriately [179]. The research also shows that the QoE of the three VoIP applications could be better by improving their buffer dimensioning algorithms. As a solution, they proposed a simple regression-based algorithm that computes the optimal de-jitter buffer size based on the current network conditions. The issue with the approprite de-jitter buffer is still opened. However, many researchers concentrate on the dimensioning algorithm. For example, Wu et al. suggested a simple algorithm that computes the optimal buffer size based on an objective QoE metric that considers both of the conversational interactivity and the speech quality [179]. Another example by Perwej and Parwej that suggest a modified buffer algorithm that was based on several existing adaptive buffer algorithms and the study on the network impairments to obtain optimised voice quality [129]. However, buffering or caching introduces more latency into the system. The buffer may be miscongured and would be either too large or too small. If a de-jitter buffer is too small then a large number of packets may be discarded, which can lead to a call quality degradation. If a de-jitter buffer is too large then the additional delay can lead to a less conversational interactivity. A typical de-jitter buffer conguration is betweeen 30 milliseconds to 50 milliseconds in depth. Sweeney and Wijesekera suggested that a tradeoff between packet loss and latency is required when sizing the de-jitter buffer and it is usually sized to be one to two times the sample interval if a static de-jitter buffer is used [3]. In the case of an adaptive jitter buffer, then the maximum size may be set between 100 milliseconds to 150 milliseconds. If the buffer size exceeds 100 milliseconds then the additional delay introduced can also lead to a conversational difficulty [129]. To overcome the additional buffer delay, Gong and Kabal have proposed a quality-based playout algorithm with a forward-error correction (FEC) design based on conversational quality including calling quality and interactivity [53, 54]. Their simulation results show that their algorithm is able to correct packet losses and improving perceived conversational quality. They use the E-Model R factor as the cost index to obtain playout delays which adapt with each talkspurt. They use sender-driven

repair algorithms, in which a sender sends redundancy information, to mitigate the impact of the missing packets due to the network impairments and buffer underflow without increasing the buffer delays.

2.4.1.3 Multipoint or Single-point Measurement

The QoS measurement points should be considered in a network design. In general, the QoS can be measured between two network edges or two network nodes, refer to Figure 2.17 on page 60 (i.e. between points A and D or B and C or E and F). Multipoint measurement requires precise synchronization of clocks of the devices used to measure performance. If time stamps are used in packets, careful clock synchronization is needed between the devices in order to calculate delay to both directions.

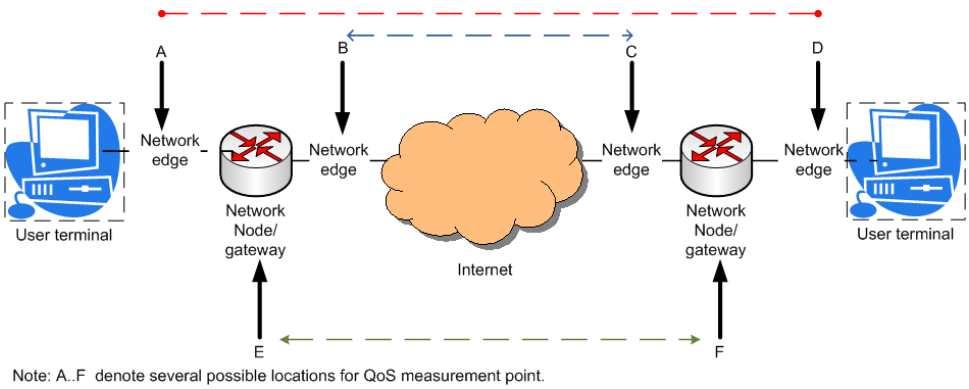


Figure 2.17: QoS measurement points

Other than network edges and network nodes, another measurement points are user terminals. Kim et al. introduced the development of a terminal agent for QoS measurement that is suitable for a Next Generation Network (NGN) environment [89]. The terminal agent, installed in the user terminal (i.e. IP-based audio and video phones), as a software or hardware chip, measured the quality index for voice and video related multimedia services, such as R-value, MOS value, call success rate, one-way delay, jitter and packet loss. The terminal agent also applied the packet capturing method when using the actual service, and analysed SIP, RTP, and RTCP protocol headers of the received packet. SIP, RTP and RTCP are discussed in Section 2.2 on page 32. In some cases, it is more appropriate to have a single-point measurement point. A single-point measurement provide end-to-end performance information, which can be used to find QoS. This setup enables the possibility to measure round-trip time (RTT). The end-to-end performance information is a valuable QoS metric and gives direct insight into the total performance of the system. For example, RTT is one of the parameter required to estimate the R-factor in E-model.

2.4.2 Bandwidth Calculator

A VoIP service requires enough bandwidth to generate a good quality call. However, there are a number of factors that affect a VoIP bandwidth requirement. In addition, the VoIP service coexists with other Internet applications. The data applications including any VoIP data that exist on the Internet are either transmitted over TCP or UDP transport modes, some might be delay sensitive, some produce traffics that are either bursty or benign in nature, others are lightweight or eat up bandwidth and so forth [160]. Voice traffic is smooth, benign and very predictable. For example, uncompressed voice traffic with G.711 codec requires about 80kbps to 95.2kbps of steady bandwidth per call depending on which TCP/IP protocol layers encapsulate the voice payload [119]. The voice traffic would not demand for more bandwidth unless there is another call in which case the bandwidth required is the multiplication of the number of calls on the communication link. However, a voice traffic is very sensitive to delay, jitter and packet loss. Video traffic, on the other hand, is predictable, bursty and bandwidth greedy. The video traffic is also sensitive to delay, packet loss and jitter. However, generally video traffic normally gets lower priority to a voice traffic. Data traffic characteristics vary from one application to another. For example, FTP traffic is bursty but insensitive to delay, jitter and packet loss. However, there is another type of data traffic that is smooth and requires uninterrupted bandwidth, for example data traffic from a citrix server to a client. This type of traffic is again sensitive to delay, jitter and packet loss. In case of a voice traffic, there are several factors that influence the bandwidth requirement. First, the codec type, the voice payload size and the coder delay. Second, the network topology and the third is the number of calls made on the communication link. The details of these factors are elaborated in the next Subsubsection 2.4.2.1.

2.4.2.1 VoIP per call bandwidth

Traffic engineering is an important mechanism for Internet network providers seeking to optimise network performance and traffic delivery by dynamically analyzing, predicting and regulating the behavior of data transmitted over that network. A major objective of a traffic engineering is to minimise or eliminate high-loss situations. The techniques of traffic engineering can be applied to networks of all kinds including wireless, PSTN and internet [176, 187]. Typically, traffic engineering involves converting a legacy 64 kbps voice channel to an IP throughput needs. The conversion factor varies depending on the codec, IP version, and the payload size. The IP throughput calculation should include IP overhead and user signalling packets [3]. In case of the VoIP traffic, the IP throughput needs is related to the VoIP per call bandwidth requirement.

VoIP per call bandwidth requirement naturally depends on the codec used. When cal-

culating the bandwidth, one cannot assume that every channel is used all the time. Normal conversation includes a lot of silence, which often means no packets are sent at all. So even if a user voice call sets up on multiple 64 kbps RTP streams over UDP over IP over Ethernet, the full bandwidth is not used at all times. A codec that sends a 64kbps stream results in a much larger IP network stream. The main cause of the extra bandwidth usage is the IP and UDP headers. VoIP sends small packets and so, many times, the headers are actually much larger than the data part of the packet. To overcome this problem there are many techniques available, for example Junaid in his paper entitled ”Emerging Methods for Voice Transport over MPLS”, suggests that other than to compress its frame headers, to reduce the bandwidth consumption, different voice packets within a same connection are multiplexed together into one MPLS frame. Similar techniques can be applied to other types of IP networks. The popular techniques are headers compression, silence suppression, packet loss concealment, queue management techniques, and encapsulating more than one voice packet into a single frame [3]. However, some of these techniques introduce other side effects, for example, silence suppression might have a negative effect on the packet loss rate, despite of the fact that it significantly reduces the load on the channel. In addition, encapsulating more that one voice packet into a single frame would increase the overall delay caused by a Coder delay [32]. Nevertheless, in general, these techniques would produce relatively higher, but acceptable, end-to-end delay.

Many vendors or reseachers used similar method of calculating the VoIP per call bandwidth [3, 119, 31]. Table 2.2 on page 63 shows a few samples of the calculation. The calculation assumes the following conditions:

i. voice calls are symmetric and that no voice conferencing is implemented

ii. signalling traffic is ignored, mostly generated by the gateway or the gatekeeper prior to establishing the voice call and when the call has been completed

iii. header size is based on Layer 2 header, IP version 4 header, TCP or UDP header and RTP header or cRTP header

iv. voice payload size depends on the type of codec in used

v. without packet loss concealment

vi. without silence supression, However, the bandwidth requirement is reduced by fifty percent if VAD is set on the sender codec

vii. for every sample period there is only one packet and each packet can contain multiple frames

Hence the calculations:

Total Packet size = (L2 header) + (IP/UDP/RTP header) + (voice payload size)

Packet per seconds = $\frac{NumberOfPacketPerSample}{(NumberOfFramePerPacket)(SamplePeriodInSeconds)}$

whereby Packet per seconds represents the number of packets that need to be transmitted every second in order to deliver the codec bit rate,

hence, **Bandwidth** = **Total Packet size**(bits) x **Packet per seconds**

Table 2.2: VoIP per call bandwidth calculation

Codec type	G.711 (PCM)	G.729A (CS-CELP)	G.729	G.723.1A (ACELP)
Sample period	20ms	10ms	10ms	30ms
No. frame per packet	1 frame per packet	2 frames per packet	2 frames per packet	1 frame per packet
Header compression	No	No	Yes - cRTP	No
Codec bit rate	64kbps	8kbps	8kbps	5.3 kbps
L2 header	Ethernet-38 bytes	Ethernet-38 bytes	Multilink Point-to-Point Protocol-6 bytes	Ethernet-38 bytes
RTP/UDP/IP header	40 bytes	40 bytes	2 bytes	40 bytes
Voice payload size (bytes)	160 bytes	20 bytes	20 bytes	20 bytes
Voice payload size (bits)	1280 bits	160 bits	160 bits	160 bits
Total Packet size (bytes)	238 bytes	98 bytes	28 bytes	60 bytes
Total Packet size (bits)	1904 bits	784 bits	224 bits	784 bits
Packet per seconds	50 pps (1000/20) pps	50 pps (1000/10)/2 pps	50 pps (1000/10)/2 pps	33.3 pps (1000/30) pps
Bandwidth per call	95.2 kbps	39.2 kbps	11.2 kbps	26.1 kbps

2.4.3 QoS Operations and Algorithms

There are three basic operations of QoS, refer to Figure 2.18 on page 65. They are listed as follows:

i. Classification and marking of packets. Classification and marking is a system of identifying packets or traffic flows and assigning certain parameters within the packet headers in order to group them. Once the traffic is identified, it can be marked or colored into groups so that QoS policies can be applied to them. Random Early Detection (RED) and Weighted Random Early Detection are examples of QoS classification and marking algorithms.

ii. Policing and shaping traffic to regulate it with an acceptable rate. Policing and shaping are QoS components used to limit traffic flow. Policing drops or remarks traffic that exceeds limits, but shaping regulates the traffic back to a defined rate by delaying or queuing the traffic. The idea of policing is to drop any traffic in a network that exceeds a limit. This is to avoid one application dominating the entire available bandwidth. This is an analogy to a set speed limit on a motorway where lorries and heavy vehicles might have different lanes and speed limit to cars. VoIP applications are rather vulnerable during periods of heavy network usage or spikes [162]. Shaping traffic on the other hand is applied on a low speed network. This is something like a traffic officer redirects cars to avoid road maintenance on a four-lane traffic going through a two-lane.

iii. Queuing and scheduling to ensure that high priority packets, (i.e. time sensitive packets such as VoIP packets), get better treatment over other data packet with less error, delay and packet loss. Queuing theory for QoS consisting primarily of scheduling algorithms like Weighted Round Robin, and Weighted Fair Queuing, and Class-Based Weighted Fair Queuing, and Priority Queuing, etc.

QoS operations are applied on high-speed networks. QoS algorithms are derived to support QoS operations. For example, Floyd and Jacobson presented the RED gateway algorithm for congestion avoidance in packet-switched network [50]. RED gateways keep the average queue size low while allowing occasional bursts of packets in the queue. RED statistically drops packets from flows before it reaches its hard limit. This causes a congested backbone link to slow more gracefully, and prevents retransmit synchronization. This also helps TCP find its 'fair' speed faster by allowing some packets to get dropped sooner keeping queue sizes low and latency under control. The probability of a packet being dropped from a particular connection is proportional to its bandwidth usage rather than the number of packets it transmits. The most effective detection of congestion can occur in the gateway itself. The gateway can reliably distinguish between propagation delay and persistent queueing delay. RED gateways can be useful in controlling the average queue size even in a network where the transport protocol cannot be trusted to be cooperative. RED gateways are designed to accompany a Transport Layer congestion control protocol such as TCP.

2.5 Tools for Assessing Voice Quality

The goal of QoS operation is to provide preferential delivery service for applications that need it by ensuring sufficient bandwidth, controlling latency and jitter, and reducing data loss. Without preferential QoS, the intrinsic QoS for most packet-switched networks does

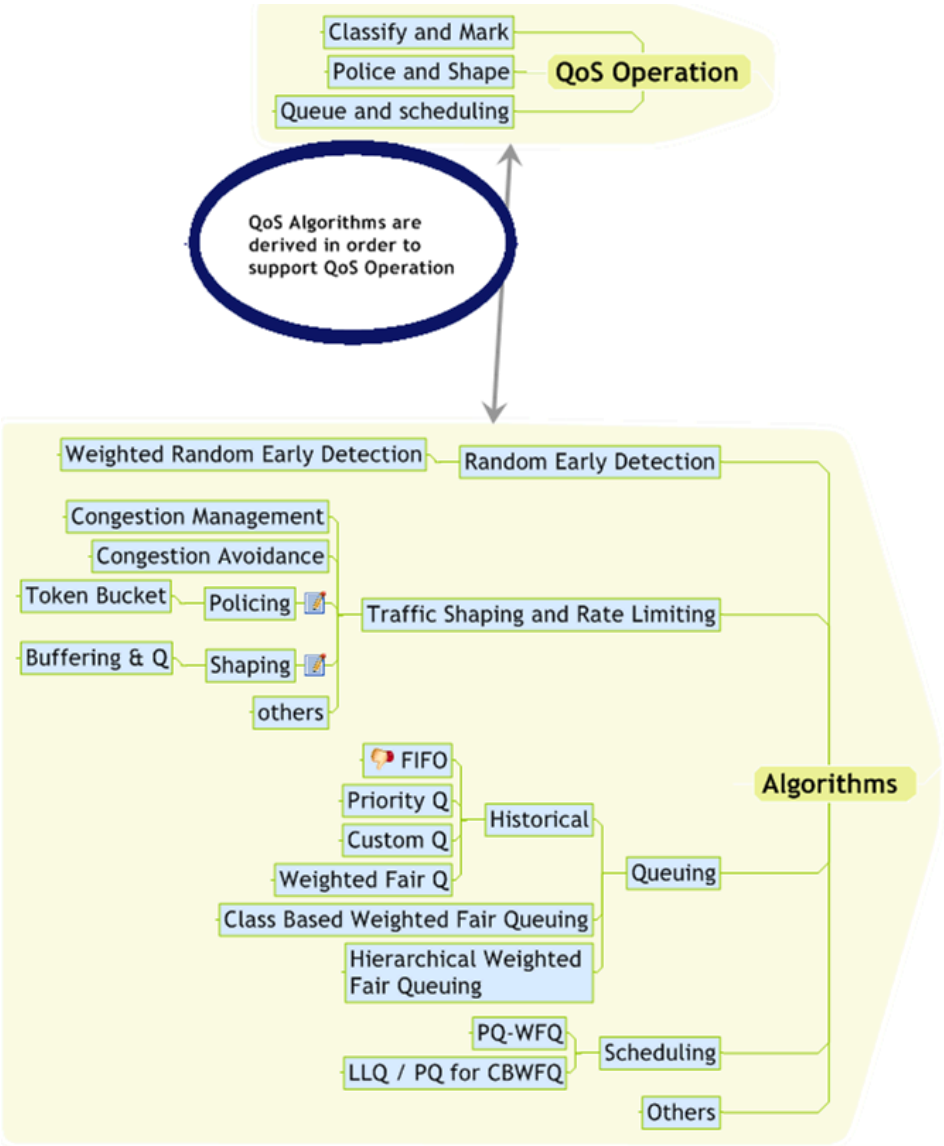


Figure 2.18: QoS operations and algorithms

not support adequate perceived QoS [66]. Before going further, let differentiate the following terms: Preferential QoS, Intrinsic QoS and Perceived QoS.

i. Preferential QoS ascribes both class of service (CoS) and type of service (ToS). The basic goal of CoS and ToS is to achieve the bandwidth and latency needed for a particular application. CoS enables a network administrator to group different packet flows, each having distinct latency and bandwidth requirement. ToS is a field in an IP header that facilitates CoS.

ii. On the other hand, intrinsic QoS refers to those characteristics that can be measured by a provider without reference to user perception of quality even though they affect user perception. Some metrics that can be measured include delay, jitter and packet loss rate.

iii. Perceived QoS is what users experience as to the effects of intrinsic QoS on their communications activities, in their environments, in handling their demands and how they react to that experiences in light of their personal expectations. In the end, users determine whether they are satisfied with the services based on the two attributes of a voice service, i.e. connection quality and connection usability. Connection quality is determined by what is heard over the connection and connection usability is determined by what is experienced in conversational exchanges over the connection. In short, user perception of the voice quality is subjective and dependent on tastes, dislikes and expectations of each individual user.

Since, users experiences are very subjective, measuring their satisfaction is nontrivial. There are essentially two approaches for assessing voice quality: subjective and objective methods. Subjective methods, refer to Subsection 2.5.1 on page 67, employ human listeners to evaluate voice quality and can evaluate all relevant aspects of voice quality. On the other hand, objective voice quality methods such as the PESQ algorithm seek to estimate certain voice quality aspects as perceived by human beings. Figure 2.19 on page 71 shows the relationship between the subjective and objective methods. Another classification is on monitoring schemes. It is difficult to measure QoS and performance statistics of traffic conveyed by the Internet because it is generated by a wide variety of applications, which have different characteristics and different quality requirements. Conventional monitoring schemes to measure QoS and the performance of networks are classified into active monitoring and passive monitoring [4, 112]. Both types have draw backs. Passive monitoring technique is carried out by observing network traffic flows. They consist of capturing packet headers and analysing them. The best example of a capture tool is tcpdump. Measurement can be performed at microscopic level on each packet travelling across the measurement point or at macroscopic level whereby measurement performed on some traffic flows. The problem with passive measurement is on

the high volume of data being captured. Active monitoring is performed by sending probe-packets to a network. The measurement flow travels from source to destination. Upon reaching its destination, it is possible to calculate metrics by analysing them. The main drawback of active measurements is that additional network traffic is introduced. This invasive characteristic can potentially modify the properties of the network that are being measured. Aida et al. had derived a mathematical foundation for a hybrid technique that enabled them to measure detailed QoS information for individual users, applications, and organizations, in a scalable and lightweight manner [4]. The results of their simulation showed that their technique gave accurate QoS estimations with only a small amount of extra traffic for active measurement.

The measurement clasifications, (i.e. subjective versus objective and/or active versus passive), influence the type of tool and method to be used in measuring voice quality in order to determine the perceived QoS.

2.5.1 Mean Opinion Score

The Mean Opinion Score (MOS) is the most widely used subjective measure of voice quality and is recommended by the ITU-T Recommendation P.800, ”Methods for subjective determination of transmission quality”. A MOS value is normally obtained as an average opinion of quality based on asking people to grade the quality of speech signals on a five-point scale (Excellent, Good, Fair, Poor and Bad) under controlled conditions. In voice communication systems, MOS is the internationally accepted metric as it provides a direct link to voice quality as perceived by the end user. The main disadvantage of subjective MOS measurement is that it involves relatively a large number of test participants and extensive sampling requirements. Therefore it is labour-intensive and relatively expensive and cannot be used for long-term or large scale voice quality monitoring in an operational network infrastructure. This has made objective methods very attractive for meeting the demand for voice quality measurement in communication networks.

2.5.2 Perceptual Speech Quality Measure

The Perceptual Speech Quality Measure (PSQM) is an objective measure of speech codecs quality recommended by ITU-T P.861 standard. The PSQM scope is limited to assessment of telephone-band speech codecs only [124]. The PSQM measure is calculated by comparing source waveforms recorded in the clear, with the test waveforms that have been encoded and decoded through the codec being tested. Two steps are involved. In the first step both source and test inputs are transformed into speech waveforms that would result from frequency filtering and shaping at the sender and receiver sides of a telephone handset including some noise. The outputs (at the earpiece) are processed through a model of hearing to create their internal

representations, comprising the sound information as it would be sensed by human ear. The source and test inputs are differentiated by the different in the frequency spectra between the two. There is still some limitation of PSQM. For example, it is difficult to determine the correlation between PSQM and MOS because there is a need to derive such mapping functions for individual languages and individual subjective tests in advance.

2.5.3 Measuring Normalizing Blocks

The Measuring Normalizing Blocks (MNB) method is developed by the US Department of Commerce in 1997 and has been proposed as an alternative technique to the PSQM [39, 170]. It is recommended for the impact evaluation of some parameters that affected the signal such as error due to the communication channel or error due to the lower than 4kbps bit rate codec [72]. The MNB algorithm comprises of two stages (i.e. a simple perceptual transformation, and a distance measure that uses hierarchies of measuring normalizing blocks) [61]. MNB transforms speech signals into an approximate loudness domain through frequency warping and logarithmic scaling. It assumes that these two factors play the most important role in modeling human auditory response. The algorithm generates an approximated loudness vector for each frame. The overall speech distortion was estimated from a linear combination of 11 or 12 MNBs calculation. The weights for each MNB are optimised with a training data set. Similar to PSQM, MNB falls under P.861 standard. The P.861 was recognized as having certain limitations in specific areas of application, for example it has been found to be suitable for assessing only a limited range of distortions [140]. It was replaced by P.862, which contains improved objective speech quality assessment algorithm that covers a wider range of network conditions, including analogue connections, codecs, packet loss and variable delay.

2.5.4 Perceptual Analysis Measurement System

Perceptual Analysis Measurement System (PAMS) is a psychoacoustic measurement technique very similar to PSQM developed by British Telecom for their internal use.

It is used as a tool to calculate a frequency spectrum error surface of test sensation to source sensation. PAMS is an objective measure of distortion by calculating the average of different varieties of differences exhibited in error surface using a regression techniques to fit test results developed from extensive in-lab subjective testing. There are several distinct features of PAMS over PSQM techniques as follows:

i. PAMS uses specially designed source waveform, a proprietary artificial speechlike test stimulus (ASTS).

ii. PAMS produces two measures of voice quality. First, is the estimation of a MOS for listening quality and sampled. Second, is to establish formulae for predicting MOS for listening effort that allow distinction between voice quality heard and usability of the connection for service attribute testing.

iii. PAMS provide procedures for checking and realigning the source and test signals that is important in the packet-switched environment. The most obvious source was adjustments in the de-jitter buffer created by an adaptive de-jitter control.

PSQM and PAMS share a similar limitation. Both PSQM and PAMS rely on subjective user testing under controlled conditions to establish the relationship between the measures calculated and predicted MOS.

2.5.5 Perceptual Evaluation of Speech Quality

Perceptual Evaluation of Speech Quality (PESQ) is an objective measure of speech quality recommended by ITU-T P.862 standard. It is the result of integration of PAMS and PSQM99, an enhanced version of PSQM. It is considered as an objective method for end-to-end assessment of narrow-band telephone networks and speech codecs. It is used across a wider range of network conditions, including analogue connections, codecs, packet loss and variable delay [139]. The range for PESQMOS is also a five-point scale between 0 (bad) to 5 (Excellent). PESQ assumes that a subjective listening level is a constant 79dB SPL at an ear reference point. A gain is applied to both original and degraded signals to bring them to this level. It compensates for any filtering that takes place in the network. It aligns both signals before comparison. Alignment comes in three stages. First the alignment of utterances. This is to detect any delay over major sections of the degraded signal compared to the original signal. Then the alignment of overlapping speech frames in order to detect any variable delay over the length of an utterance which is important in packet-based networks. The third stage is performed after an auditory transform has been calculated to realign any speech frames with very large disturbance and to improve the models accuracy which may not be correctly identified by the earlier time alignment process. PESQ measures audible errors that occurred in the degraded signal. A quality score is calculated based on these errors [102].

There are some limitations of PESQ that are in common with PAMS that limit their applicability on packet-switched due to psychoacoustic test procedures [138, 11]. PSQM, PAMS and PESQ should not be used as a stand-alone gauge of quality of packet-switched voice services. Application of PESQ may reveal poor prediction of MOS value for end-to-end call across hybrid transport. MOS value reflects the effects of both noise and waveform distortion. It is imposible to predict whether the poor quality is due to attributes of the circuit-switched net-

work or packet-switched transport or both. In July 2011 a new objective listening algorithm standard, (i.e. P.863), substitues the P.862 standard , refer to Subsection 2.5.7 on page 70. However, the PESQ measurement tool is used in this research. Subsection 4.6.1 on page 103 describes how the tool is used in the research.

It is important to highlight that PESQ is not really able to adequately align signals with the type of jitter found in VoIP systems. Hence, this may cause some higher variations in the results than might be expected. Moreover, although P.863 POLQA addresses these issues but it was not available when the experimental work of the thesis was carried out.

2.5.6 E-model

E-model is a planning tool for use in transmission planning that assigns a certain equipment impairment factor to each piece of equipment in the transmission chain. It is a recommendation of ITU-T G.107 standard. E-Model was initially designed by ITU-T for defining the QoS of the PSTN. The model estimates the conversational quality from mouth to ear as perceived by the user at the receive side, both as listener and talker [76, 108].

The output of an E-model calculation is a single scalar, called an R factor, that is derived from delays and equipment impairment factors. Once the R factor is obtained, it can be mapped to an estimated MOS. The range of the R factor is nominally from 0 (poor) to 100 (excellent). However, values of below 50 are generally unacceptable and typical telephone connections do not get above 94 (i.e. Ro is approximately 93.4), giving a typical range of 50 to 94 [80]. The E-Model algorithm is explained further in Subsection 3.3.3 on page 84. Although an E-model is an excellent planning tool, it can never replace real measurements on the final network, since it has to make some very wide range of assumptions.

2.5.7 Objective Listening Quality Assessment

Another measuring tool is the Objective Listening Quality Assessment (OLQA). It is a standard under ITU-T P.863 Recommendation. It has replaced ITU-T P.862 Recommendation since July 2011 [77]. It describes an objective method for predicting overall listening speech quality from narrowband (300 to 3400 Hz) to superwideband (50 to 14000 Hz) telecommunication scenarios as perceived by the user in an ITU-T P.800 or ITU-T P.830 Absolute Category Rating (ACR) listening only test. Unlike the PESQ which was developed for narrowband frequency, the OLQA supports two operational modes, one for narrowband and one for superwideband. The algorithm compares a reference signal with a degraded signal as a result of passing the reference signal through a communications system. The output of the algorithm is a prediction of the perceived quality of the degraded signal. P.863 was not available at the time of the start up of our work, hence was not used in the research.

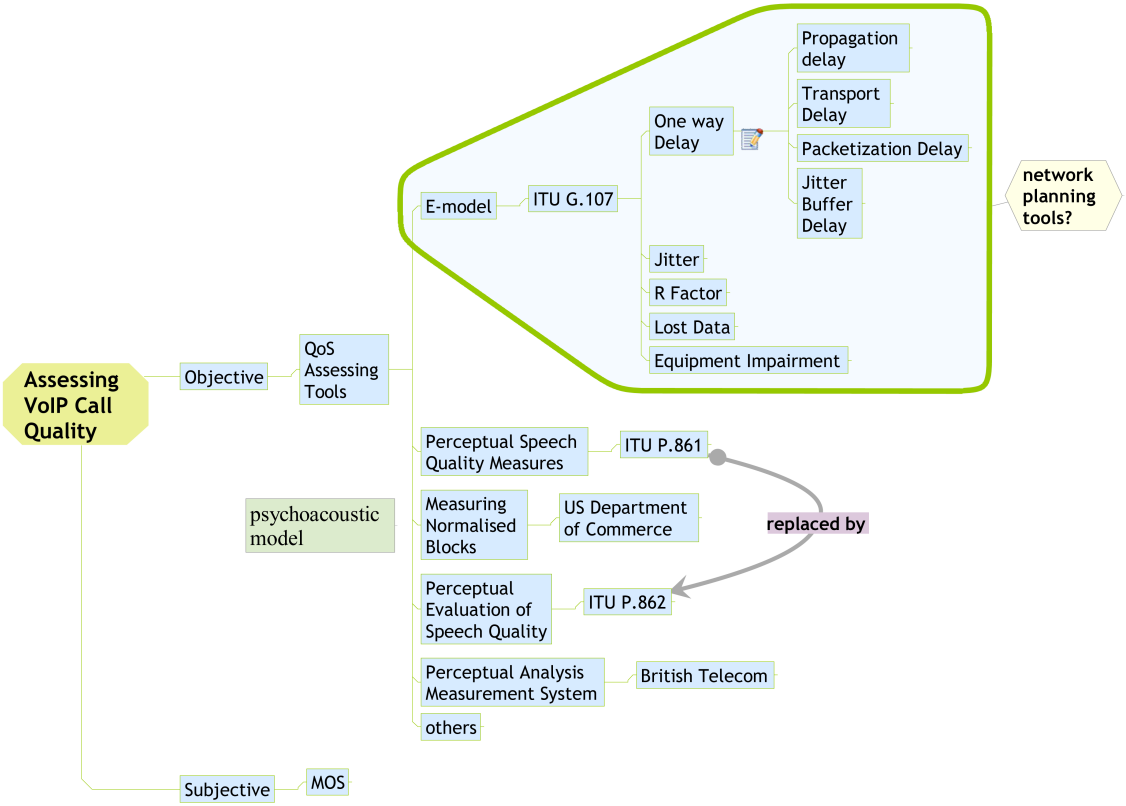


Figure 2.19: VoIP QoS Assessing tools

2.5.8 Summary

In this section VoIP QoS have been discussed. It is believed that two major areas that need emphasized here are bandwidth allocation and end-to-end delay. Several telephony QoS assessment tools like E-Model, PSQM, MNB, PESQ and PAMS were mentioned, refer to Figure 2.19 on page 71. It is stressed that QoS is more important to VoIP than PSTN due to the nature of their respective networks. The parameters that affect QoS like jitter, codec types, VoIP protocols, available bandwidth, data size, end-to-end distance, echo and errors were also highlighted. In the latest development , there is the ITU-T P.863 Recommendation (OLQA) for predicting overall listening speech quality for narrowband to superwideband signalling frequencies. P.863 substitutes P.862 standard since July 2011.

In the next section, several previous works by other researchers are presented. These researches provide a general idea on the type of research that worth pursuing.

2.6 Previous Work

Research on VoIP QoS and VoIP security have inspired a few interesting research areas. There are three main streams: QoS, Security, both QoS and Security. However, they are not the only research interests. There are others such as research on integration and interoperability among diversified VoIP technologies, research on standardization of functionalities and protocols, research on how to cater for emergency numbers and many more that are equally important. Research on VoIP QoS is mainly on how to improve any selected existing QoS algorithms or inventing new algorithms, study on different codec types, measuring VoIP performance or finding out the impact of a particular VoIP architecture. On the other hand, research on VoIP security focus on VoIP security issues that are related to SIP signalling, TCP/IP layered security like TLS, DTLS, IPSec and SRTP, encryption and decryption and authentication techniques, VoIP Forensic and IDS/IPS and security assessment of the underpinning architecture based on the existing standards. For the QoS and Security, the research is related to the development of lightweight security algorithms that they do not consume too much bandwidth and other resources and also the impact of a particular security mechanism on the overall VoIP performance. Many more related researches can be found in [86].

In the next subsections several examples of related research works by other researchers of each stream are presented.

2.6.1 Quality of Service (QoS)

Peh et al. proposed a procedure to evaluate the maximum number of ongoing calls that can be supported by the hybrid network while maintain the call quality at the desired level using the ITU-T specified E-model. WiMAX and WiFi is set up and used as a testbed to study the VoIP Performance on a hybrid wireless network [126].

Pitts et al. performed an exploration on how Random Early Detection (RED) algorithm that was developed to improve the congestion avoidance behaviour of TCP/IP traffic can also benefit real-time interactive applications [130]. A RED controlled buffer, modelled as an M/M/1 queue gives explicit formulas relating load, performance, and RED configuration parameters. Results for VoIP show that RED regulates the mean delay and delay variation under congested conditions, and that the new analysis accurately bounds performance.

Muppala et al. conducted a study on the performance of VoIP traffic aggregates over differentiated services (Diffserv) enabled network using expedited forwarding (EF) per hop behavior (PHB) [115]. The delay and jitter performance of the VoIP traffic generated by different standard voice codec algorithms, both under Diffserv with EF PHB and with best-effort service were compared for both homogenous and heterogenous voice traffic aggregates.

Lakaniemi et al. performed a study on a system for measuring application-level VoIP QoS whereby a simple method for combining measurement data with speech coding simulation software was derived [94]. This method enable reliable evaluation of subjective speech quality being measured based on MOS.

2.6.2 Security

Recreating a crime scene using a network forensics approach can be an invaluable way of investigating the source of the jitter, delay, or other ”call quality crime” [162]. It involves storing all data and voice packets that cross a network in order to go back and identify the problem, but it can prove invaluable, as network administrators hardly have the time to sit idly, waiting for something to happen.

Leung and Chan reverse-engineered Skype, a popular peer-to-peer voice over IP application [99]. Skype’s ability to traverse NAT and bypass firewalls, as well as induced bandwidth burden due to the super node mechanism, make Skype a considerable threat to enterprise networks security and availability. Because Skype uses both encryption and overlays, detection and blocking of Skype is non-trivial. With the forensic knowledge gained, enterprises are able to regulate or block Skype activities over their networks.

Hermant et al. offered solution to detect and overcome hybrid floods [151]. They offered the VoIP Flooding Detection System, an online statistical anomaly detection framework that generated alerts based on abnormal variations in a selected hybrid collection of traffic flows. It viewed collections of related packet streams as evolving probability distributions and measuring abnormal variations in their relationships based on the measurement of variability between two probability distributions.

Abdelnur et al. presented a security management framework for VoIP network [1]. This framework is capable to perform advanced security assessment tasks for such a network. The developed tool and some of its key components was highlighted. The learned experience while implementing the framework on a internal testbed was revealed.

2.6.3 QoS and Security

Most existing VoIP security solutions neglected the other important attribute of VoIP technology, the VoIP QoS. In most cases it seems impossible to have a secured VoIP services with excellence QoS. A firewall deployment at network gateway for example refrains any calls setup, therefore closes up any end-to-end transmission being made. However, a few researchers are able to address some of the problem which had both security and QoS features, by balancing out these two criteria into one particular algorithm, refer to [107] for details.

Ranganathan and Kilmartin analysed the performance of secure SIP-based VoIP networks [134]. They examine the options for securing VoIP networks and the impact of these options to the performance of secure networks. Though the test was on OPNET simulation, the knowledge gained apparently can be used by network designers when considering a real secure VoIP networks. Cha et al. analysed the effect of a security protocol on the performance of SIP particularly on the call setup delays that occur with various security protocols [25]. Several simulations were performed on ns-2 for three security protocols and three transport-layer protocols suggested for SIP.

Leckie investigated the impact of DoS attacks on SIP infrastructure, (i.e. on an open source SIP server) that posed a serious security threat to the quality, reliability and availability of VoIP operations [97]. He identified four attack scenarios that could exploit the vulnerabilities in the existing SIP authentication protocols and demonstrated the practical impact of these attacks on the target server. His experimental results showed that the current SIP implementation was highly vulnerable to DoS attacks and countermeasures were required to form more resilient servers. He also proved that authentication alone was not enough to mitigate the vulnerability of the target servers, instead it contributed to the DoS attacks.

2.6.4 Summary

In this section three different kinds of research areas are highlighted. A few examples of each area are presented. Previous work by other researchers inspire new researches on VoIP. Since the subjects are broad, the research area is scoped into VoIP standards that are governed by IETF, ITU-T and proprietary sevices like Skype, Google Talk and YMSG. An investigation and comparison on the performance of these services under stringent VoIP security mechanisms would be a topic of interest. The result of the investigation could help system administrator and network designer to understand better the relationship between VoIP security and performance. Next, in chapter 3 we discuss the methodology on how we conduct our research.

Chapter 3

Methodology

*"**R**ead: In the name of thy Lord Who createth, **C**reateth man from a clot. **R**ead: And thy Lord is the Most Bounteous, **W**ho teacheth by the pen, **T**eacheth man that which he knew not. "*

Al-Quran:Chapter 96 verses 1-5

This chapter describes the strategy and methodology of our project. The instrumentation framework, test scenarios and a method for testing, analysis and verification processes are presented. Subsequently, the project scope is defined. Section 3.1 introduces several performance criteria that are related to VoIP services. In section 3.2 the instrumentation framework has been put forward. Section 3.3 defines the testing method that is tailored for the project. Section 3.4 summaries the discussion of this chapter.

3.1 Introduction

The methodology is depicted in Figure 3.1 on page 76. The process starts by identifying test scenarios, setting up monitoring tools, setting up IP VPN, setting up VoIP services, setting up packet interceptor tool, monitor service performance, analyse data until validation on the result. A new framework is proposed in order to assist the testing, analysis and verification processes, taking into considerations the conditions that are discussed in Subsection 3.2.1 on page 78. For example, we exclude NAT and firewall implementations from our testbed so that we can reach the VoIP servers without difficulty. The framework defines the requirement of our testbed architecture, hence, we have left some details regarding the implementation. Details of the implementation are available in Chapter 4 on page 87.

Services like Skype, Google Talk and YMSG are popular among their users. These services are deployed differently as describe in Figure 3.1 on page 82, however, they have one common goal that is to provide either voice or instant messaging services. Since each service deploys different level of security, some are less secure than others. For example, YMSG

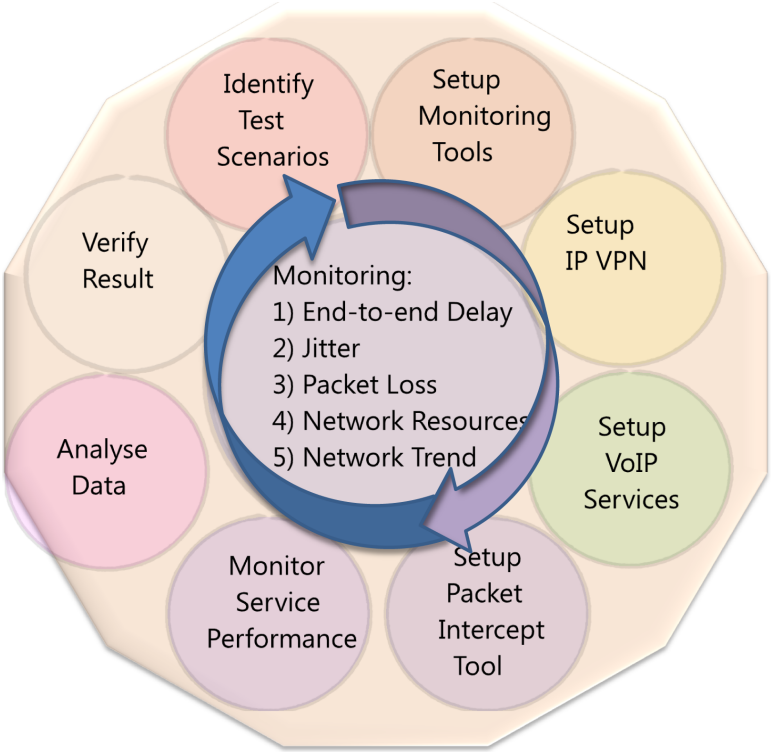


Figure 3.1: Methodology

contents are potentially the least secure because they are sent in plain text. It is interesting to observe how different security characteristics affect the performance of these services. Since they are proprietary services, users have no control over the security features that come with them. Nevertheless, users expect the performance of each VoIP service to be of good quality and secure all the time. Therefore, they might take extra precaution by securing their network further using the method suggested by Butcher et al. [18]. Each VoIP performance should not degrade further once the extra security features are in place. Frameworks that support the monitoring of a VoIP performance are scarce. The nearest similarity is an experimentally study on the relationship between resource utilisation on a wireless LAN and the quality of VoIP calls transmitted over the wireless medium [117]. Another proposal on the deployment of a monitoring architecture based on their proposed Management Information Base (MIB) that collects, stores and displays speech quality information about concluded voice calls [38].

Once the experimental environment has been established, VoIP traffic analysis that link both VoIP security and VoIP performance is performed. However, first, the initial performance of each service over the insecure Internet is recorded. These measurements are compared to the performance of the services within several secure environments. In order to understand the performance of voice communications, call quality, voice quality[1], network quality[2] and

[1]i.e. echo, clipping, distortion, noise and *delay*
[2]i.e. signalling, accessibility and availability

service quality[3] must all be defined [68]. They relate the quality of service to the user's call quality and it is divided into three areas: voice quality that affects the user's conversation; network quality that affects the user's experience on the network; and service quality that affects the amount and type of services offered to the user.

In this project, however, the VoIP performance is focussing on the voice quality and security features that are being deployed. The voice quality is influenced by the traffic delay due to one-way delay, jitter and/or packet loss. In addition, network resources like available bandwidth, routers' CPU power and memory size can also influence the voice quality. The other aspects of VoIP performance like the quality of call setup, interoperability between different VoIP services and additional features of each service, are beyond the scope of this project. These can be found in [68, 25]. This project emphasises on the media quality and security of a VoIP service.

3.2 Instrumentation

A framework for instrumenting VoIP under different security scenarios is devised. Refer to Figure E.1. The architecture collects together a number of available tools and allows monitoring and control of network and host-system resources. The result is estimated using E-Model as implemented by other researchers in [117, 126]. The E-Model was initially designed by ITU-T for defining the QoS of PSTN therefore there is a significant challenge because VoIP and PSTN having different design criteria [76]. However, Carvalho has reformulated the equation for VoIP speech quality evaluation and the results are calculated from this formula [24]. We used Carvalho version of E-Model to estimate the readiness of our testbed to support any VoIP calls. The E-model is used to compare several VoIP services over several network scenarios. The Carvalho formula is defined in Subsection 3.3.3 on page 84. The output of an E-Model calculation is a single scalar, known as R factor, derived from delays and equipment impairment factors. The R factor ranges from 0 (poor) to 100 (excellent). This metric is described in Subsection 2.5.6 on page 70. Another metric is the Mean Opinion Score (MOS), which also comes from the telephony world. This metric is described in Subsection 2.5.1 on page 67. The mapping between network characteristics and a quality score makes MOS valuable for network assessment and tuning. MOS works well but is expensive to deploy and takes a long time to process [63]. The good news is that the human behavioural patterns have been heavily researched and captured. By mapping the R factor to MOS value, the voice quality of any call can be estimated.

Each service security score is estimated using Common Vulnerability Scoring System

[3] i.e. externality, services and *security*

(CVSS). CVSS is grouped into three main metrics, the Base Metrics, the Temporal Metrics and the Environmental Metrics. The Base Metrics group is set by vendors and once set does not change. The Temporal Metrics group is set by vendors but can change with time. The Environmental Metrics group is optionally set by users and provide final score. The Base Metrics group consist of the impact bias, access complexity, access vector, authentication impact, integrity impact, confidentiality impact and availability impact. The Temporal Metrics group deal with exploitability, remediation level and report confidence. The Environmental Metrics group deal with collateral damage and target distribution [175].

3.2.1 Service Conditions

Many VoIP systems employ the Session Initiation Protocol (SIP), However, network topology can impose a problem to a SIP-based VoIP system. A SIP-based VoIP call cannot be established if one of the SIP softphones is situated behind a NAT gateway or behind a restrictive firewall [185]. This is referred to as the NAT and firewall problem.

Firewalls normally allow incoming traffic from external hosts only if the session was initiated from the internal network. Incoming calls from untrusted external sources are filtered out by firewalls. When user uses a private IP address and local UDP port to receive voice for SIP call, voice packets from the remote party connected to public Internet will never reach the user because private IP addresses are not routable in the public Internet. Since NATs and firewalls prevent SIP and H.323 signals from reaching any devices several possible solutions have been proposed. For example, Interactive Connectivity Establishment (ICE), Session Traversal Utilities for NAT (STUN) and Traversal Using Relay NAT (TURN) were proposed to overcome the NAT and firewall problem within SIP and H.323 protocols [100, 125, 177]. Another property of NATs is the port mapping is kept only if there is traffic in both directions. For example if Alice is in a call with Bob, and for a while only Bob talks, then Alice's NAT may close the mapping, which effectively terminates the call.

On the other hand, Skype, a popular peer-to-peer VoIP application, is able to traverse NAT and bypass firewalls, as well as induce a bandwidth burden due to the supernode mechanism. Any node with a public IP address, having sufficient CPU, memory and network bandwidth is a candidate to become a supernode. Each Skype client stores a list of supernode IP addresses and port pairs and buddies list in the Windows registry of its host [146] . This makes Skype a considerable threat to both security and availability of enterprise networks. Enterprises IT managers must be made aware of the situation.

Google Talk is based on eXtensible Messaging and Presence Protocol (XMPP) technology. Core XMPP is stable but XMPP community continues to define various XMPP extensions. For example jingle, is added to overcome the NAT and firewall problem. XMPP extensions

may not be part of the standardization therefore may change as XMPP community works to standardize features [57].

Information regarding YMSG protocols are less transparent. Firewalls and proxy servers cause the majority of YMSG connection problems, but it can be configured to get through most firewalls and proxies without too much trouble [67]. The ports and servers are easily configurable to use something else including HTTP port 80 and FTP port 21. YMSG uses different ports for different applications. For example, TCP 5050 is for chat and messenger, UDP 5000-5010 for voice chat and 5100 for webcam. Blocking instant messengers ports are difficult because YMSG like any other instant messengers will try a standard range of preconfigured ports if they find their standard ports are blocked

It seems that Skype, SIP, YMSG and Google Talk have their own solutions dealing with NAT and firewall problems. This emphasises the need for enterprises to strengthen their network and deploy strategic IT policies to protect and minimise securities exploitation by their own employees. The two main factors in the security protocol that affect the call setup delay especially during network congestion were the security handshake and message authentication. The security channel is established during security handshake. This creates an additional delay in the course of generating a key and exchanging messages. SIP messages are encrypted for confidentiality and authentication. In this procedure the delay is due to the increased packet size and computation necessary for encryption [25].

Ranganathan analysed the effect of employing encryption and authentication algorithms and the additional effect caused by the dynamic key exchange algorithms, IKE, on the VoIP network performance [134]. The results of the performance analysis are:

i) Between encryption and authentication, encryption is more expensive operation.

ii) When both encryption and authentication services are employed there is 1.4% increase in SIP call setup times and there is an increase of 1.6% in media stream delays.

Moreover the research conducted by Ranganathan showed that the 1.4% increment on the call setup and 1.6% increment on media stream delays, due to encryption and authentication are insignificant to determine the effect of security implementation to a voice quality.

The research in [25] demonstrated the effect of a security protocol on the performance of SIP particularly on the call setup delays that occur within the three security protocols (i.e. TLS, DTLS and IPSec) and the three transport protocols (i.e. TCP, UDP and SCTP). UDP with any combination of security protocols performs much better than the combination of TCP. TLS/SCTP may have more effect on the performance on average. DTLS has been proposed as a replacement for TLS. DTLS is designed to operate on unreliable transport protocols like UDP and is not strict about the sequence of messages. For connectionless transport layer protocol, DTLS/UDP suprisingly performed better than UDP/IPSec. DTLS/UDP and UDP/IPSec

are the best performers among the combinations of popular security protocols in different layers. However, UDP has a side effect of high failure rate for a call setup because of the lack of congestion control.

A defined process is essential in order to properly instrument and compare different VoIP scenarios and security schemes. Before deciding on test case scenarios, the similarities and the differences of each VoIP service are identified. Table 3.1 on page 82 shows the differences and similarities of the services, among other thing pertaining to call managements, network types and security features. These characteristics affect the performance of each service. Unfortunately users have less control over these features. However, with the right information, users can choose services that are more reliable and secure. Several factors affect the quality of a VoIP call. These factors include the speech codec, packetisation, packet loss, delay, jitter, and the network architecture to provide QoS [56, 17]. A summary of the factors is presented in Figure 3.2 on page 81. Test scenarios should be selected to stress these parameters. Subsection 3.3.1 on page 83 discusses about the test scenarios.

3.2.2 Scope

The project focuses on the quality of the media part of VoIP services rather than the call signalling. We would not verify any security protocol's features. We assume the correctness of the features suggest in the reference documents of TLS, SRTP and IPSec [41, 10, 104, 85]. For example, TLS protocol provides communications security over the Internet. The protocol allows client-server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. On the other hand, SRTP can provide confidentiality, message authentication, and replay protection to the RTP traffic and to the control traffic for RTP. IPSec is a suite of protocols designed to provide end-to-end security at the network layer using encryption and authentication techniques. IPSec can be composed of one or both of the following network layer protocols: Encapsulated Security Payload (ESP) and Authentication Header (AH) . ESP and AH provide two mechanisms for protecting data being sent over an Internet Security Association and Key Management Protocol (ISAKMP). ESP can be used for integrity and confidentiality and AH for integrity only. In this project we use Vyatta routers. The routers support IPSec ESP protocol [173]. Currently they do not support AH protocol. For this project only three VoIP applications are tested. They are Skype, Google Talk and SIP-based clients.

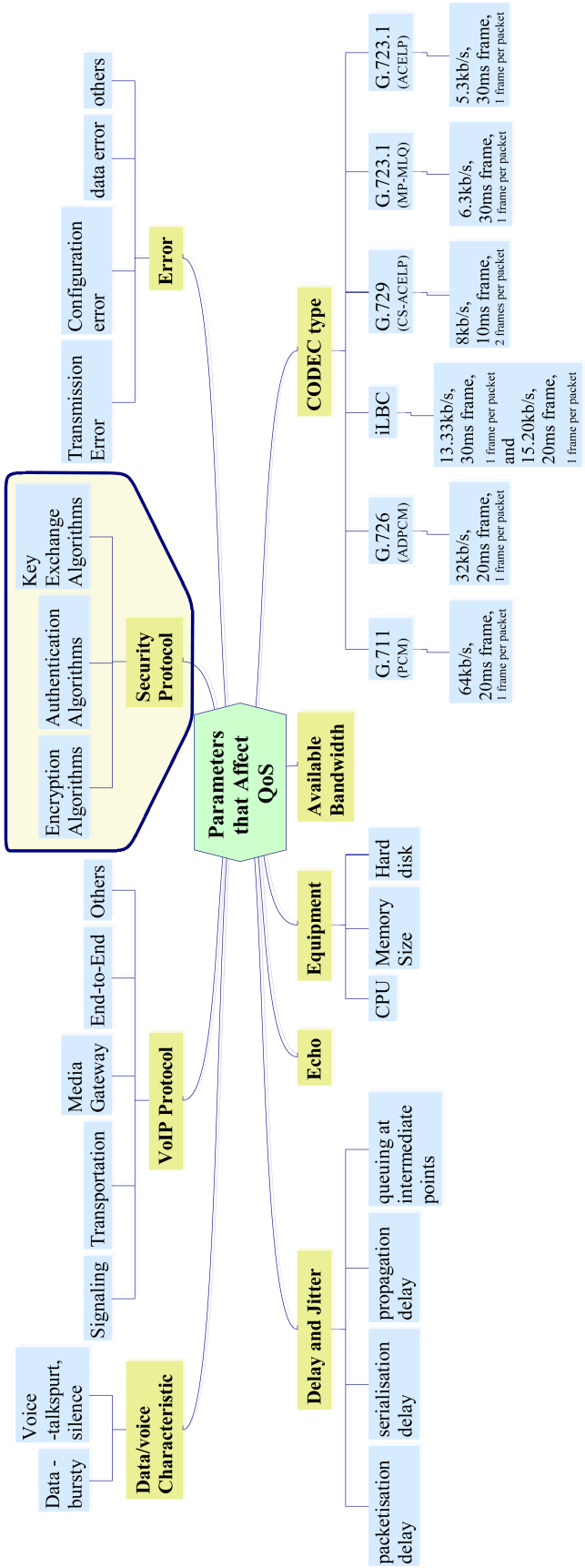


Figure 3.2: Parameters that affect QoS

Table 3.1: Different Types of VoIP Services

Service Name	Main Service	Call Management	Network Type	Servers / Intermediate Devices	NAT and Firewall	Transport	Security Features
Express Talk	VoIP	SIP	P2P telephony	SIP Proxy, Redirect Server and Registration and Admission Server	cannot run behind corporate firewalls and NAT	RTP and SRTP over UDP or TCP	DoS prevention, user-to-user authentication and user-to-proxy authentication
SKYPE	VoIP	Proprietary	P2P telephony	Super Node	can bypass corporate firewalls and NAT	Proprietary	Global Decentralised User Directory, Encryption of End-to-end instant messages
Google Talk	Instant Messaging	XMPP	Client-Server, Communication on port 5222 and 5269 for Client-to-Server and Server-to-Client respectively	Web and application servers	cannot run behind corporate firewalls and NAT (however, is possible with jingle)	TCP or UDP	Simple Authentication and Security Layer-Mutual Authentication, Integrity checking
YMSG	Instant Messaging	Proprietary	Client-Server, communication on default port 5050 and others	Web and application servers	cannot run behind corporate firewalls and proxies but can run behind PAT and NAT.	TCP or UDP	User message is sent in binary format, Authentication on user login, VoIP handled by Yahoo Server

3.3 Testing Method

This section defines the testing method. This procedure is important to ensure that the test is executed in a systematic manner, in order to ensure the integrity of data collected for the experiment. First, test scenarios are identified. Several test scenarios to stress the early observation are described in Subsection 3.3.1 on page 83. Next, test parameters are defined. Subsection 3.3.2 on page 84 discusses the test parameters. Then, Subsection 3.3.3 on page 84 describes an outline procedure for the testing, analysis and verification processes. The detail processes are refined in Section 4.5 on page 99 and Section 4.6 on page 103.

3.3.1 Test Scenarios

This subsection describes the test scenarios that are required to stress the parameters that are presented in Figure 3.2 on page 81. The voice quality of pre-recorded message[4] is compared to post-recorded message[5] using PESQ method. The method is described in Subsection 2.5.5 on page 69.

Although ITU-T P.862 indicated that the correlation coefficient between the objective and subjective scores were 0.935 for both known and unknown data, the PESQ algorithm cannot be used to replace subjective testings [73]. The PESQ algorithm does not provide a comprehensive evaluation of transmission quality. It only measures the effects of one-way speech distortion and noise on speech quality. It is possible to have high PESQ scores, yet poor quality of the connection overall. Even with the setback, PESQ is still the right tool to find the effects of delay, packet loss rate, jitter and additional delay due to the extra security at network layer, transport layer and application layer and codec like G.711, iLBC and Speex on voice quality on each VoIP service for this project. This is because the setback lies at the gateway of PSTN and Internet connectivity whereas this project concentrates purely on PC-to-PC VoIP services. According to Hermann et al., one-way delay and jitter implies voice quality distortion. By introducing delay, jitter and other parameters into the packet interceptor, several degraded signals could be captured [68]. The analysis on these signals would give good indicator how these parameters affect voice quality. The test scenarios are described as follows: Compute the two response variables $pesqmos$ (i.e. PESQ Mean Opinion Score) and $crude\_delay$ (i.e. the delay due to the crude alignment between the degraded audio and the original audio) for three VoIP services:SIP-based client, Google Talk and Skype on a secure and on an unsecure network varying values of the independent variables $delay$, $jitter$ and plr (i.e. packet loss rate).

The value of these variables are varied as follows:

[4]original message
[5]degraded message

$delay$ has values between 0ms and 400ms and is increased in steps of 10ms, and
plr has values between 0% and 3% and is increased in steps of 0.1%, and
$jitter \in$
$\{jitter0, jitter1, jitter2, jitter3, jitter4, jitter5, jitter6, jitter7, jitter8, jitter9\}$
where
$jitter0$= (0.33 of 100ms delay, 0.33 of 200ms delay, 0.33 of 300ms delay)
$jitter1$= (0.20 of 100ms delay, 0.30 of 200ms delay, 0.50 of 300ms delay)
$jitter2$= (0.50 of 100ms delay, 0.30 of 200ms delay, 0.20 of 300ms delay)
$jitter3$= (0.30 of 100ms delay, 0.50 of 200ms delay, 0.20 of 300ms delay)
$jitter4$= (0.20 of 100ms delay, 0.50 of 200ms delay, 0.30 of 300ms delay)
$jitter5$= (0.50 of 100ms delay, 0.20 of 200ms delay, 0.30 of 300ms delay)
$jitter6$= (0.30 of 100ms delay, 0.20 of 200ms delay, 0.50 of 300ms delay)
$jitter7$= (0.05 of 100ms delay, 0.05 of 200ms delay, 0.90 of 300ms delay)
$jitter8$= (0.05 of 100ms delay, 0.90 of 200ms delay, 0.05 of 300ms delay)
$jitter9$= (0.90 of 100ms delay, 0.05 of 200ms delay, 0.05 of 300ms delay)
$jitter0..9$ are not standard but variation of delays that have been introduced by creating three pipelines with different probability of delays to see the effect of the independent variable $jitter$ to each VoIP service.

3.3.2 Test Parameters

This subsection describes the input parameters and the output parameters of this project. Figure 3.3 on page 85 highlights these parameters. The input parameters are clasified into three categories such as network performance variables, VoIP applications and Network layer security protocol. Network perfomance variables like packet loss rate, delay, jitter and network bandwidth are fed into a packet interceptor program: Dummynet. VoIP application: Skype client, SIP-based client and Google Talk client are installed into Client 1 and Client 2 computers, refer to Figure 4.1 on page 90. IPIP and IPSec tunnels are configured on R1, R2 and TR routers. Semi-automated test-kit scripts control what parameters are to be tested and how long would they run. These scripts are discussed in details in the fourth stage of the implementation stages in Section 4.4 on page 97.

3.3.3 Testing, Analysis and Verification

To start the testing the sender is fed with raw audio that has been stored using a loss-less coding scheme. This prevents results that would be artefacts of the codec rather than the network. Then the message is recorded at the receiver end and is compared with the sent audio.

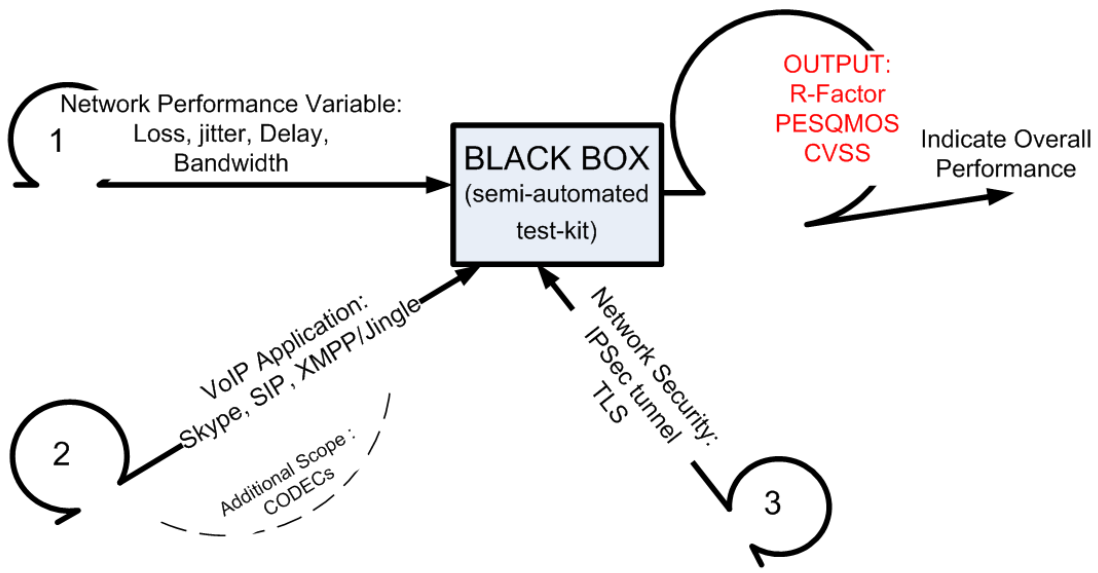


Figure 3.3: Experimental test parameters

An appropriate holding time[6] is required [135, 106]. A shorter holding time is desirable as it allows more tests to be run. However, it should be long enough to assure reliability in the evaluation. In this project the holding time is set to 10 seconds.

The framework should allow the comparison of several VoIP services over several network scenarios. This will be compared using the E-model [24] to estimate the R-factor of each service:

$$R = 93.4 - Id(t_i - t_{i-1}) - Ie(codec, loss)$$

Where:

- $Id(t_i - t_{i-1})$ is a function of one-way delay
- $Ie(codec, loss)$ is a function of codec and packet loss rate

and R-factor to MOS conversion according to [24] :

$$\text{For } R < 6.5 : MOS = 1$$
$$\text{For } 6.5 \leq R \leq 100 : MOS = 1 + 0.035R + 7.10^{-6}R(R-60)(100-R)$$
$$\text{For } R > 100 : MOS = 4.5$$

[6]test signal length

The overall performance trends of the network can be analysed looking at the graph of TCP, UDP and ICMP protocols. The CPU and memory load for the call can also be captured. Results can be compared with other studies by Ranganathan and Kilmartin on secure SIP based VoIP networks [134].

3.4 Summary

This project is to measure selected VoIP services for the benefit of Internet users and any company IT managers. These services are popular among their users, however, there is still a lack of studies on the security impact on the quality of their services. The approach is to have a hybrid architecture whereby a live testbed is complemented with a packet interceptor simulator to create a suitable environment for comparing the performance of different VoIP services. This approach allows full control of the parameters set in the experiment. The method can be carried out for other IP-based and VoIP services.

The process starts with the identification of test scenarios to verify results. In order to identify the VoIP performance the following parameters are measured: End-to-end delay, Jitter, Packet Loss Rate, Network Resources and Network Trend based on protocol statistics. The overall performance is estimated based on R-factor score of the VoIP QoS and CVSS Based Metrics score for the security. The project scope defines which items to be included or to be excluded from the project, with respect to VoIP services, security features and test cases. PESQ and E-model are the two tools used to measure the MOS values of the degraded audios and to predict the network quality, respectively. CVSS is used to calculate the vulnerability scores on OpenVPN TLS and IPSec. Next, Chapter 4 describes the project implementation.

Chapter 4

Project Implementation

"I hear and I forget. I see and I remember. I do and I understand."

Confucius

This chapter explains the project implementation stages. The implementation of this project is divided into six stages. The first stage is to setup a network in the laboratory. The network is connected to Internet via JANET, the UK's education and research network. The network is secured either by a site-to-site TLS openVPN or a site-to-site IPSec in tunnel mode. The second stage is to setup monitoring tools and packet interceptor. The monitoring tools are configured in two different computers, CACTI and TCPdump. Dummynet, the packet interceptor is installed into the third computer. The third stage is to install VoIP clients. SIP softphone, Google Talk and Skype are installed into sender and receiver machines. The fourth stage is to write scripts to automate the testing. The scripts are used to control the test parameters.

The fifth stage is to collect and process data. The audio streams are collected and saved as .wav files. In addition to that, data packets transmitted from sender to receiver are captured and saved as .pcap files. The PESQ software compares the original audio and the collected audio streams and produces a .txt report. The .txt is transformed into the formats that are recognized by R. R is used to plot graph of mean and box and whisker plot of the MOS. Ipsumdump and Ipaggcreate scripts transform .pcap files into .txt files. Ipsumdump collates and filters all IP packets travel from sender to receiver. Ipaggcreate aggregates IP packets travel from sender to receiver. This information is later used in calculating the network throughput for VoIP connectivity. Internet Control Message Protocol (ICMP) ping is used to find the statistic of the round trip time between each connectivity which is required in estimating the R-factor. The last stage is the validation and verification, where the PESQ, E-Model and CVSS calculators are validated, and the data collected and the sample result produced are verified.

4.1 First Stage: Network

The testbed is set as shown in Figure 4.1 on page 90. In addition, Figure 4.2 on page 91 shows the laboratory setup overall view. R1, R2, and R3 are the routers that are placed in the laboratory. R1 and R2 are connected to R3. A transit router, TR*, theoretically can be placed anywhere in the globe. For this implementation TR* is situated in the laboratory as well. HP, IT and Router-to-JANET are routers that are administered by the Loughborough IT services. They are located within Loughborough University but at different locations to R1, R2, R3 and TR*. A bidirectional IPIP tunnel is created between R1 and TR*. There is also a bidirectional IPIP tunnel from R2 to TR*, refer to Table 4.3 on page 94 for the setup. IPSec tunnels are configured to secure the IPIP tunnels. Another implementation is to have a site-to-site OpenVPN tunnel between R1 and R2 with TLS as an authentication and key exchange mechanism.

IPSec could be carried out either in a transport mode or a tunnel mode. Transport mode is allowed between two end hosts only, whereas tunnel mode is required when at least one endpoint is a security gateway. A security gateway is an intermediate system that implements IPSec functionality (i.e. a router). In this project, IPSec is configured in a tunnel mode between R1 and R2 routers. In the 90s, most VPN is carried out using IPSec with Internet Key Exchange (IKE) protocol to be used with Internet Security Association and Key Management Protocol (ISAKMP) protocol. ISAKMP provides a framework to establish security associations. A security association (SA) is a bundle of security features that are shared between two peers to support a secure communication. The SA may include properties such as cryptographic algorithm and mode, traffic encryption key and parameters for data network to be passed over the secure connection. ISAKMP is designed to be key exchange independent. IKE is a profile of ISAKMP. IKE provides authenticated keying material for use with ISAKMP. It consists of two phases. In the first phase, IKE establish a secure authenticated communication channel by using the Diffie-Hellman key exchange algorithm to generate a shared secret key to encrypt further IKE communications. This negotiation results in one single bi-directional security association. The authentication can be performed using either a pre-shared secret key or other mode like a public key encryption. During the second phase, the IKE peers use the secure channel established in the first phase to negotiate SA on behalf of other services like IPSec. IKE SA is bi-directional but AH or ESP SA is unidirectional. Therefore, in a bi-directional traffic, the channels are secured by a pair of security associations. For this experiment, we used a pre-shared secret key mode, refer to Table 4.4 on page 94 for the setup. We are aware with the setback of the site-to-site IPSec with a pre-shared secret key, whereby each endpoint that involves in the communication must keep the other endpoint pre-shared key. As a result, if there are n IPSec tunnels, network administrator must remember to main-

tain n pre-shared secret keys from time to time to reduce the possibility of compromised keys. A network administrator must relies and trusts another network administrator to execute the task of exhanging a pre-shared secret key at a remote site. The pre-shared secret keys are exchanged through email or phone. Due to this condition, a site-to-site IPSec with preshared key is not scalable for a large number of tunnels. In our case, the secure channels are between two endpoints only.

OpenVPN uses TLS with X.509 certificates, and requires a public key infrastructure (PKI) to generate the certificates. Vyatta has several ways to authenticate a pair of virtual tunnel. For example, using a pre-shared secret key or TLS certificate. TLS is a cryptographic protocol that uses a public key cryptography and does not require the two endpoints to have a pre-shared secret. Using PKI, the administrator generates a certificate and the associated files for each endpoint. All certificates are signed by the certificate authority (CA) of the PKI. The certificate for an endpoint contains many pieces of information, one of which is the endpoint's name, which is stored in the Common Name field of the certificate. The administrator transfers each certificate and the associated files to the corresponding endpoint using a pre-established, secure channel. When two endpoints want to establish the VPN tunnel, one takes a passive role whiles the other endpoint must take an active role and initiate the TLS session with the passive endpoint. Once the active endpoint initiates the TLS session, the two sides authenticate each other using their public/private key pairs and the CA's public key, which is known to both endpoints. After the two endpoints have authenticated each other, they establish a shared secret using a public key cryptography. Each endpoint then derives a set of keys for the session. These keys are then used for encryption and Message Authentication Code (MAC) on the tunnel data to provide data confidentiality and integrity. These keys are only used for the one session, and therefore they are called session keys. In this project a site-to-site OpenVPN tunnel with TLS is configured between R1 and R2 routers, refer to Table 4.5 on page 95 for the setup, whereby R1 takes the passive role whereas R2 takes the active role. A site-to-site TLS OpenVPN is more scalable than a site-to-site IPSec tunnel with pre-shared secret key mode. This is because for each router under his responsibility, a network administrator needs to know the routers public/private key pairs and the CA public key only.

We have configured three virtual LANs, namely VLAN2, VLAN3 and VLAN4 in a DELL switch. The DELL switch is managed through VLAN1, if and when required. Alternatively, the DELL switch could also be managed through a web-link using the url to the switch management IP address. There are twenty four ports on the DELL switch. The ports are labelled as g1, g2, g3 upto g24. All ports are in VLAN1 unless they are assigned to other VLAN. All devices that assist in system monitoring (i.e. CACTI, TCPdump and Dummynet) are in VLAN4 with subnet6. The DELL switch is connected to R1 via port g7. CACTI, is connected to the DELL switch of port g8. CACTI is used for performance monitoring. It is a complete

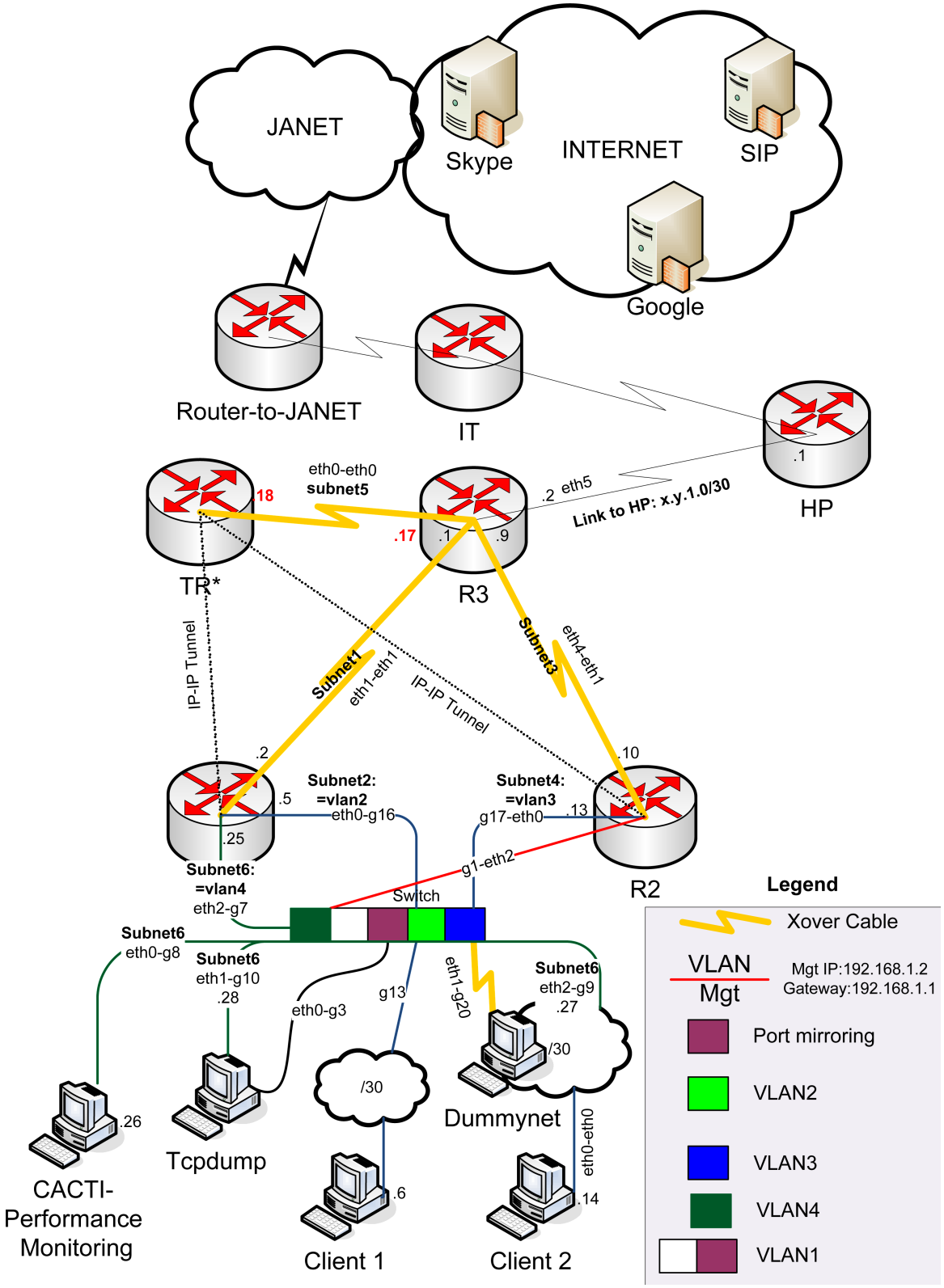
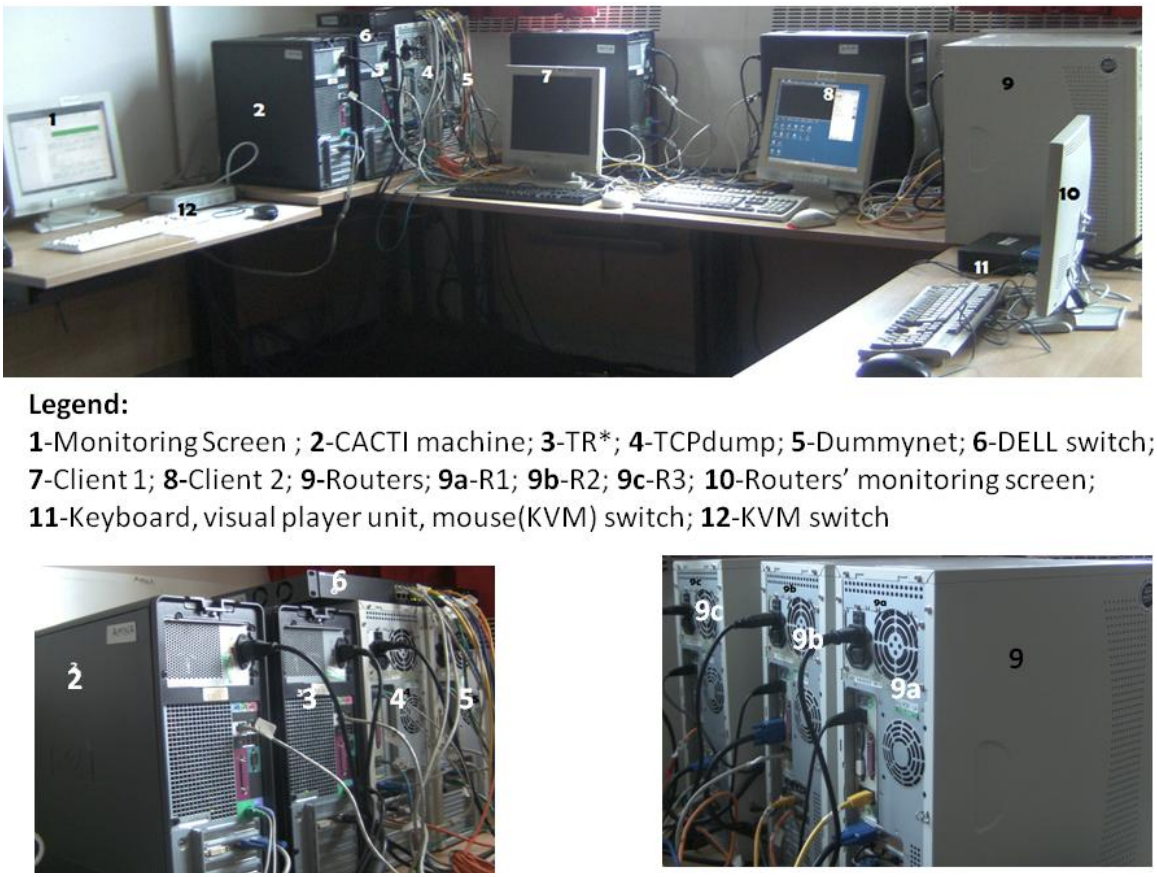


Figure 4.1: Experimental Network



Legend:
1-Monitoring Screen ; **2**-CACTI machine; **3**-TR*; **4**-TCPdump; **5**-Dummynet; **6**-DELL switch; **7**-Client 1; **8**-Client 2; **9**-Routers; **9a**-R1; **9b**-R2; **9c**-R3; **10**-Routers' monitoring screen; **11**-Keyboard, visual player unit, mouse(KVM) switch; **12**-KVM switch

Figure 4.2: Laboratory Setup:Overall view

Legend:
7-Client 1; **8**-Client 2;
6-DELL switch;
1-Monitoring Screen; **12**-KVM switch;
10-Routers' monitoring screen; **11**-KVM switch;

Figure 4.3: Specific devices:Front view

frontend to RRDTool, it stores all the necessary information to create graphs and populate them with data in a MySQL database [22]. The packet sniffer (i.e. TCPdump) computer is connected to port g10 and the packet interceptor a.k.a Dummynet is connected to port g9.

Client 1 and Client 2 are in two differents subnets and VLANs. Client 1 is in VLAN2 with subnet2 and Client 2 is in VLAN3 with subnet4. Client 1 is connected to the switch port g13. Router R1 is connected to the switch port g16. Client 2 is connected to the ethernet port eth0 of the packet interceptor. The packet interceptor is then connected to the switch port g20 with a crossover cable. Router R2 is connected to the switch port g17. The g16 and g17 ports are the gateways to VLAN2 and VLAN3 respectively. The g13, g16, g17 and g20 ports on the switch is mirrored to the g3 port. The g3 port of the switch is connected to the TCPdump ethernet port eth0. Any packets going through these ports are captured by TCPdump. These packets can then be analysed by TCPtrace or Ipsumdump passively. The problem of time syncronization between R1, R2, Client 1 and Client 2 is solved since all traffic is mirrored to the g3 port. In addition an audio cable is connected between Client 1 microphone and Client 2 speaker. Any degraded audio is recorded back into Client 1 machine. Client 1 and Client 2 are installed with Express Talk Business Edition version 4.2.6, Google Talk version 1.0.0.104 and Skype version 4.2.0.158 clients. Virtual Audio Cable (VAC) version 4.9, Sound eXchange (SoX) version 14.3.1 and Audacity version 1.2 software are installed in Client 1 and Client 2 as well. A raw audio using a loss-less coding scheme is recorded using Audacity. This prevents results that would be artefacts of the codec rather than the network. VAC and SoX are explained in Section 4.4 on page 97. Both Client 1 and Client 2 have the following specification: Microsoft Windows XP Professional, version 2002 with service pack 3, 3.20GHz Pentium (R) 4 CPU, 3.19GHz, 1.00GB of RAM and 250GB Hard disk.

Before any configuration can be made, a /27 IPv4 network addresses is subdivided into a smaller network. Table 4.2 on page 94 shows the details of these subnets. Table 4.3 and Table 4.4 on page 94 show the IPIP tunnel setting and the site-to-site IPSec tunnel setting respectively. Table 4.5 on page 95 shows the site-to-site OpenVPN with TLS setting. The routers and switch configurations' scripts that are mentioned in this subsection are available in the Appendix A.1 on page i and xxi respectively.

4.2 Second Stage: Monitoring Tools and Packet Interceptor

There are three computers that are installed with the monitoring tools and packet interceptor. The first computer is labelled as CACTI. The second computer is labelled as TCPdump. The computer with packet interceptor is labelled as Dummynet. Figure 4.1 on page 90 shows the experimental network with the locations of these computers with respect to other equipments. Figure 4.2 on page 91 shows the position of these computers in the laboratory. These com-

Table 4.1: Device Specification

Item	Description	Operating System	System Specification
1	Client 1	Microsoft Windows XP Professional, version 2002 with service pack 3	3.20GHz Pentium (R) 4 CPU, 3.19GHz, 1.00GB of RAM and 250GB Hard disk
2	Client 2	Microsoft Windows XP Professional, version 2002 with service pack 3	3.20GHz Pentium (R) 4 CPU, 3.19GHz, 1.00GB of RAM and 250GB Hard disk
3	TCPdump	Linux 2.6.24-29-generic i686 Distributor ID: Ubuntu Description: Ubuntu 8.04.4 LTS Release: 8.04 Codename: hardy	Model Family: Western Digital Caviar family Device Model: WDC WD400BB-00DEA0 Serial Number: WD-WMAD14441462 Firmware Version: 05.03E05 User Capacity: 40,020,664,320 bytes
4	Dummynet	Linux 2.6.32-14-generic-pae i686 Distributor ID: Ubuntu Description: Ubuntu 10.04 LTS Release: 10.04 Codename: lucid	Model Family: Western Digital Caviar family Device Model: WDC WD400BB-00DEA0 Serial Number: WD-WMAD11688819 Firmware Version: 05.03E05 User Capacity: 40,020,664,320 bytes
5	CACTI	Linux 2.6.35-28-generic x86_64 Distributor ID: Ubuntu Description: Ubuntu 10.10 Release: 10.10 Codename: maverick	Model Family: Western Digital Caviar SE Serial ATA family Device Model: WDC WD1600JD-75HBC0 Serial Number: WD-WMAL93844163 Firmware Version: 08.02D08 User Capacity: 160,000,000,000 bytes
6	Router 3	Linux 2.6.35-1-586-vyatta i686	Version: VC6.2-2011.02.09 Description: Vyatta Core 6.2 2011.02.09 Copyright: 2006-2011 Vyatta, Inc. Built by: autobuild@vyatta.com
7	Router 2	Linux 2.6.35-1-586-vyatta i686	Version: VC6.2-2011.02.09 Description: Vyatta Core 6.2 2011.02.09 Copyright: 2006-2011 Vyatta, Inc. Built by: autobuild@vyatta.com
8	Router 1	Linux 2.6.35-1-586-vyatta i686	Version: VC6.2-2011.02.09 Description: Vyatta Core 6.2 2011.02.09 Copyright: 2006-2011 Vyatta, Inc. Built by: autobuild@vyatta.com
9	Router TR*	Linux 2.6.35-1-586-vyatta i686	Version: VC6.2-2011.02.09 Description: Vyatta Core 6.2 2011.02.09 Copyright: 2006-2011 Vyatta, Inc. Built by: autobuild@vyatta.com
10	Dell Switch		Dell PowerConnect 5324

Table 4.2: Project IP Addresses

Subnet	From	Destination	Port Interface	Description
Management of VLANs	192.168.1.1/24	192.168.1.2/24	eth2-g1	R2 to VLAN1
Link to Internet	172.x.1.2/30	172.x.1.1/30	eth5-	R3 to HP
Maintain by IT services				HP to IT IT to Router-to-JANET
Subnet1	158.125.253.2/30	158.125.253.1/30	eth1-eth1	R1 to R3
Subnet2	158.125.253.5/30 158.125.253.6/30		eth0-g16 eth0-g13	R1 to VLAN2 Client 1 to VLAN2
Subnet3	158.125.253.10/30	158.125.253.9/30	eth1-eth4	R2 to R3
Subnet4	158.125.253.13/30 158.125.253.14/30		eth0-g17 eth1-g20 eth0-eth0	R2 to VLAN3 Dummynet to VLAN3 Client 2 to Dummynet
Subnet5	158.125.253.18/30	158.125.253.17/30	eth0-eth0	TR* to R3
Subnet6	158.125.253.25/29 158.125.253.26/29 158.125.253.27/29 158.125.253.28/29		eth2-g7 eth0-g8 eth2-g9 eth1-g10	R1 to VLAN4 CACTI to VLAN4 Dummynet to VLAN4 TCPdump to VLAN4

Table 4.3: IPIP tunnel

Interface	local-ip	remote-ip	IP address	Description
lo lo lo	10.0.5.10/24 10.0.6.2/24 10.0.1.18/24 10.0.2.18/24			Loopback on R2 Loopback on R1 Loopback on TR*
tun0	10.0.5.10	10.0.1.18	10.0.3.10/24	IPIP tunnel R2 to TR*
tun0	10.0.6.2	10.0.2.18	10.0.4.10/24	IPIP tunnel R1 to TR*
tun0 tun1	10.0.1.18 10.0.2.18	10.0.5.10 10.0.6.2	10.0.3.18/24 10.0.4.18/24	IPIP tunnel TR* to R2 IPIP tunnel TR* to R1

Table 4.4: Site-to-site IPSec tunnel

Interface	peer	local-ip	tunnel	Description
eth1 on R2	158.125.253.2	158.125.253.10	tunnel1	esp-group ESP-IPIP ike-group IKE-IPIP local-subnet 10.0.5.10/32 remote-subnet 10.0.6.2/32 pre-shared-secret Allahuakhbar
eth1 on R1	158.125.253.10	158.125.253.2	tunnel1	esp-group ESP-IPIP ike-group IKE-IPIP local-subnet 10.0.6.2/32 remote-subnet 10.0.5.10/32 pre-shared-secret Allahuakhbar

Table 4.5: Site-to-site OpenVPN TLS

Interface	local-address	remote-address	remote-host	Description/Other Requirement
vtun0	10.18.1.1	10.18.1.2	158.125.253.10	R1 to R2 Common Name is RHzgrave ca-cert-file /*path-to-key*/ca.crt cert-file /*path-to-key*/RHzgrave.crt dh-file /*path-to-key*/dh1024.pem key-file /*path-to-key*/RHzgrave.key role passive static interface-route to 158.125.253.4/30 next-hop-interface vtun0
vtun0	10.18.1.2	10.18.1.1	158.125.253.2	R2 to R1 Common Name is RRHzgrave ca-cert-file /*path-to-key*/ca.crt cert-file /*path-to-key*/RRHzgrave.crt key-file /*path-to-key*/RRHzgrave.key role active static interface-route to 158.125.253.12/30 next-hop-interface vtun0

puters are installed with LINUX UBUNTU operating systems. Table 4.1 on page 93 list out the devices specification.

CACTI is a complete frontend to RRDTool. CACTI stores all of the necessary information to create graphs and populate them with data in a MySQL database [21]. In this project SNMP traffics are polled and are used for creating traffic graphs. The prerequisites for CACTI are RRDTool, PHP, MySQL and Apache Web server [20]. The RRDTool is available from RRDTool homepage [121]. RRDtool is the OpenSource industry standard, high performance data logging and graphing system for time series data. It supports custom monitoring shell scripts or create whole applications using its Perl, Python, Ruby, TCL or PHP bindings. Installing RRDTool is trivial. That is to extract the downloaded RRDTool source code into the CACTI directory. PHP, MySQL and Apache server could be installed separately. Instead in this project LAMP (LINUX, Apache, MySQL and Phyton) is installed. LAMP consists of PHP, MySQL and Apache server for LINUX. The process of installing CACTI is available from Cacti manual 0.8.7 [20].

The tcpdump packet sniffer has been used for a long time and is the basis for most other open source packet sniffers. As its name implies, tcpdump collects and dumps data on TCP/IP networks. Most LINUX distributions come with tcpdump installed by default or it can be obtained from its official web site [111]. The tcpdump utility is a command-line tool. In this project tcpdump is installed into TCPdump computer. Dummynet is a tool originally designed for testing networking protocols, and since then used for a variety of applications

including bandwidth management. It could be used to enforce queue and bandwidth limitations, delays, packet losses, and multipath effects. It also implements a variant of Weighted Fair Queueing called WF2Q. It can be used on user's workstations, or on machines acting as routers or bridges. Dummynet is available from this official site [142]. In this project, Dummynet is compiled for LINUX 10.04. The kernel object ipfw mod.ko is installed into /*path-to-kernel*/../ netfilter. The bridge-utilities package is also installed. The package is available at Ubuntu.Package:bridge-utils website [163]. We use Dummynet to simulate delay, packet loss and jitter between Client 2 and router R2. The scripts for Dummynet is available in the Appendix A on page xxii.

4.3 Third Stage: VoIP clients

Installing and registering clients are trivial but nonetheless is essential for this project. As such it is highlighted here. Skype, Google Talk and SIP-based clients are installed into Client 1 and Client 2. The installers for Skype and Google Talk are available at their official homepages [153, 57]. For SIP-based clients, two types of SIP user agents are installed, Express Talk and Ekiga. Express Talk is a SIP softphone that works on personal computer [155]. Although Express Talk provides functionality for account creation and registration, however, it is not a SIP service provider. One needs to provide its own SIP servers (i.e. Registration, Authentication, Redirect and Proxies servers) for Express Talk to function as a SIP service provider. For example, Express Talk could be combined with a virtual Private Branch Exchange (PBX) system like Asteriks system in a company telephony suite to create a customize company telephony system. However, we use SIP account from a third party SIP service provider like Ekiga in Express Talk. This is to show that in general, a SIP URI is recognizable by another SIP User Agent. Ekiga is a GUI application and it provides SIP service. Ekiga executable version or source code can be downloded from Ekiga homepage [46].

In order for the clients to function properly, they are registered with the appropriate service providers. At least, two accounts of each client are required, one account is for Client 1 and another account is for Client 2. For Skype accounts the registrations are with Skype server. For Google Talk accounts the registrations are with Google Talk server and for SIP-based accounts the registrations are with Ekiga server. The next step is to create buddies (i.e. contacts) with each client. At the moment Skype users can only have Skype buddies. Although Skype SIP communications are also possible, it incurs extra charges. Google Talk users would have Google Talk buddies. SIP-based users can talk to any SIP-based buddies. With buddies system in placed, users can filtered out any unwanted users from their groups. Hence, this feature allows users to determine their own social group and a way of eliminating unwanted calls. For example, during the experiment, from time to time, several unfamiliar contacts will request to

join in our Skype contact group.

4.4 Fourth Stage: Script To Automate Experiment

There are three different parameters to be set on Dummynet. They are delay, packet loss rate (plr) and jitter profiles. Delay has a value which is between 0 ms to 400 ms. Plr is between 0% to 3% and there are ten jitter profiles. From these parameters set, there are more than 80 different test values to be tested on each VoIP service. Hence it takes plenty of time to run these tests manually. The idea here, is to run these tests repeatedly with less human intervention. Scripts are written for this purpose. The scripts run on a specific hardware. Figure 4.4 on page 99 shows the boundary of these scripts. Prior to running the scripts, Client 2 is set to have auto-accept incoming call and an audio cable is connected from the audio out port of Client 2 to the microphone port of Client 1, refer to Figure 4.4 on page 99. The script perform the following tasks in sequence:

1. Load IPSec or OpenVPN TLS configuration into R1 and R2 routers.
2. Set a testing parameter in Dummynet.
   - 2.1 Start tcpdump to dump packet transmits from Client 1 to Client 2 into TCPdump computer through ethernet port eth0.
   - 2.2 Activate VoIP client as a caller in Client 1.
   - 2.3 Make a call from Client 1 to Client 2.
   - 2.4 Once the call is successful, SoX plays audio message from Client 1 for 10 seconds.
   - 2.5 Using SoX to record the audio message back in Client 1 and save the recorded audio.
   - 2.6 Abort the tcpdump program.
3. Repeat step 2 for ten times.
4. Reset parameter to original setting in Dummynet (i.e. no packet filtering)
5. Set R1 and R2 routers to default state (i.e. enable the IPIP tunnel only)
6. Repeat step 1 to 5 for all test parameters.

Some setting are done on the SoX audio parameters to cater for the appropriate VAC cable ports before running the scripts. VAC is a software that transfer wave streams between applications and devices [116]. It creates a set of virtual audio devices named ”Virtual Cables”.

Each virtual audio device consists of a pair of the waveform input and output devices. Any application can send audio stream to an output side of a cable, and any other application can receive this stream from an input side. All transfers are made digitally. VAC is useful to record application's audio output in real time or transfer a sound stream for other application to process it. For example, VAC can be used to record calls and conversations of a VoIP application.

On Client 1, the AudioOUT parameter of SoX is set to one of the VAC cable ports. The microphone of Skype, Google Talk and Express Talk must also be set with the same VAC cable port. There are ten cable ports available at any one time. If the settings are done incorrectly, the audio is still played by the SoX player but through a wrong channel, resulting in no audio being passed to the clients' microphone. Similarly, on Client 2, the speaker of Skype, Google Talk and Express Talk are set to one of the VAC cable ports. In addition the AudioIN parameter of SoX is set to the same VAC cable port. If the settings are done incorrectly, the audio is still passed to the clients' speaker but through a wrong channel, resulting in no audio being recorded by the SoX recorder. Microphone and speaker are considered as audio devices. Most VoIP clients allow users to set these devices to suit their own requirements.

Figure 4.5 on page 100 shows the call time line. It takes about 15 seconds for one cycle of the tasks described on page 97 to be completed. In the diagram it shows that Client A is making a call to Client B. Client A is making the call from Client 1 and Client B is receiving the call at Client 2. All scripts are generated from Client 1. In order to perform these tasks, ssh servers are installed on the Dummynet and TCPdump computers. In addition ssh client is installed on a remote control computer. For this implementation the scripts are remotely controlled from Client 1. Hence ssh client is installed into this computer. Other important thing that should be highlighted here is the Skype client is activated through Skype library scripts from CPAN. When the Skype perl script is executed, it would attach to the Skype client that was installed earlier. The Skype perl script assists in the automation of the task of initiating call from Client 1 computer. However, if there is no CLI interface existed and only the GUI version of any clients existed then some improvision are applied. In this case, AutoIt is used to automate the action of making call from Client 1 and a script to receive call at Client 2. AutoIt codes are used to initiate calls for Google Talk and Express Talk clients.

Google Talk is based on XMPP/libjingle project. Google Talk source codes are in C++. The codes are available for downloaded from Google Talk for Developers site [57]. Several attempts have been made to compile the source codes on Microsoft Windows computer but failed because the prerequisite software GIPS VoiceEngine Lite is no longer available for public consumption. Therefore the software is downloaded from its official website, (i.e. http://www.google.com/talk/index.html) and is installed into Client 1 and Client 2 computers. To ensure that Google Talk executes smoothly, AutoIt is used to automate the task of

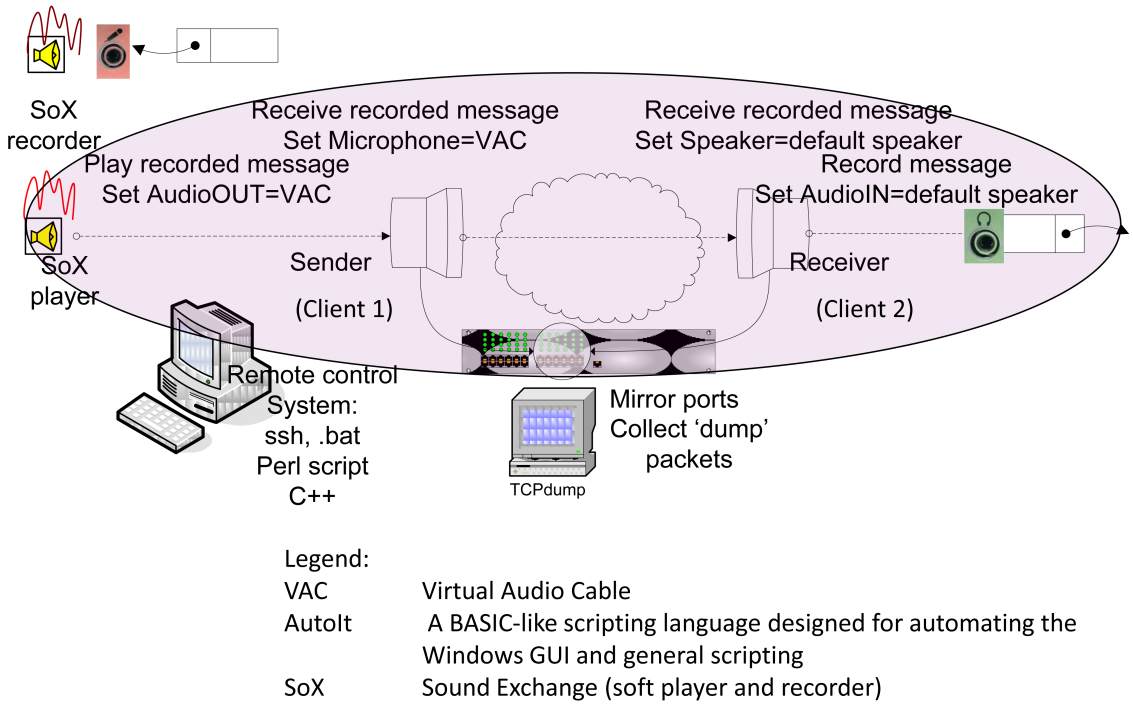


Figure 4.4: Scope for a semi-automated VoIP Test-kit

initiating call from Client 1 computer. AutoIt can be downloaded from Autoit v3 website [79]. Since Google Talk does not support auto-accept incoming call, an AutoIt script is written for this purpose. Similar codes are written for Express Talk. The scripts for the tasks mentioned in this subsection are available in the Appendix B on page xxvi.

4.5 Fifth Stage: Data Collection and Data Processing

This is one of the core stage of the implementation stages. The data collection and data processing processes are done in an orderly manner. This is essential because the collected data is used to evaluate the performance of each VoIP service. At the end of each data collection process some files with .wav and .pcap extentions are generated. A special script for PESQ run is prepared. Example of this script is available in Appendix C.1 on page xxxiii. Ipsumdump and Ipaggcreate scripts are prepared to generate .txt report from .pcap files. The report contains the time series information pertaining to IP packets and payload length of UDP segments travel from Client 1 and Client 2 computers, as shown in the sample below.

```
Epoch_time payload_size_in_byte
1307039254.818301 32
1307039254.818411 32
1307039254.832169 92
.....
.....
1307039953.693428 172
```

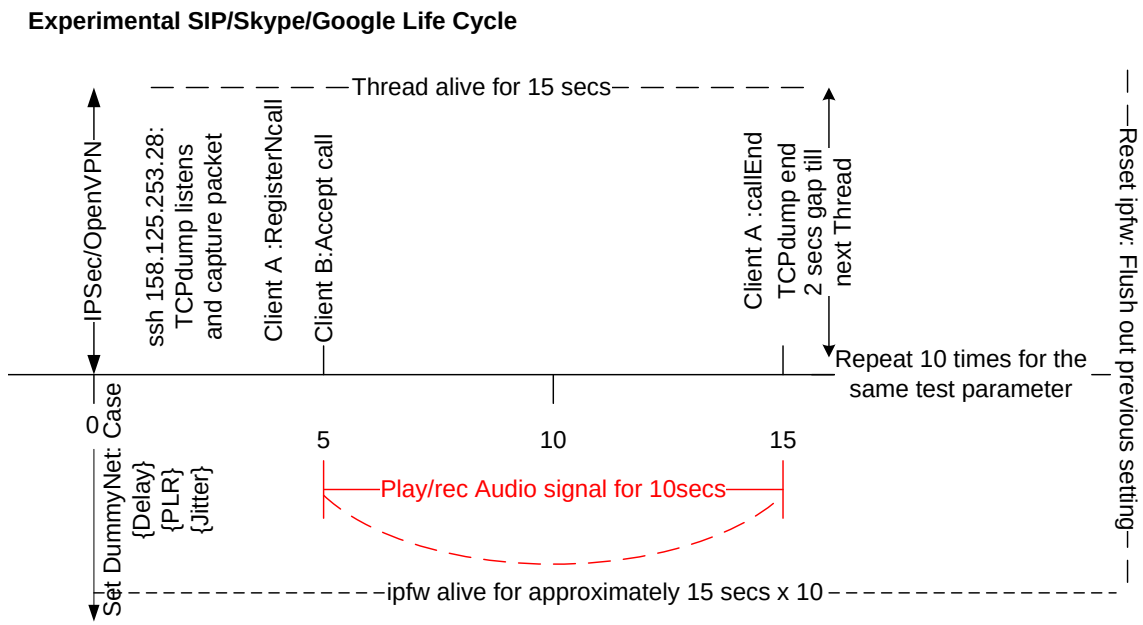


Figure 4.5: Call thread time line

```
1307039953.693582 172
```

Once the .wav data is collected then PESQ program is used to get the PESQMOS reading of the degraded audio. This audio is compared against the original audio message. In addition other details like crude_delay and sample frequency are also presented in the report. The device that is used for playing and recording the data must not distort the audio other than due to the parameters that are introduced in the scripts in the previous Section 4.4. The report produced by the PESQ script is in .txt format, as shown in the sample below.

```
REFERENCE DEGRADED PESQMOS PESQMOS SUBJMOS COND SAMPLE_FREQ CRUDE_DELAY
.......................
sample1234.wav Gtalk-delay0to90-2011-22-08-16-44-49.wav SQValue=2.845 2.845 0 0 16000 1.464
sample1234.wav Gtalk-delay0to90-2011-22-08-16-45-03.wav SQValue=2.811 2.811 0 0 16000 1.38
sample1234.wav Gtalk-delay0to90-2011-22-08-16-45-17.wav SQValue=2.815 2.815 0 0 16000 1.468
sample1234.wav Gtalk-delay0to90-2011-22-08-16-45-32.wav SQValue=2.766 2.766 0 0 16000 1.344
sample1234.wav Gtalk-delay0to90-2011-22-08-16-45-46.wav SQValue=2.841 2.841 0 0 16000 1.44
.......................
```

This report is transformed into .xls format that is recognised by R . It is used to produced box and whisker graphs and graphs of plot of mean.

The information from .pcap file is required to find the network throughput. In addition to the tcpdump report, RTT is extracted from ICMP log report. To generate the ICMP log, ping command is issued from Client 1 to Client 2 with different test parameters being setup in Dummynet. The RTT is one of the parameter required to estimate the R-factor score of E-model. The R-factor score objectively estimates the quality of any audio signal passing through the testbed. The R-factor score and the PESQMOS is cross validate between each other. The cross validation process is discussed further in Section 4.6 on page 103.

4.5.1 Data Collection

Data collection is a process of sending data to a central point from one or more locations. In this project there are two types of data that are being captured. The first one is the degraded audio signal in the form of .wav. The second one is the data packets that are propagated between Client 1 and Client 2 computers in the form of .pcap data. There are a few important points that should be highlighted here:

i. There should not be further loss of precision on the degraded audio signal except due to the testing parameters that are described in the Subsection 3.3.2 on page 84. Therefore the SoX player and SoX recorder frequencies are set to 96000 Hertz. This is higher than the frequency of the codecs used in the experiment.

ii. The holding time for the audio signal is set to 10 seconds. It was suggested that the recommended test signal length is between 8 seconds to 10 seconds for testing networks [135]. However, the long distance call mean of short call durations is 3.5 minutes and the long distance call mean for long call durations is 10 minutes on exponentially distributed call durations [105]. In this project the holding time is set to 10 seconds as shorter holding time allows more tests to be run.

iii. Data packets captured includes all activities happened between Client 1 and Client 2 computers. So the size of the data packets captured is bigger than expected. Further filtering would be required to determine the actual size of the audio data. For example, using Ipsumdump to extract interested packets, refer to Appendix C.1.2 on page xxxiii.

4.5.2 Data Processing

To explain data processing process, Extraction-Transformation-Loading (ETL) concept is borrowed. Data processing is part of ETL process. ETL tools are pieces of software responsible for the extraction of data from several sources, their cleansing, customization and insertion into data warehouse [165]. However, the transformation part of the ETL process is more relevent to this project. There are two data sources that require processing before reports can be produced, the .wav and .pcap files. The degraded audios signal are fed into PESQ program to generate a report which consists of the original audio source, the degraded audio, the PESQMOS value, the crude_delay and the sample frequency of each comparison. A sample run script is available in Appendix C.1 on Subsection C.1.1 on page xxxiii. The .txt report is then transformed into .xls. Several additional fields are added to the existing records[1] before the file is imported into the R software. R is used to plot the required graphs. For this project,

[1]Refer to Appendix D.1.1 on page xli

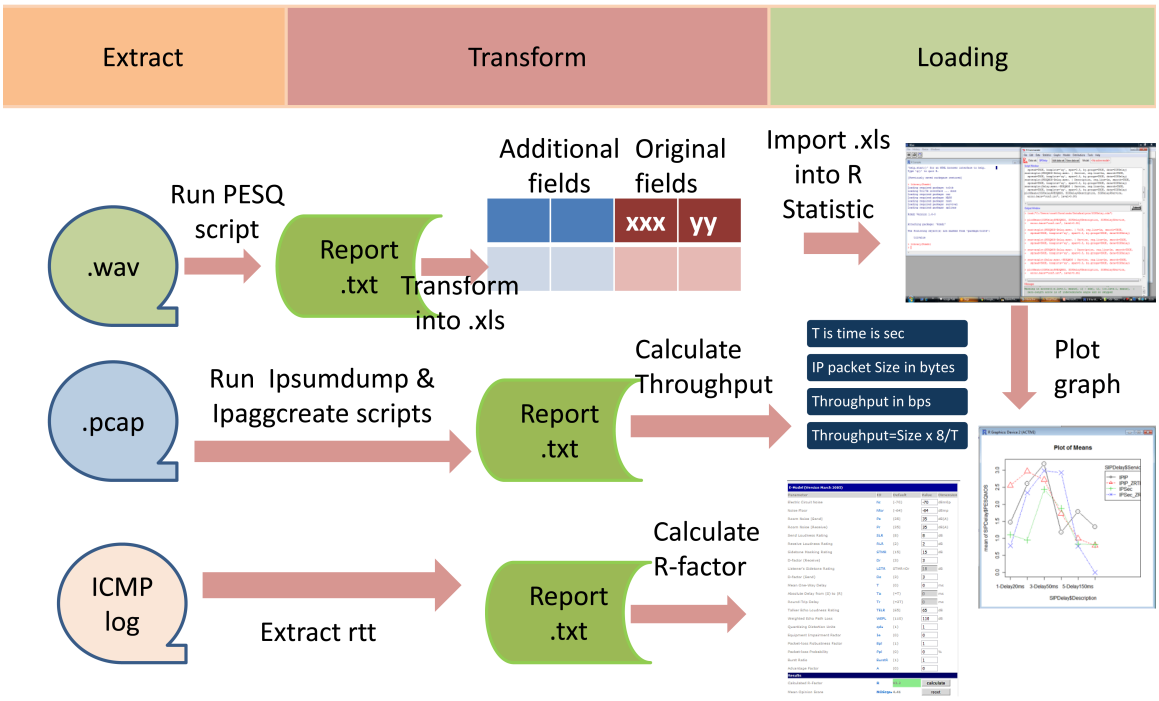


Figure 4.6: Data processing process

two types of graphs are plotted. They are the plot of means and the box and whisker graph. Figure 4.6 on page 102 depicted this process. Ipsumdump extracts raw data from a .pcap file and produces a .txt report containing epoch time and IP packet size in bytes. The data is updated into rrd database. Rrdtool is used to generate time versus throughput graph, refer to the graphs on page 135. Optionally, the total IP packet can be calculated from the .txt report produced by Ipsumdump. Alternatively, Ipaggcreate aggregates the total IP packet. Refer to Appendix C.1.2 on page xxxiii. $Throughput$ is an average rate of successful message delivery over a communication channel. It is measured in data packets per second or bits per second. $Throughput$ is defined as in equation (1):

$$Throughput = \frac{Size}{T} \text{ data packets per second} \tag{1}$$

where T is data transfer time in seconds and $Size$ is data payload size in bytes. The payload size includes Layer 2 and Layer 3 headers and it depends on the type of the codec in used. Table 2.2 on page 63 shows several payload size for different codec types.
In this experiment T is fixed to 10 seconds. To change $Throughput$ to bps, multiply the value by 8.

Last but not least is to extract RTT values from ICMP log report. RTT is the time taken for a signal to be sent plus the length of time it takes for an acknowledgment of that signal to be

received. It is also known as ping time. By default, ping waits 4 seconds for each response to be returned before displaying the 'Request Timed Out' message. The ICMP log report consists of a ping statistic that consists of RTT average, minimum and maximum times. In addition to that, it consists of the number of packet transmitted and received and the percentange of packet loss. Here, Client 1 sent ICMP ping to Client 2 for 10 counts of 100 bytes ping packet for every test parameters set on the Dummynet and the ICMP log shows RTT value between 0ms and 1600ms. The ICMP log is available in Appendix D.1.2 on page xliii.

4.6 Sixth Stage:Validation And Verification

The validation and verification process emphasizes on three things: Validation on the tools; Validation on the data collection process; and Verification on the result.

4.6.1 Tools

There are three different measurement tools used in this project. The PESQ calculator is used to measure the PESQMOS and crude_delay of audio streams. The PESQ measuring tool is described in Subsection 2.5.5 on page 69. PESQMOS is a value between 0 and 5. In which case higher value implies better score. Crude_delay is the delay due to the crude alignment between the degraded audio and the original audio. Crude_delay of zero implies that the two audio streams are 100% in synchronization.

In this project the PESQ software is from ITU-T Recommendation P.862 with limited user permission [73]. The ITU-T code can be used free of charge as long as it is for purely academic work and the project must not be the result of any contracted work funded by one industry or a consortium of several industries. The tool is used for understanding PESQ algorithm or to evaluate the ability of the PESQ algorithm to perform its intended function of predicting the speech quality of a system. The software is owned by British Telecommunications plc (BT) and Royal KPN NV (A Dutch landline and mobile telecommunications company). However, all rights are assigned to Psytechnics Limited and OPTICOM GmbH. To check the accuracy and consistency of the software, two untreated audio streams were tested with sample frequencies of 8000Hz and 16000Hz. The result is as follows:

REFERENCE	DEGRADED	PESQMOS	PESQMOS	SUBJMOS	COND	SAMPLE_FREQ	CRUDE_DELAY
sample1234.wav	sample1234.wav	SQValue=4.500	4.500	0.000	0	16000	0.0000
sample1234.wav	sample1234.wav	SQValue=4.500	4.500	0.000	0	8000	0.0000
sampleabcd.wav	sampleabcd.wav	SQValue=4.500	4.500	0.000	0	16000	0.0000
sampleabcd.wav	sampleabcd.wav	SQValue=4.500	4.500	0.000	0	8000	0.0000

It shows that the PESQMOS are 4.500 and CRUDE_DELAY are 0.0000 for both frequencies. So why is the PESQMOS is 4.5 and not 5.0? What happen to the extra 0.5 score? Does

not PESQ return score that conforms to ITU-T P.862? According to the 2004 Technical report by Malden Electronic Ltd, the score lies between the scale of -0.5 and 4.5 [102]. This score correlates with the quality subjective score. However, PESQ score tends to be optimistic on a low quality speech and pessimistic on a high quality speech. Several mappings are available to produce more realistic correlation between PESQMOS and quality subjective score. Here is the mapping for wideband measurement or PESQ-WB [102]:

$$PESQ\text{-}WB = 0.999 + \frac{4.999-0.999}{1+\exp -1.3669*PESQMOS+3.8224}$$

The second tool, E-Model calculator predicts the readiness of a network to handle VoIP services. The E-model takes into account a wide range of telephony-band impairments, in particular the impairment due to low bit-rate coding devices and one-way delay, as well as the 'classical' telephony impairments of loss, noise and echo. It can be applied to assess the voice quality of wireline and wireless scenarios, based on circuit-switched and packet-switched technologies. The sample report in Appendix D.1.3 on page lviii shows the results of running emodel.java program. The emodel.java program executes several test cases in batch mode. The NIST E-model calculator processes each test case one at a time [78]. The emodel.java code is available in Appendix C.1.4 on page xxxv. For validation, several one-way delays in ms and packet loss probabilities are feeded into the emodel.java program for G.711 codec type. The results are verified against NIST E-model calculator. The results are impartial for both tools.

The third tool is the CVSS calculator. To find the CVSS score, each service security condition is mapped into CVSS Base Metrics. CVSS is a vulnerability scoring system that was designed to provide an open and standardized method for rating IT vulnerabilities. It helps IT managers, vulnerability bulletin providers, security vendors, application vendors and researchers prioritize and coordinate a joint response to security vulnerabilities by communicating the properties of a vulnerability. In this project we use CVSS version 2 [49]. Table 4.6 explains the meaning of each parameter used in the assessment. The score ranging from 0 to 10. The higher the vulnerability, the higher the score. We customised an openCVSS.py class so that we can execute the test cases in batch mode. The openCVSS.py class was written by Dixon [42]. The NIST CVSS calculator executes a test case at any one time. There are about 64 samples output produced by openCVSS.py. They are presented in Figure 4.7 on page 106. Eight of them are recalculated using NIST CVSS calculator and presented in Table 4.7 on page 107. There are two inconsistencies found. One is related to the impact score and another one is regarded the base score. The different is about 0.1 for each. These differences might be due to the rounding to one decimal place.

Table 4.6: Base Metrics

Base Metrics [49]	
Access Vector (AV) • Local (L) • Adjacent (A) • Network (N)	This metric reflects how the vulnerability is exploited. The more remote an attacker can be to attack a host, the greater the vulnerability score.
Access Complexity (AC) • High (H) • Medium (M) • Low (L)	This metric measures the complexity of the attack required to exploit the vulnerability once an attacker has gained access to the target system. The lower the required complexity, the higher the vulnerability score.
Authentication (Au) • Multiple (M) • Single (S) • None (N)	This metric measures the number of times an attacker must authenticate to a target in order to exploit a vulnerability. This metric does not gauge the strength or complexity of the authentication process, only that an attacker is required to provide credentials before an exploit may occur. The fewer authentication instances that are required, the higher the vulnerability score.
Confidentiality Impact (C) • None (N) • Partial (P) • Complete (C)	This metric measures the impact on confidentiality of a successfully exploited vulnerability. Confidentiality refers to limiting information access and disclosure to only authorized users, as well as preventing access by, or disclosure to, unauthorized ones. Increased confidentiality impact increases the vulnerability score.
Integrity Impact (I) • None (N) • Partial (P) • Complete (C)	This metric measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and guaranteed veracity of information. Increased integrity impact increases the vulnerability score.
Availability Impact (A) • None (N) • Partial (P) • Complete (C)	This metric measures the impact to availability of a successfully exploited vulnerability. Availability refers to the accessibility of information resources. Attacks that consume network bandwidth, processor cycles, or disk space all impact the availability of a system. Increased availability impact increases the vulnerability score.

Sample Case	Access Vector	Access Complexity	Authentication	Exploita-bility Score	Confident-iality Impact	Integrity Impact	Availabil-ity Impact	Impact Score	Base Score
AV:A/AC:M/Au:S/C:P/I:C/A:N	Adjacent Network	Medium	Single Instance	4.4	Partial	Complete	None	7.8	5.8
AV:A/AC:H/Au:N/C:P/I:C/A:P	Adjacent Network	High	None	3.2	Partial	Complete	Partial	8.6	5.8
AV:L/AC:M/Au:M/C:C/I:C/A:N	Local	Medium	Multiple Instance	2.2	Complete	Complete	None	9.2	5.8
AV:N/AC:L/Au:M/C:P/I:P/A:P	Network	Low	Multiple Instance	6.4	Partial	Partial	Partial	6.4	5.8
AV:N/AC:M/Au:N/C:P/I:P/A:N	Network	Medium	None	8.6	Partial	Partial	None	4.9	5.8
AV:N/AC:M/Au:N/C:N/I:P/A:P	Network	Medium	None	8.6	None	Partial	Partial	4.9	5.8
AV:N/AC:H/Au:M/C:C/I:P/A:P	Network	High	Multiple Instance	3.2	Complete	Partial	Partial	8.6	5.8
AV:N/AC:H/Au:M/C:P/I:P/A:C	Network	High	Multiple Instance	3.2	Partial	Partial	Complete	8.6	5.8
AV:A/AC:L/Au:N/C:P/I:P/A:P	Adjacent Network	Low	None	6.5	Partial	Partial	Partial	6.4	5.8
AV:A/AC:M/Au:S/C:C/I:P/A:N	Adjacent Network	Medium	Single Instance	4.4	Complete	Partial	None	7.8	5.8
AV:A/AC:H/Au:N/C:C/I:P/A:P	Adjacent Network	High	None	3.2	Complete	Partial	Partial	8.6	5.8
AV:A/AC:H/Au:N/C:P/I:P/A:C	Adjacent Network	High	None	3.2	Partial	Partial	Complete	8.6	5.8
AV:N/AC:M/Au:N/C:P/I:N/A:P	Network	Medium	None	8.6	Partial	None	Partial	4.9	5.8
AV:A/AC:M/Au:S/C:C/I:N/A:P	Adjacent Network	Medium	Single Instance	4.4	Complete	None	Partial	7.8	5.8
AV:A/AC:M/Au:S/C:P/I:N/A:C	Adjacent Network	Medium	Single Instance	4.4	Partial	None	Complete	7.8	5.8
AV:L/AC:M/Au:M/C:C/I:N/A:C	Local	Medium	Multiple Instance	2.2	Complete	None	Complete	9.2	5.8
AV:A/AC:M/Au:M/C:P/I:C/A:P	Adjacent Network	Medium	Multiple Instance	3.5	Partial	Complete	Partial	8.6	5.9
AV:A/AC:H/Au:S/C:C/I:C/A:N	Adjacent Network	High	Single Instance	2.5	Complete	Complete	None	9.2	5.9
AV:A/AC:H/Au:M/C:C/I:C/A:P	Adjacent Network	High	Multiple Instance	2	Complete	Complete	Partial	9.5	5.9
AV:A/AC:H/Au:M/C:P/I:C/A:C	Adjacent Network	High	Multiple Instance	2	Partial	Complete	Complete	9.5	5.9
AV:L/AC:L/Au:M/C:C/I:C/A:N	Local	Low	Multiple Instance	2.5	Complete	Complete	None	9.2	5.9
AV:L/AC:M/Au:N/C:P/I:C/A:P	Local	Medium	None	3.4	Partial	Complete	Partial	8.6	5.9
AV:L/AC:H/Au:N/C:C/I:C/A:P	Local	High	None	1.9	Complete	Complete	Partial	9.5	5.9
AV:L/AC:H/Au:N/C:P/I:C/A:C	Local	High	None	1.9	Partial	Complete	Complete	9.5	5.9
AV:L/AC:H/Au:M/C:C/I:C/A:C	Local	High	Multiple Instance	1.2	Complete	Complete	Complete	10	5.9
AV:A/AC:M/Au:M/C:C/I:P/A:P	Adjacent Network	Medium	Multiple Instance	3.5	Complete	Partial	Partial	8.6	5.9
AV:A/AC:M/Au:M/C:P/I:P/A:C	Adjacent Network	Medium	Multiple Instance	3.5	Partial	Partial	Complete	8.6	5.9
AV:A/AC:H/Au:M/C:C/I:P/A:C	Adjacent Network	High	Multiple Instance	2	Complete	Partial	Complete	9.5	5.9
AV:L/AC:M/Au:N/C:C/I:P/A:P	Local	Medium	None	3.4	Complete	Partial	Partial	8.6	5.9
AV:L/AC:M/Au:N/C:P/I:P/A:C	Local	Medium	None	3.4	Partial	Partial	Complete	8.6	5.9
AV:L/AC:H/Au:N/C:C/I:P/A:C	Local	High	None	1.9	Complete	Partial	Complete	9.5	5.9
AV:A/AC:H/Au:S/C:C/I:N/A:C	Adjacent Network	High	Single Instance	2.5	Complete	None	Complete	9.2	5.9
AV:L/AC:L/Au:M/C:C/I:N/A:C	Local	Low	Multiple Instance	2.5	Complete	None	Complete	9.2	5.9
AV:L/AC:M/Au:S/C:C/I:C/A:N	Local	Medium	Single Instance	2.7	Complete	Complete	None	9.2	6
AV:L/AC:M/Au:M/C:C/I:C/A:P	Local	Medium	Multiple Instance	2.2	Complete	Complete	Partial	9.5	6
AV:L/AC:M/Au:M/C:P/I:C/A:C	Local	Medium	Multiple Instance	2.2	Partial	Complete	Complete	9.5	6
AV:L/AC:H/Au:S/C:C/I:C/A:C	Local	High	Single Instance	1.5	Complete	Complete	Complete	10	6
AV:N/AC:M/Au:S/C:P/I:P/A:P	Network	Medium	Single Instance	6.8	Partial	Partial	Partial	6.4	6
AV:L/AC:M/Au:M/C:C/I:P/A:C	Local	Medium	Multiple Instance	2.2	Complete	Partial	Complete	9.5	6
AV:L/AC:M/Au:S/C:C/I:N/A:C	Local	Medium	Single Instance	2.7	Complete	None	Complete	9.2	6
AV:N/AC:H/Au:N/C:P/I:C/A:N	Network	High	None	4.9	Partial	Complete	None	7.8	6.1
AV:N/AC:H/Au:N/C:N/I:C/A:P	Network	High	None	4.9	None	Complete	Partial	7.8	6.1
AV:N/AC:H/Au:S/C:P/I:C/A:P	Network	High	Single Instance	3.9	Partial	Complete	Partial	8.6	6.1
AV:A/AC:L/Au:N/C:N/I:C/A:N	Adjacent Network	Low	None	6.5	None	Complete	None	6.9	6.1
AV:A/AC:H/Au:S/C:C/I:C/A:P	Adjacent Network	High	Single Instance	2.5	Complete	Complete	Partial	9.5	6.1
AV:A/AC:H/Au:S/C:P/I:C/A:C	Adjacent Network	High	Single Instance	2.5	Partial	Complete	Complete	9.5	6.1
AV:L/AC:L/Au:N/C:P/I:C/A:P	Local	Low	None	3.9	Partial	Complete	Partial	8.6	6.1
AV:L/AC:L/Au:M/C:C/I:C/A:P	Local	Low	Multiple Instance	2.5	Complete	Complete	Partial	9.5	6.1
AV:L/AC:L/Au:M/C:P/I:C/A:C	Local	Low	Multiple Instance	2.5	Partial	Complete	Complete	9.5	6.1
AV:N/AC:H/Au:N/C:C/I:P/A:N	Network	High	None	4.9	Complete	Partial	None	7.8	6.1
AV:N/AC:H/Au:N/C:N/I:P/A:C	Network	High	None	4.9	None	Partial	Complete	7.8	6.1
AV:N/AC:H/Au:S/C:C/I:P/A:P	Network	High	Single Instance	3.9	Complete	Partial	Partial	8.6	6.1
AV:N/AC:H/Au:S/C:P/I:P/A:C	Network	High	Single Instance	3.9	Partial	Partial	Complete	8.6	6.1
AV:A/AC:H/Au:S/C:C/I:P/A:C	Adjacent Network	High	Single Instance	2.5	Complete	Partial	Complete	9.5	6.1
AV:L/AC:L/Au:N/C:C/I:P/A:P	Local	Low	None	3.9	Complete	Partial	Partial	8.6	6.1
AV:L/AC:L/Au:N/C:P/I:P/A:C	Local	Low	None	3.9	Partial	Partial	Complete	8.6	6.1

Figure 4.7: CVSS Base Score report

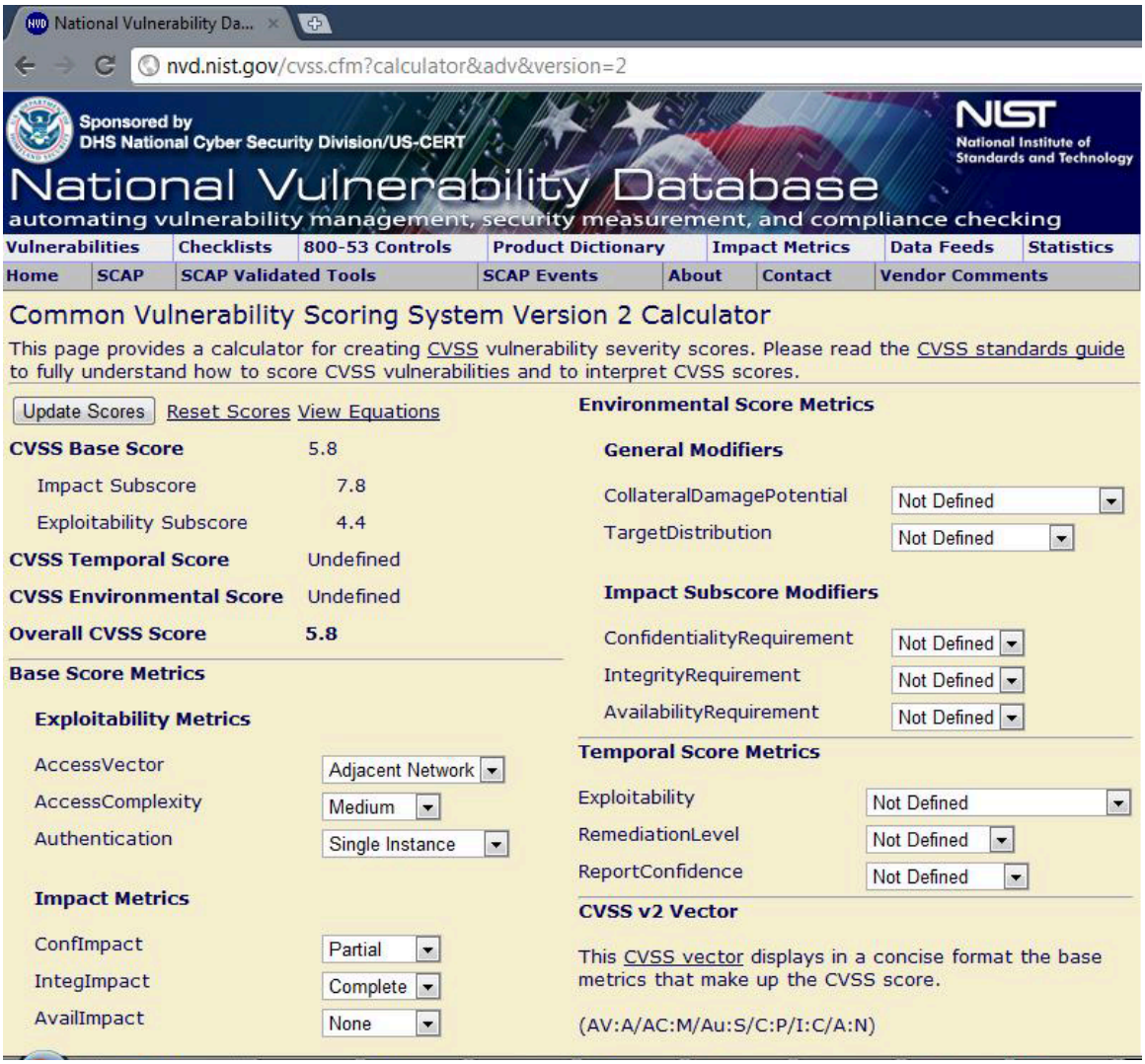


Figure 4.8: NIST CVSS calculator version 2

Table 4.7: Several output from NIST CVSS calculator version 2

Sample	Exploitability Subscore	Impact Subscore	CVSS Base Score
(AV:A/AC:M/Au:S/C:P/I:C/A:N)	4.4	7.8	5.8
(AV:A/AC:H/Au:N/C:C/I:P/A:P)	3.2	8.5	5.8
(AV:L/AC:L/Au:M/C:C/I:C/A:N)	2.5	9.2	5.9
(AV:L/AC:H/Au:M/C:C/I:C/A:C)	1.2	10	5.9
(AV:N/AC:M/Au:S/C:P/I:P/A:P)	6.8	6.4	6
(AV:A/AC:L/Au:N/C:N/I:C/A:N)	6.5	6.9	6.1
(AV:L/AC:L/Au:M/C:P/I:C/A:C)	2.5	9.5	6.2
(AV:L/AC:L/Au:N/C:P/I:P/A:C)	3.9	8.5	6.1

4.6.2 Data Collection

As mentioned earlier, data collection is one of the most important step in this project because the data collected is analysed to evaluate the perceived voice quality of Skype, Google Talk and Express Talk applications. For validation, the scripts as in Appendix B.1 on page xxvi are executed repeatedly for several times. As much as possible the scripts are run with less human intervention. For every parameters set on Dummynet, ten readings are taken, (refer to Section 4.4 on page 97). The consistency of the results are checked. Sample reports are available in Appendix D.1 on page xli. We also plot box and whisker graphs to check for consistency of the data collected. For example, Figure 4.9 on page 108 shows the data taken for Google Talk over IPIP tunnel on $delay$ between 0 ms to 400 ms range. In this example, the scores are quite consistent. There are more related graphs in Appendix D.1.6 on page lxx.

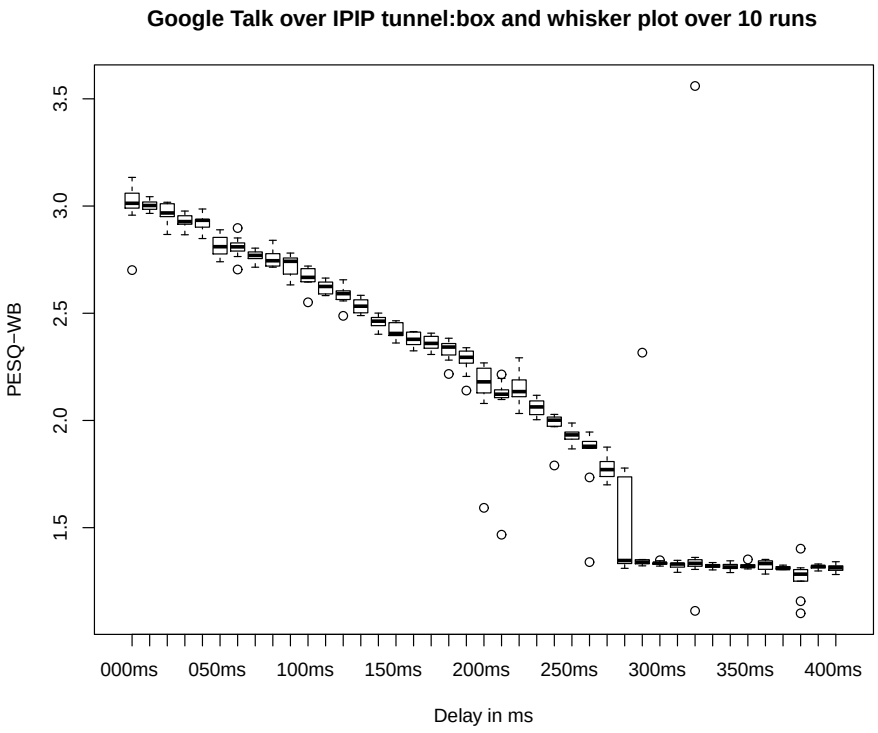


Figure 4.9: Graph of Delay versus PESQ-WB

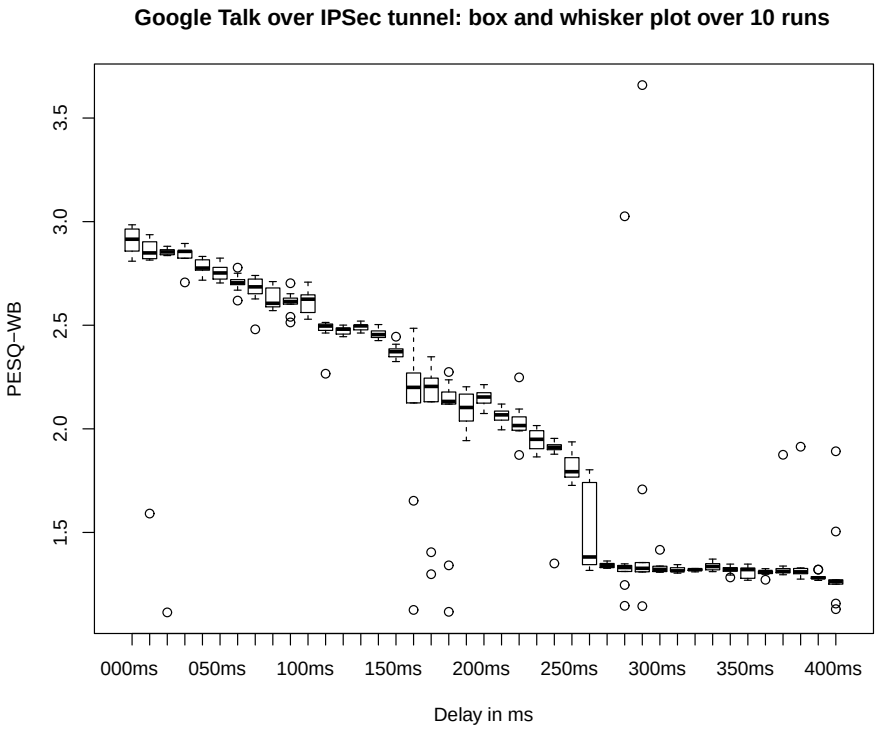


Figure 4.10: Graph of Delay versus PESQ-WB

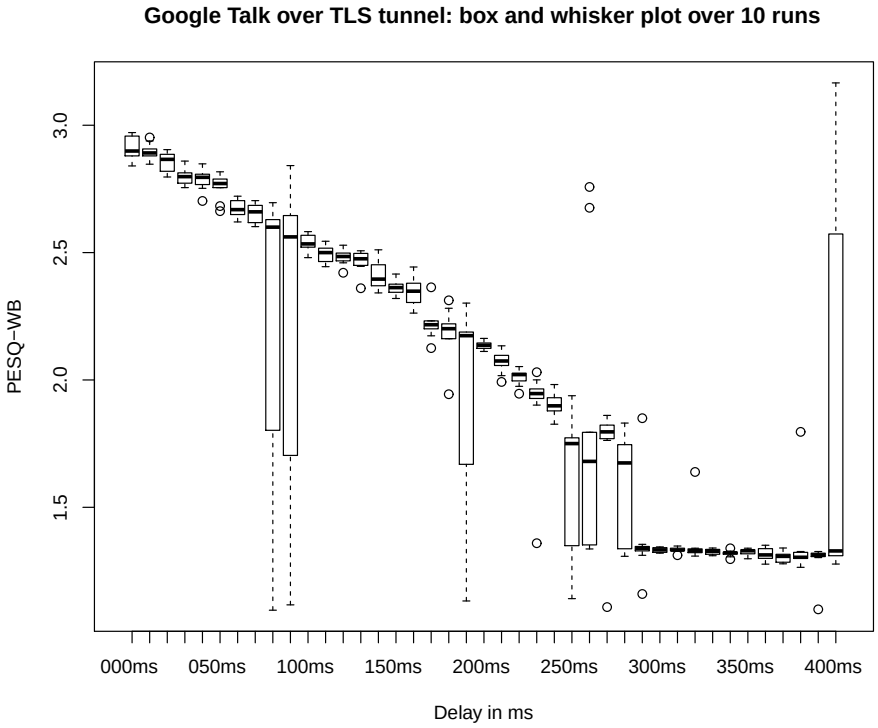


Figure 4.11: Graph of Delay versus PESQ-WB

4.7 Summary

This chapter explains the project implementation stages. There are six stages. Each stage is important as they aim to increase consistency and reduce error of the collected data. These data are used for analysing the selected VoIP performance. Next, Chapter 5 describes the result and analysis of the experiment.

Chapter 5

Results and Analysis

"Indeed the cure for ignorance is to ask."

Al-Hadith Abu Dawud

This chapter discusses the results of the experiments conducted as described in Chapter 4, to evaluate the performance of Skype, Google Talk[1] and Express Talk. There are three significant scores. First is the PESQ-WB score, for each VoIP service. Second is the CVSS base score and third, is the throughput for each test condition. The test conditions were described in Subsection 3.3.1 on page 83. PESQ-WB score is the wideband measurement based on PESQMOS score. They are described in Subsection 4.6.1 on page 103. We plotted the graphs of delay versus PESQ-WB, packet loss rate versus PESQ-WB and box and whisker graph on jitter. We analysed the effects of delay, packet loss rate and jitter on the voice qualities. In addition, CVSS scores predict the security vulnerability of the VoIP services in the testbed. We also analysed the throughput gained by each experiment. According to Chiang et al., throughput can reflect the bandwidth requirement and the achieved voice quality of a VoIP application whereby higher throughput often results in better voice quality [28]. The process of extracting throughput values and producing time series graph using a rrdgraph function is in Subsection 4.5.2 on page 101. The rrdgraph is a graph function of rrdtool that is used to present the rrd data into a nice graphical representation or numerical report [122].

It is important to check the readiness of our testbed to support VoIP services. This is because we would like to ensure that any results obtained are not influenced by other factors other than the parameters set on each test scenario. This is our way of controlling the experiment. Hence, R factor is used to gauge the readiness of the testbed to run the VoIP services. E-model calculator maps R factor values to MOS scores. R-factor values of below 50 are generally unacceptable and typical telephone connections do not get above 94, giving a typical range

[1] In this project sometimes it is referred to as Gtalk

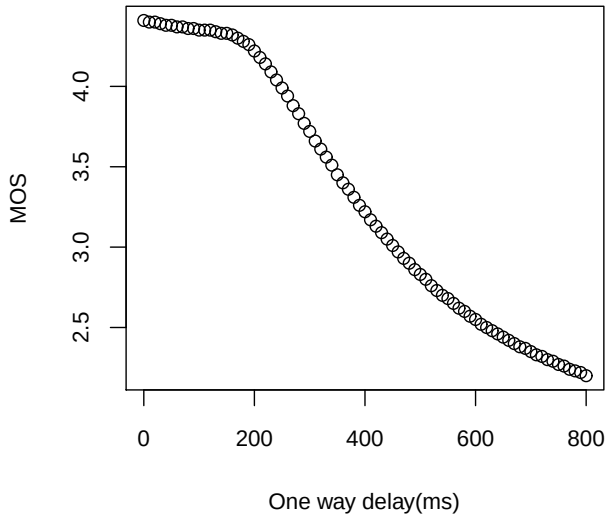


Figure 5.1: One way delay vs MOS

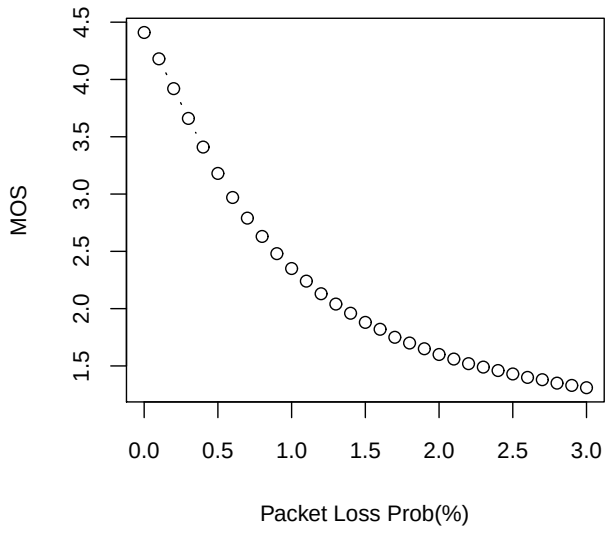


Figure 5.2: Packet Loss Probability vs MOS

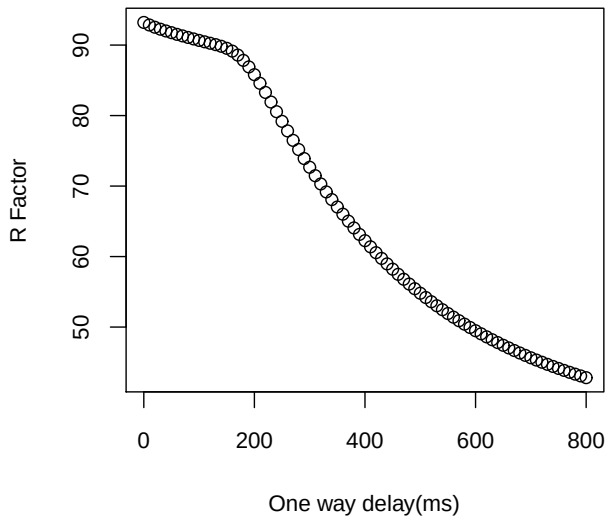


Figure 5.3: One way delay vs R

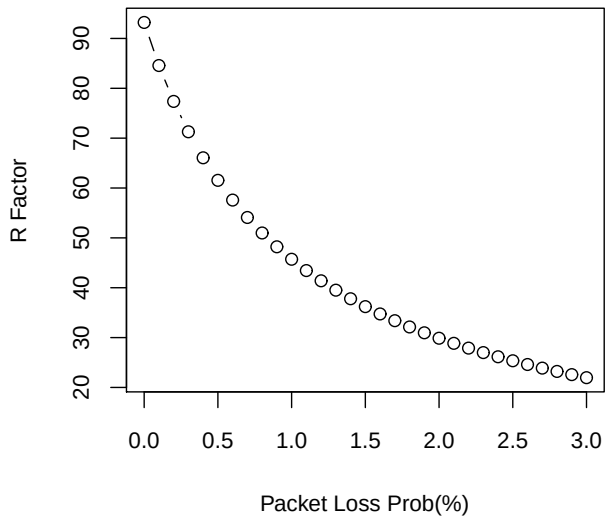


Figure 5.4: Packet Loss Probability vs R

of acceptable scores of 50 to 94 [80]. We have discussed about E-model in Subsection 2.5.6 on page 70. The E-model calculator program is available in Appendix C.1.4 on page xxxv. Figure 5.1 on page 112 shows the MOS scores based on E-model prediction for the testbed network with the one-way delay between 0ms to 800ms and Figure 5.2 on page 112 shows the MOS scores for the packet loss probability between 0% to 3%. These scores influence the range of the $delay$ and plr parameters that were set for this project. In addition, it is recommended that an upper limit of 400ms for a one-way delay of general network planning. This is because practically many voice calls and interactive data applications might be affected by much lower delays [74]. Figure 5.1 on page 112 shows that the MOS value is approximately 3.25 at 400ms delay and approximately 4.3 at 150ms delay. Figure 5.3 on page 112 shows that the quality of the audio becomes poor beginning at 600ms mark (i.e. where R value is 50 and MOS is approximately 2.5). In addition, Figure 5.2 on page 112 shows that the MOS values decrease exponentially to packet loss probability. Figure 5.4 on page 112 indicates that the voice quality is poor for plr beyond 0.8%. Theoretically the testbed is suitable to carry voice data for any delays less than 150ms and any packet loss probability that is less than 0.1%. So, it is important to inspect the PESQ-WB scores of the VoIP clients beyond these limits.

5.1 Voice Quality:PESQ-WB Scores

5.1.1 Delay

Normal speech consists of talkspurts that last for a few hundred milliseconds and silence periods, which occur within spoken words and between words. In a packet-based network, voice packets are generated periodically at the sender and transmitted across the network [110]. In most cases, users would not notice any delays in a conversation if they are less that 150ms [74, 43]. The experiment shows that voice quality drops due to a delay for Skype and Google Talk. This is based on the graphs that are depicted on page 114 and page 115. However, the quality of Express Talk is better than Skype or Google Talk, refer to Figure 5.8 on page 116. There are no significant differences in Express Talk performances in the secure tunnels (i.e. either IPSec or TLS) to those executed in the IPIP tunnel. From our observation, Skype and Express Talk did not show very much variation with delay but Google Talk degrades with higher delay as shown in Figure 5.8. We have no clear answer to this. It is probably due to an unknown factor at work in the codecs. We have no way to determine the reason due to the closed nature of the Google Talk and the Skype systems.

Next, we analysed Skype performances. The experiment shows that overall, Skype has much better voice quality than Google Talk, refer to Figure 5.8 on page 116, Figure 5.9 on page 116 and Figure 5.10 on page 117. However, Skype performances are quite random for

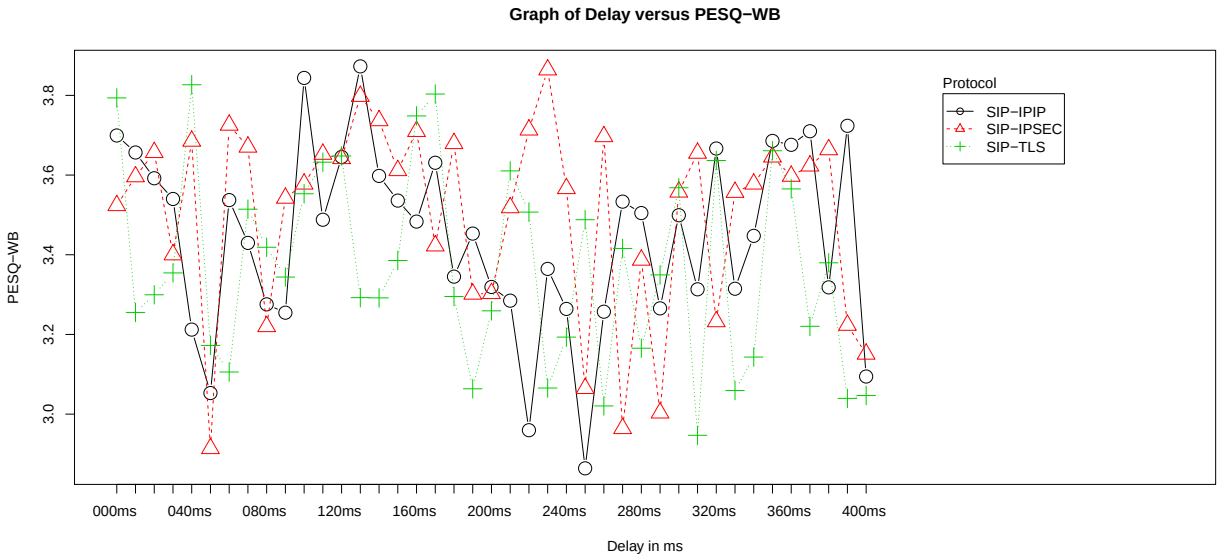


Figure 5.5: SIP protocol: The result of taking the mean of 10 runs

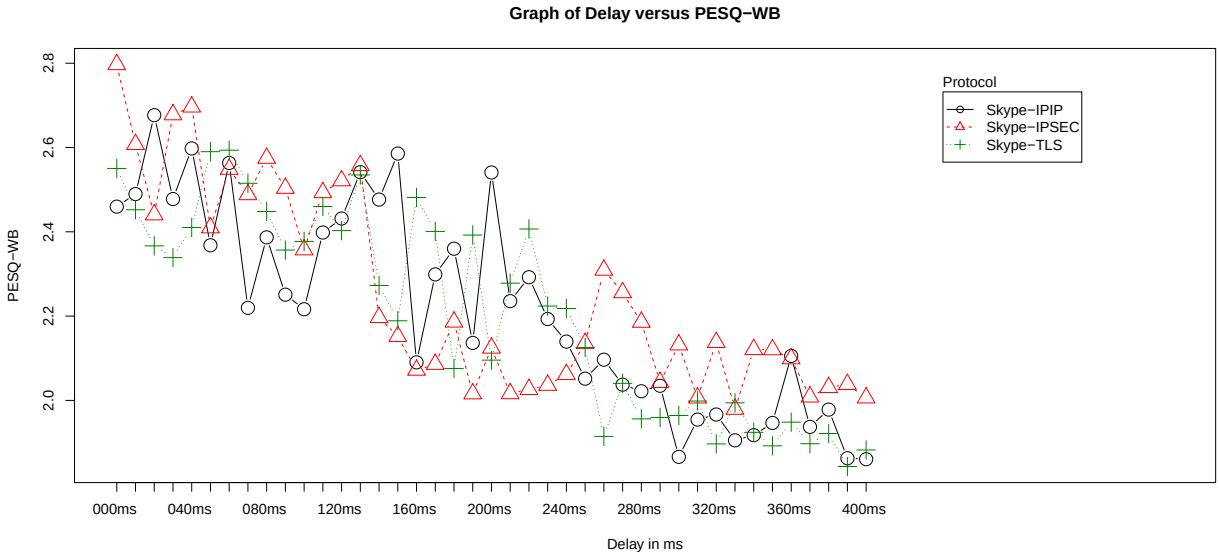


Figure 5.6: Skype protocol:The result of taking the mean of 10 runs

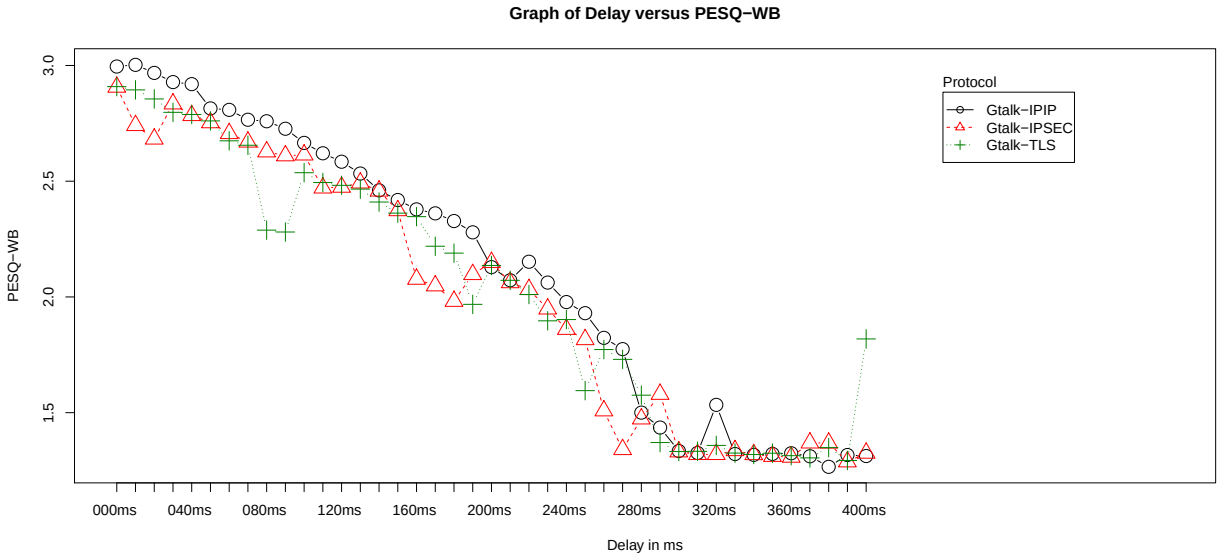


Figure 5.7: Google Talk protocol:The result of taking the mean of 10 runs

all three network conditions. Interestingly, the scores for Skype are more stable in the IPSec tunnel than to the TLS tunnel. The graph in Figure 5.6 on page 114 shows that, despite the random behaviour, there are substantial decrease in Skype performances due to delay. The majority of the scores are below 2.5 which indicate poorer quality than the acceptable telephony quality. The experiment also shows that the performance of Google Talk is slightly better in the IPSec tunnel than to the TLS, refer to Figure 5.7 on page 115. Google Talk performances drop in both secure tunnels (i.e. IPSec and TLS). Google Talk performances drop linearly with $delay$ when the delays are below 300ms mark. The voice quality scores have worsen after the 300ms delay.

In summary, the observation so far indicates that the voice qualities for Skype shows slight improvement in an IPSec tunnel. It also shows that Google Talk performance decreases due to additional processing to authenticate security tunnels at Network layer (i.e. IPSec) and Transport layer (i.e. TLS). On the other hand, Express Talk performance are almost similar in any of the tunnels. It also shows that, in general, network delays can cause substantial defect on the voice qualities for some of the VoIP protocols. Eventhough the graph in Figure 5.1 on page 112 has suggested that the starting point for the MOS for all VoIP clients should be around 4.3, however, each client has lower PESQ-WB than expected. At this point, we could not conclude what causes this defect. We suspect that the actual delay in the testbed is more than the theoretical delay value. In order to prove this, we execute ICMP ping command from Client A to Client B. The result is shown in Appendix D.1.2 on page xliii. However, there were no unusual activities reported in the log. Another potential cause is due to a different bandwidth requirement imposed by each VoIP client to carry the voice data. In which case,

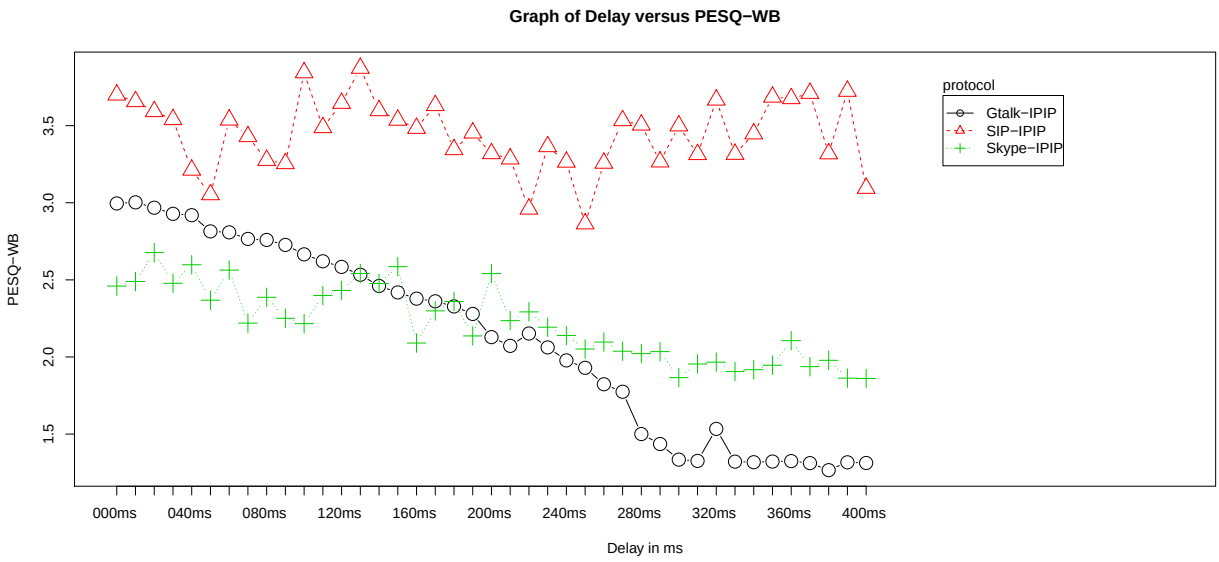


Figure 5.8: IPIP tunnel: The result of taking the mean of 10 runs

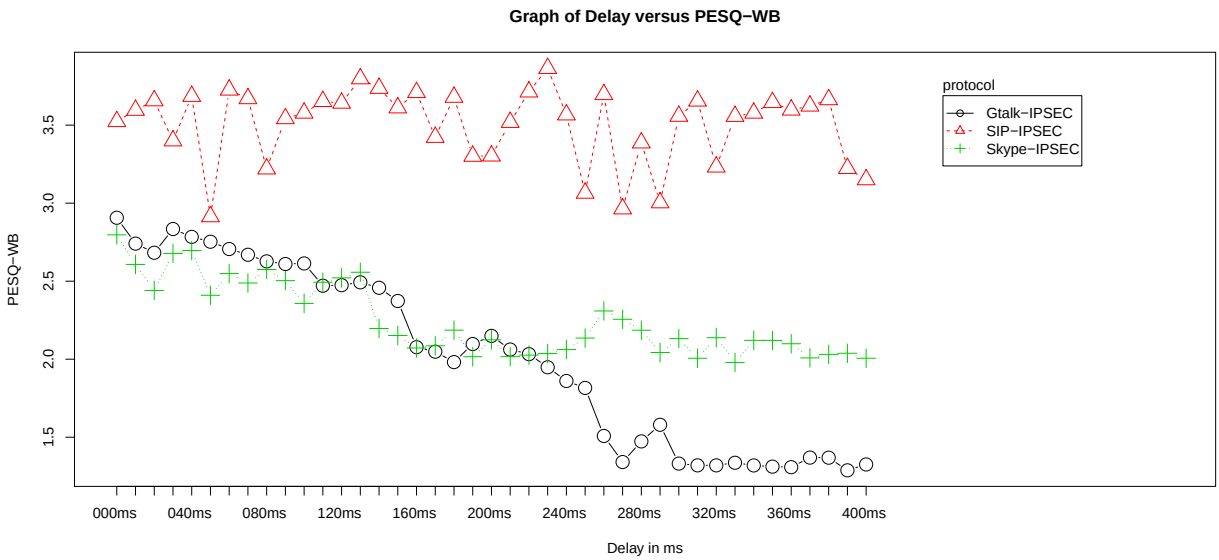


Figure 5.9: IPSec tunnel: The result of taking the mean of 10 runs

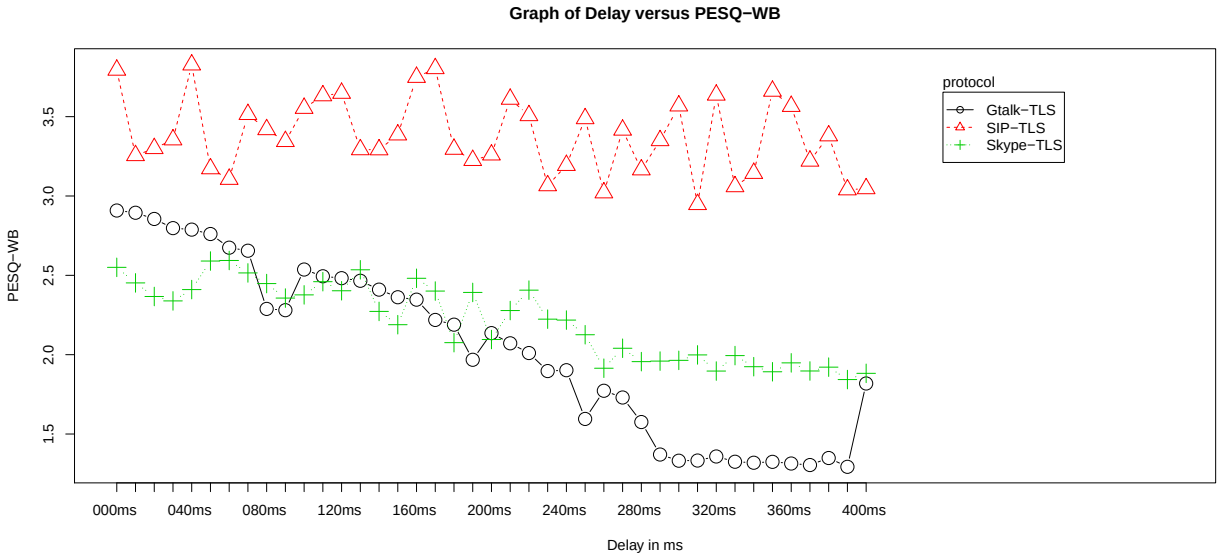


Figure 5.10: TLS tunnel: The result of taking the mean of 10 runs

we analyse the actual throughput of each client for every test cases. Section 5.3 on page 131 discusses these throughputs and their effects on voice qualities.

5.1.2 Packet Loss Rate

In a TCP/IP network, packet loss can be caused by numerous factors such as insufficient bandwidth, network connection problems, faulty networking hardware, faulty network drivers or normal routing routines. However, what matter most is how many packets are lost at the final destination. If the final destination is not affected, then all other delay and packet loss are just an artifact of router configuration or something similar. Hence, it should not cause any problem because it does not impact network availability to the end users in general. However, the data packets might be transmitted over a longer duration.

This is not the case for time sensitive applications. For time sensitive applications such as VoIP, packet loss during transmission might cause conversational difficulty because in most cases the network transport protocol is UDP. UDP provides no recovery for lost packets. Applications that use UDP are expected to define their own mechanisms for handling packet loss. In general, small percentage loss would not effect the overall service quality. The only effect seen due to the occasional dropped packet is jitter. However, if the percentage of loss of the total packet stream is high then it affects the quality significantly. The experiment involving packet loss rate in this project would determine the plr limit of each VoIP service that would have significant impact on the voice quality. In addition, it would show whether IPSec and TLS have significant influences on these performances.

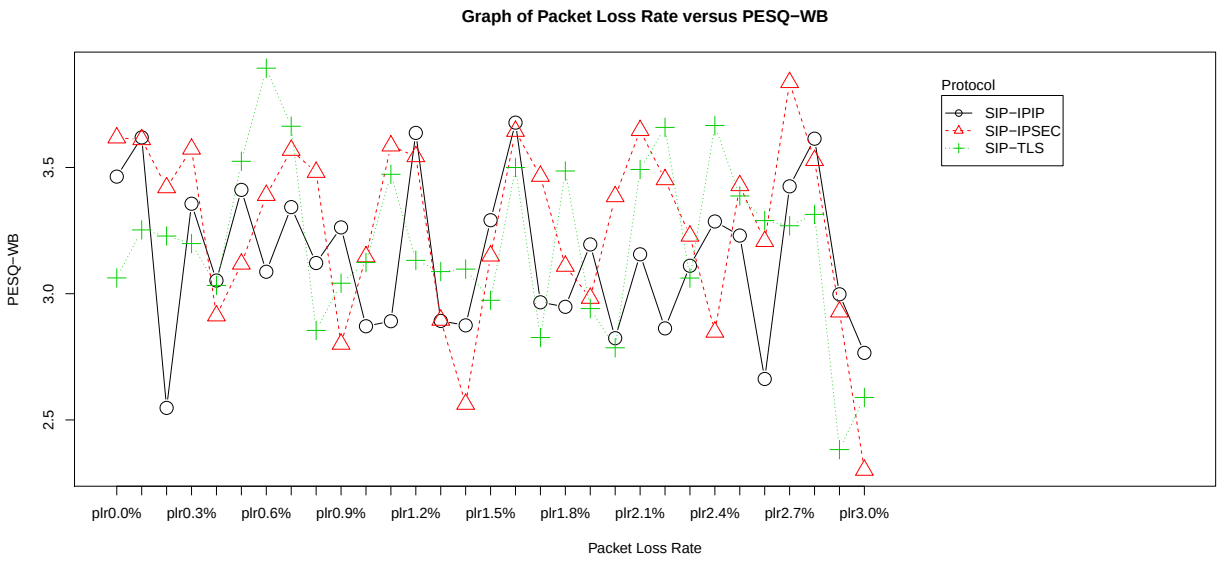


Figure 5.11: SIP Protocol: The result of taking the mean of 10 runs

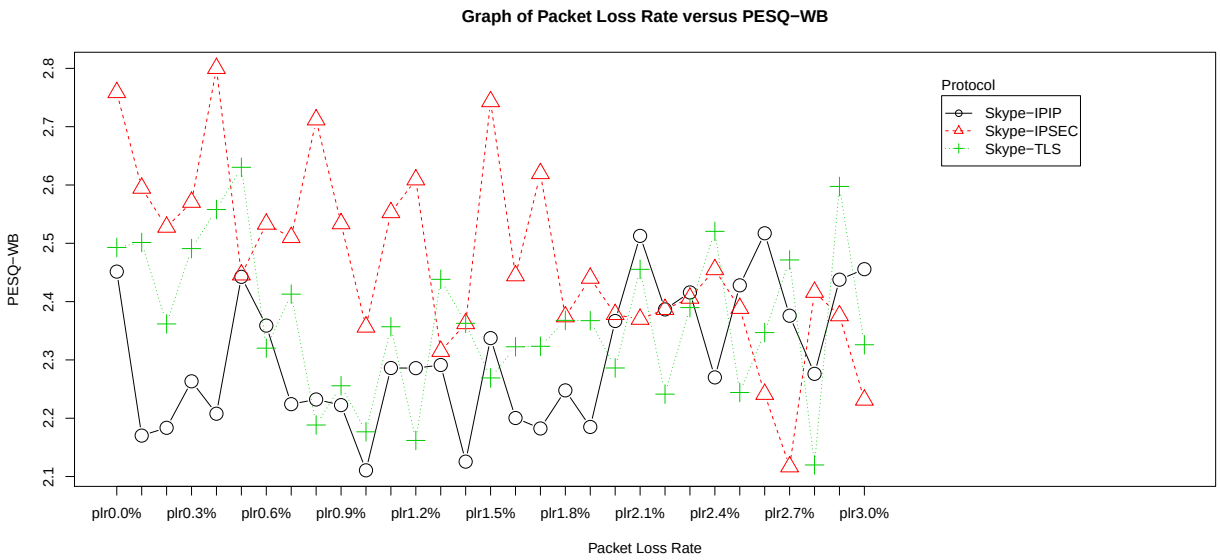


Figure 5.12: Skype Protocol: The result of taking the mean of 10 runs

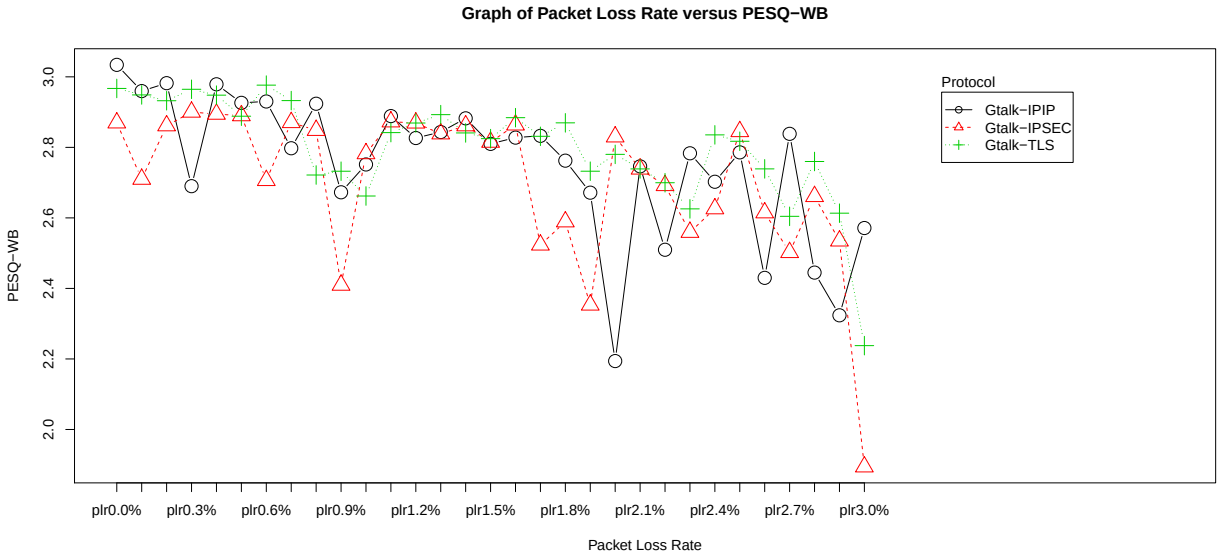


Figure 5.13: Google Talk Protocol: The result of taking the mean of 10 runs

Analysis on the graphs depict on page 118 and page 119 show that packet loss rate have significant effect on the PESQ-WB scores regardless of the percentage of loss. In all cases the scores are quite random although Google Talk in the TLS and Google Talk in the IPSec show more stable scores for plr between 0% to 1.5%. Eventhough the graph in Figure 5.2 on page 112 has suggested that the starting point for the MOS for all VoIP clients should be around 4.3 at 0% plr and drops exponentially due to packet loss probability, however, each client has much lower PESQ-WB scores than expected. From the experiment, refer to Figure 5.11 on page 118, it can be deduced that Express Talk has better scores than Skype or Google Talk regardless of the network conditions. There is no significant difference in term of Express Talk performance in an IPSec or a TLS or just an IPIP tunnel without security. Figure 5.12 on page 118 also shows that Skype performance is very sensitive to packet loss. The score can be very low even in a unsecured network. This is demonstrated by the low score in the IPIP tunnel when the plr is between 0% and 1%. Generally, Skype performances have no clear pattern. However, the scores are much better in an IPSec tunnel. Figure 5.13 on page 119 demonstrates that Google Talk is more tolerance to packet loss. Google Talk performance drops slightly due to the Network layer and Transport layer securities. However, the performance are better than Skype. Most of the PESQ-WB scores are above 2.5. Nevertheless, the PESQ-WB scores for both IPSec and TLS are quite stable.

In summary, the experiment shows that even a small percentage of a packet loss rate can significantly reduces a VoIP performance. Skype performances are sensitive to packet loss (i.e. plr between 0% to 3.0%). Google Talk and Express Talk on the other hand, can handle packet loss better than Skype.

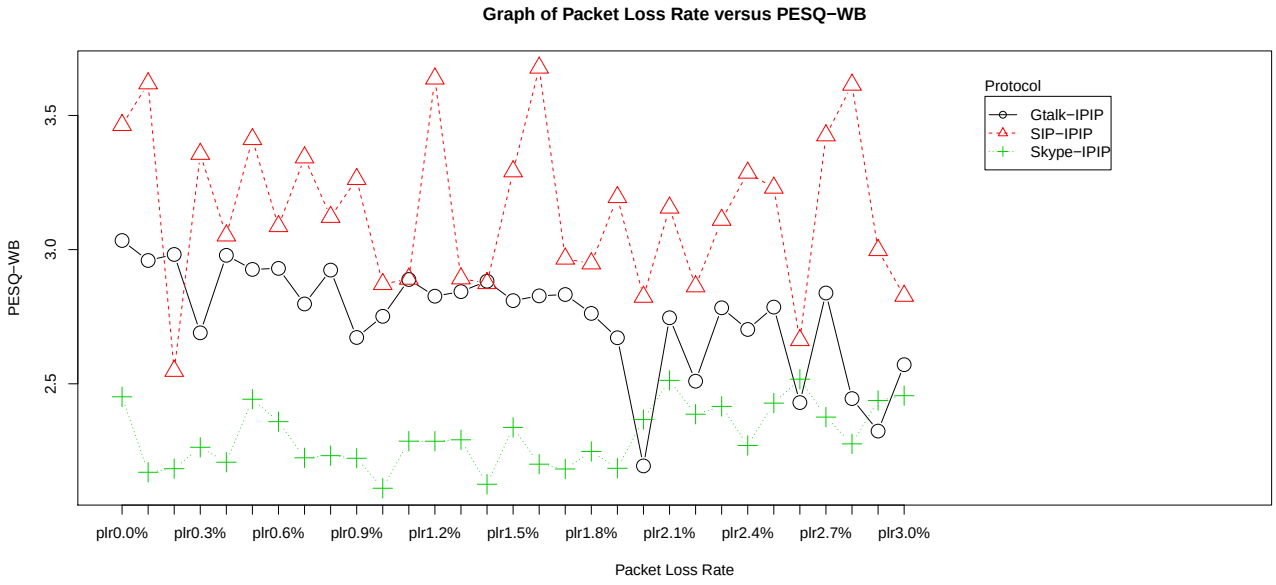


Figure 5.14: IPIP protocol:The result of taking the mean of 10 runs

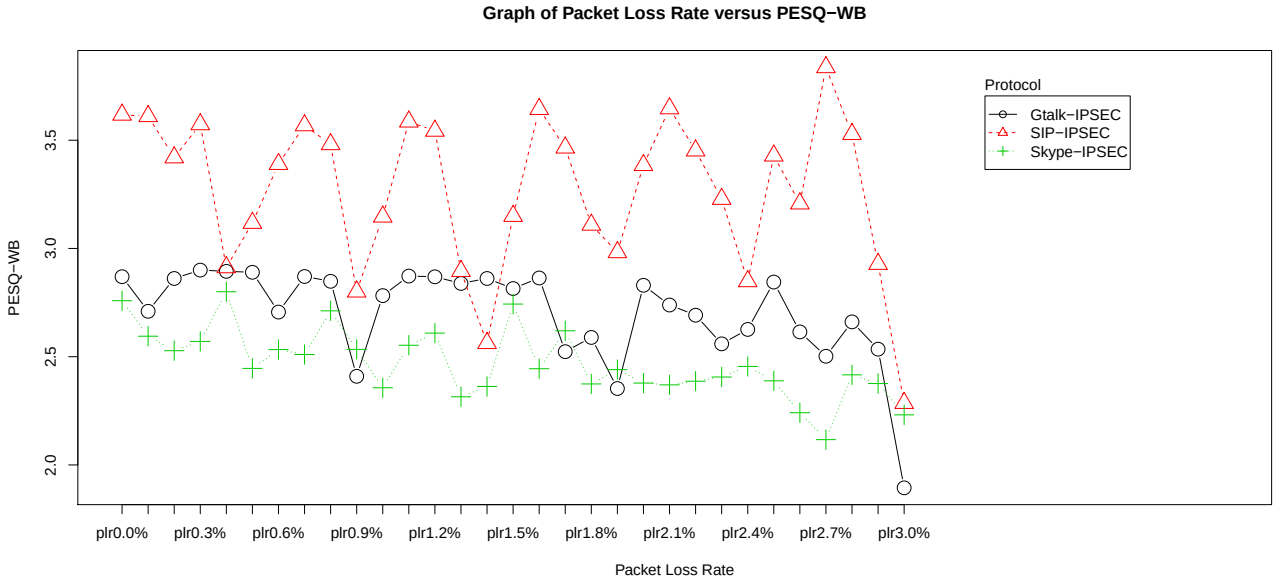


Figure 5.15: IPSec protocol:The result of taking the mean of 10 runs

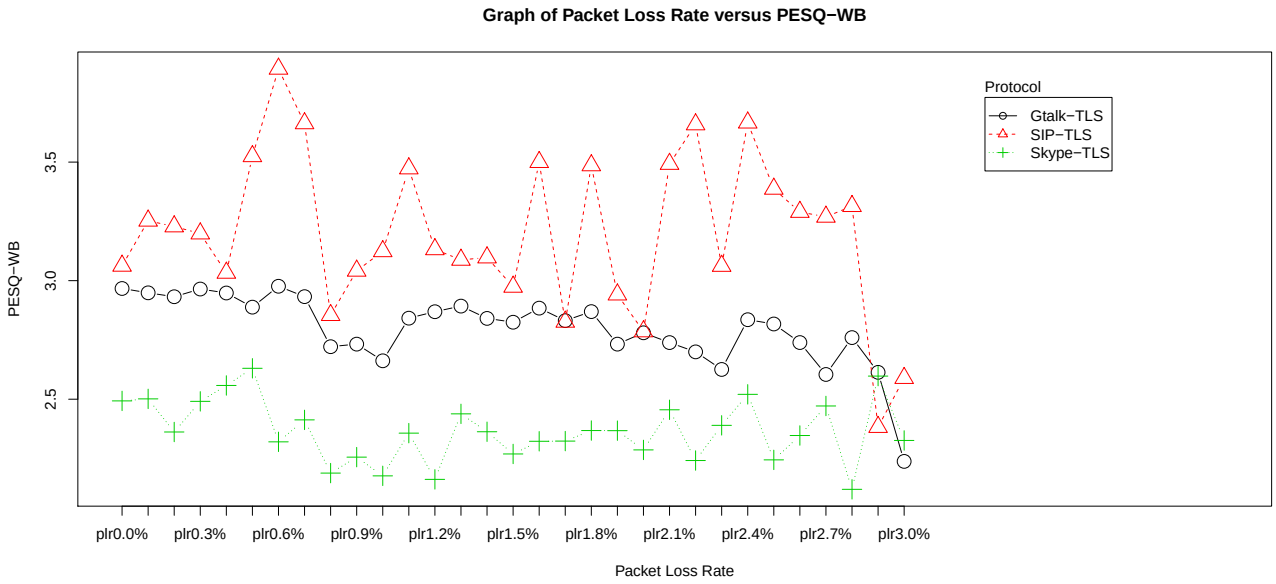


Figure 5.16: TLS protocol:The result of taking the mean of 10 runs

5.1.3 Jitter

Jitter is the variation of delay between packets arriving. Jitter is caused by network congestion, timing drift, route changes, queuing at the egress gateway or buffering delay. The amount of allowable jitter depends greatly on the application. In VoIP, jitter introduces distortions in audio signals which lead to low audio quality. Jitter can also lead to loss of transmitted data between network devices. In this project, jitters have been introduced by creating three pipelines with different probability of delays to see the effect of the independent variable jitter to each VoIP service. The three delays are below 400ms as recommended by the ITU-T G.114 Recommendation. However, the probability of each pipeline to transfer data varies, refer to Table 5.1 on page 123. Each profile creates different RTT values. There is no particular reason for choosing 100ms delay, 200ms delay and 300ms delay of pipelines. However, for many intra-regional routes in the range of 5000km or less, VoIP users are likely to experience mouth-to-ear delays that is less than 150ms [74]. In a worst case scenario a VoIP mouth-to-ear path is likely to see a delay of just 300ms. So these delays seem to mimic the typical long distance calls. The difference between a maximum RTT and a minimum RTT seems to influence the delay variation reading. The RTT magnitude and rate of occurance of each pipeline determine the damage done on the voice quality. This observation is demonstrated in the box and whisker graphs of Figure 5.17, Figure 5.18, Figure 5.19, Figure 5.20, Figure 5.21, Figure 5.22, Figure 5.23, Figure 5.24 and Figure 5.25. The performances of Skype, Google Talk and Express Talk were affected by these jitters as shown by the different scores for each condition on every different test runs.

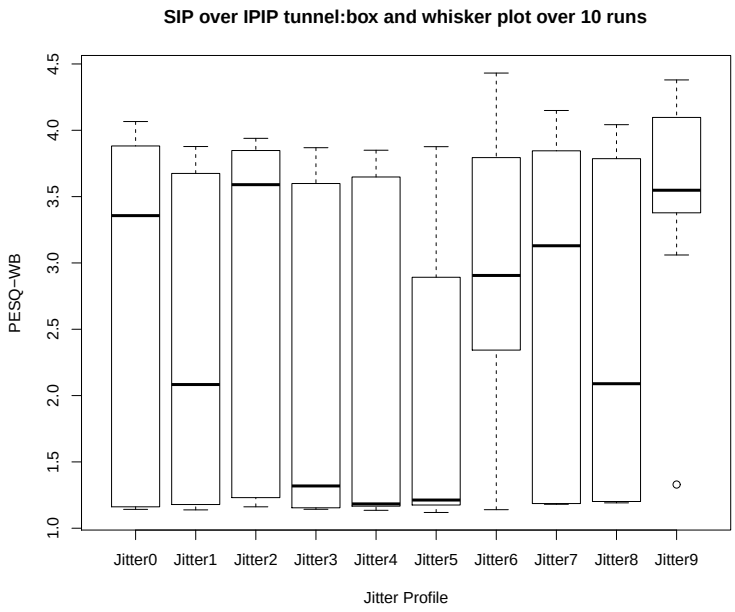


Figure 5.17: SIP IPIP Protocol: Box and whisker plot over 10 runs

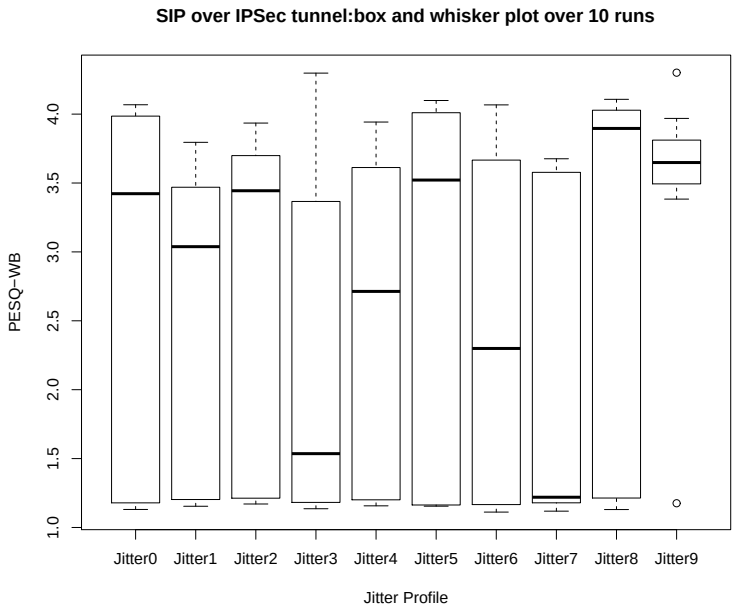


Figure 5.18: SIP IPSec Protocol: Box and whisker plot over 10 runs

Table 5.1: Jitter Profiles

Profile/Pipeline	**100ms delay** (Probability)	**200ms delay** (Probability)	**300ms delay** (Probability)	**Round Trip Time**
jitter0	0.33	0.33	0.33	Minimum = 400ms, Maximum = 897ms, Average = 649ms
jitter1	0.2	0.3	0.5	Minimum = 199ms, Maximum = 899ms, Average = 618ms
jitter2	0.5	0.3	0.2	Minimum = 0ms, Maximum = 496ms, Average = 335ms
jitter3	0.3	0.5	0.2	Minimum = 98ms, Maximum = 700ms, Average = 398ms
jitter4	0.2	0.5	0.3	Minimum = 396ms, Maximum = 897ms, Average = 587ms
jitter5	0.5	0.2	0.3	Minimum = 97ms, Maximum = 799ms, Average = 434ms
jitter6	0.3	0.2	0.5	Minimum = 396ms, Maximum = 1098ms, Average = 737ms
jitter7	0.05	0.05	0.9	Minimum = 898ms, Maximum = 1200ms, Average = 1038ms
jitter8	0.05	0.9	0.05	Minimum = 597ms, Maximum = 800ms, Average = 768ms
jitter9	0.9	0.05	0.05	Minimum = 297ms, Maximum = 398ms, Average = 357ms

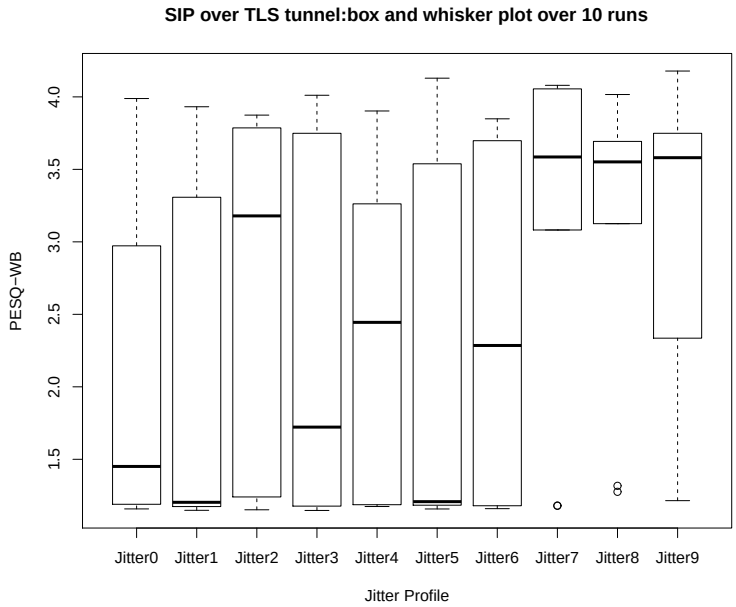


Figure 5.19: SIP TLS Protocol: Box and whisker plot over 10 runs

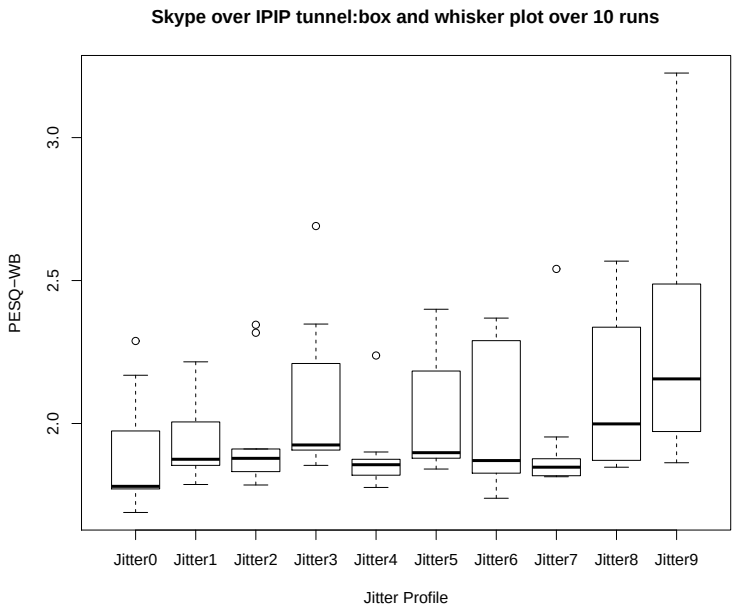


Figure 5.20: Skype IPIP Protocol: Box and whisker plot over 10 runs

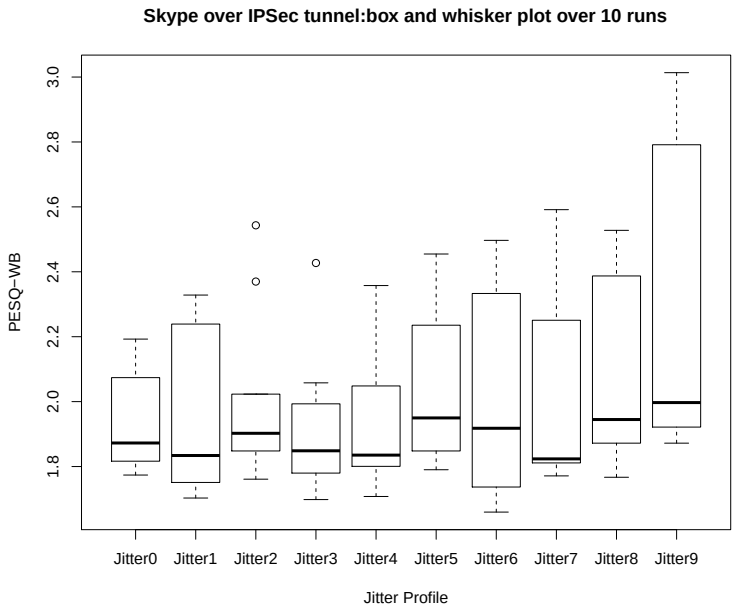


Figure 5.21: Skype IPSec Protocol: Box and whisker plot over 10 runs

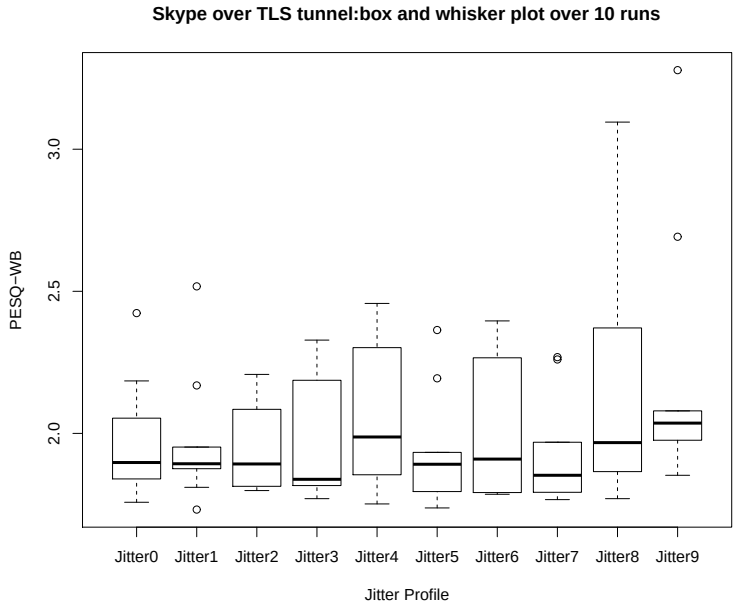


Figure 5.22: Skype TLS Protocol: Box and whisker plot over 10 runs

Our analysis on the Express Talk shows that jitters affect its performances. The mean scores of Express Talk are above 2.0 in all jitter profiles except for Jitter3, Jitter4 and Jitter5 as depicted in Figure 5.17 and Jitter3 and Jitter7 in Figure 5.18 on page 122 and Jitter0, Jitter1, Jitter3 and Jitter5 in Figure 5.19 on page 123 respectively. Generally, this indicates that more than fifty percent of the scores are above 2.0. In addition, the PESQ-WB scores are widely spread within the range of 1.0 to 4.0 approximately with a few outliers. However, to understand the effect of each individual jitter, we analyse each plot separately. For example, referring to Figure 5.17, for Jitter0, the score ranges from 1.3 to 4.1, approximately a 2.8 score spread, which in PESQ-WB scores is quite a bit of difference. The first quartile reading is 1.5 which means that seventy five percent of the scores in this test case are 1.5 or more. The third quartile tells us that twenty five percent of these calls have scored 4.0 or higher which are really good scores. The median cuts the data in half at 3.4. This indicates that fifty percent of the calls received PESQ-WB scores of 3.4 or higher. These scores skew to the right. From this we can conclude that there is a wide range of scores that reflect the effect of Jitter0 to the voice quality. The analysis on Figure 5.18 and Figure 5.19 show similar outcomes except for the Express Talk in TLS the scores skew to the left and for the Express Talk in IPSec, the third quartile starts at 3.0 instead of at 4.0. This shows that for Express Talk, jitters in a secure tunnel may reduce the PESQ-WB scores further.

Generally, the scores of Skype are much lower and uncertain in all jitter profiles, refer to Figure 5.20 and Figure 5.21 on page 124 and Figure 5.22 on page 125 respectively. Regardless

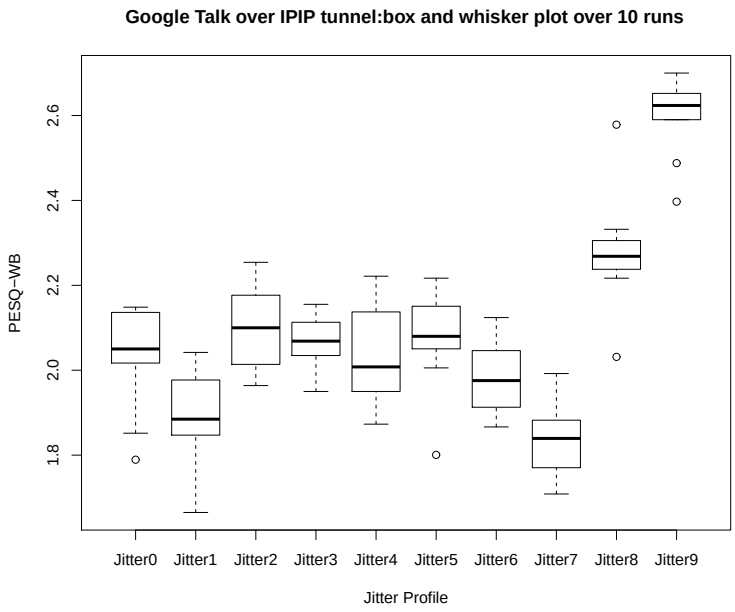


Figure 5.23: Gtalk IPIP Protocol: Box and whisker plot over 10 runs

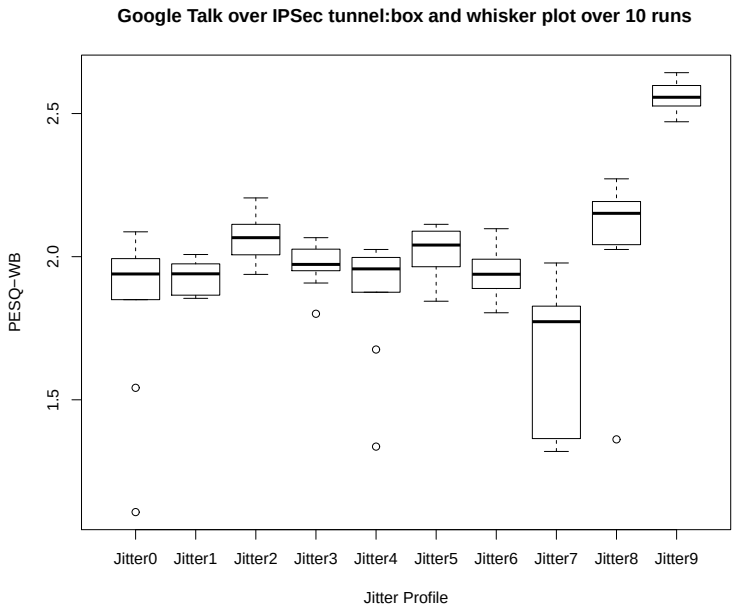


Figure 5.24: Gtalk IPSec Protocol: Box and whisker plot over 10 runs

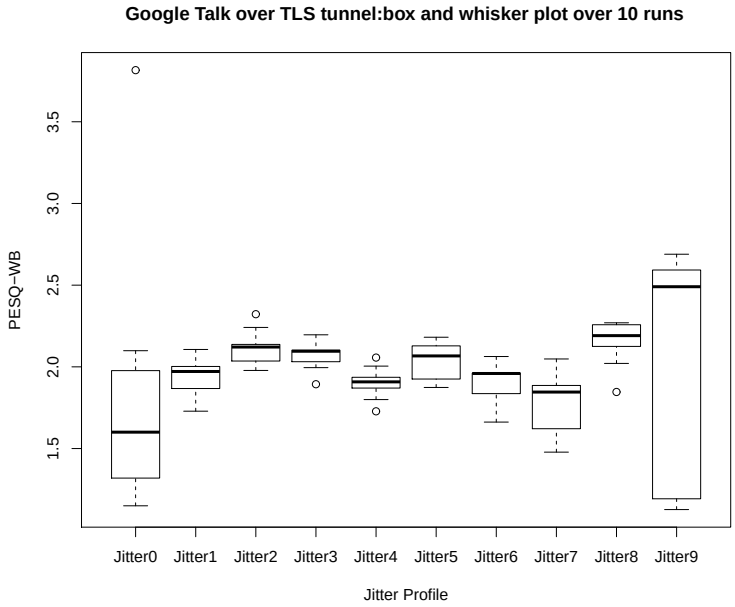


Figure 5.25: Gtalk TLS Protocol: Box and whisker plot over 10 runs

whether the tunnels are secure or unsecure the medians are below 2.0. This shows that fifty percent of the scores are less than 2.0. In most cases the maximum scores are around 2.6 with a few exception, for example Jitter9 in IPIP and IPSec tunnels, whereby the maximum scores are around 3.0. However, the overall scores skew to the left which indicates that jitters cause Skype to have low performances. In addition, there are more outliers as compare to the scores for the Express Talk. In these scenarios, TLS and IPSec have less influence on Skype performances. This shows that Skype is more sensitive to jitters than any other impairments.

Google Talk scores are much higher than Skype but lower than Express Talk. The scores spread is also smaller than Express Talk. This indicates that Google Talk scores are quite consistence with fewer outliers as compared to Skype. Most of the medians are about 2.2, which indicates that fifty percent of the scores are higher than 2.2. Refer to Figure 5.23, Figure 5.24 on page 126 and Figure 5.25 on page 127 respectively. However, in most cases the maximum scores are around 2.5. Any scores less than 2.5 indicate that the services have very low voice quality. Generally, when there were jitters, IPSec and TLS have little influenced on Google Talk performances, refer to Figure 5.24 and Figure 5.25 on page 126 since their scores are at par with those in the IPIP tunnel. The results also indicate that Google Talk is more sensitive to jitters than any other impairments like plr and $delay$.

In summary jitters affect Skype and Google Talk more than Express Talk. In most cases secure or unsecure tunnels have less effect on the overall performances. In case of Skype and Google Talk, jitters cause their scores to be low but the scores distribution has smaller spread

than Express Talk. On top of that, jitters introduce outliers in their data samples. On the other hand, for Express Talk, jitters cause the scores to have wide spread of values. Some of the scores are quite good but some are very low. One way to overcome jitters in any VoIP services is by having an appropriate size of de-jitter buffer. We have discussed about de-jitter buffer briefly in the third paragraph of Subsubsection 2.4.1.2 on page 58.

5.2 Security:Vulnerability Score

5.2.1 CVSS base score

We computed the CVSS base scores for SIP, Google Talk and Skype based on the profiles of the VoIP services as depicted in Table 3.1 on page 82 and Table 4.6 on page 105. Figure 5.26 on page 129 shows these scores. We define the CVSS score in the third paragraph of Subsection 4.6.1 on page 104. The scores range from 0 to 10. The higher the vulnerability, the higher the score. Other than that , the Exploitability Score is also obtained from the CVSS calculator [49]. Overall, SIP has the lowest CVSS base score for each security protocol implemented as compared to Google Talk or Skype. In all conditions the CVSS base scores reduce with the implementation of IPSec or TLS. Both IPSec and TLS protect the confidentiality and integrity of any data that is transmitted through their tunnels [172, 178]. The different between confidentiality and integrity is that, the first refers to limiting information access and disclosure to only authorized users and preventing access by, or disclosure to, unauthorized users. On the other hand, the latter measures the trustworthiness and guaranteed veracity of the information. The confidentiality is maintained if the data is encrypted and only authorized users are allowed access. On the other hand, the integrity is obtained by applying a hashing algorithm like Hash-Based Media Authentication Code (HMAC). When TCP is used as the transport protocol, TLS can be used to protect VoIP messages. TLS, a successor to SSL, is a protocol that ensures privacy between communication applications and their users on the Internet [25, 152, 113]. For a server and client communication, TLS ensures that no third party may eavesdrop or tamper with any message travels between the two devices. IPSec may also be used to protect the layer three communications, regardless of transport protocol types of either TCP or UDP [25, 113]

SIP has several logical servers with dedicated functions. SIP servers located within the internet cloud. According to Salsano, SIP supports two forms of encryption: end-to-end and hop-by-hop. S/MIME performs end-to-end encryption, whereas IPSec and TLS perform hop-by-hop encryption of the whole SIP messages in order to protect the information that should be accessed by intermediate entities, such as From, To, and Via headers. The full description of SIP security mechanisms can be found in [147]. The difference between an end-to-end

Service Name	Protocol	TCP/IP Layer	Access Vector	Access Complexity	Authentication	Exploitability Score	Confidentiality Impact	Integrity Impact	Availability Impact	Impact Score	Base Score
SIP	IPIP	Internetworking	Network	Medium	Multiple Instance	5.5	Partial	Partial	Partial	6.4	**5.4**
SIP	IPSec	Internetworking	Network	High	Multiple Instance	3.2	None	None	Partial	2.9	**1.8**
SIP	OpenVPN TLS	Transport	Network	High	Multiple Instance	3.2	None	None	Partial	2.9	**1.8**
SIP	ZRTP	Application	Network	Medium	Multiple Instance	5.5	None	None	Partial	2.9	**2.8**
Gtalk	IPIP	Internetworking	Network	Low	Single Instance	8	Partial	None	Partial	4.9	**5.5**
Gtalk	IPSec	Internetworking	Network	Medium	Multiple Instance	5.5	None	None	Partial	2.9	**2.8**
Gtalk	OpenVPN TLS	Transport	Network	Medium	Multiple Instance	5.5	None	None	Partial	2.9	**2.8**
Gtalk	ZRTP	Application	Network	Low	Multiple Instance	6.4	None	None	Partial	2.9	**3.3**
Skype	IPIP	Internetworking	Network	Low	Single Instance	8	Partial	None	Partial	4.9	**5.5**
Skype	IPSec	Internetworking	Network	Medium	Multiple Instance	5.5	None	None	Partial	2.9	**2.8**
Skype	OpenVPN TLS	Transport	Network	Medium	Multiple Instance	5.5	None	None	Partial	2.9	**2.8**

Figure 5.26: CVSS Base Score

security and a hop-by-hop security is that for the end-to-end security the whole message is encrypted either using a symmetric or an asymmetric key where the necessary key-variables and algorithms are shared by both parties. The end-to-end encryption involves an uninterrupted protection of the confidentiality and integrity of transmitted data by encoding it at its starting point and decoding it at its destination. It involves encrypting data at source with knowledge of the intended recipient, allowing the encrypted data to travel safely through public networks to its destination where it can be decrypted. For the hop-by-hop encryption, any data transmitted between the two network devices that involve in the tunnelling is protected, refer to Figure 5.27. However there is no guarantee that the data will still be protected when it travels between end points and the network devices.

SIP protocol is defined in Subsection 2.2.1 on page 34. When a call is made, SIP registeration server authenticates user. Typically SIP call signalling travels in the internet cloud from one proxy server to another. SIP has the ability to protect its service from Denial of Service (DoS) attack. SIP has multiple instance of authentication. SIP provides user to user authentication and user to proxy authentication. SIP access complexity is medium because so far there is no evidence of man in the middle (MiTM) attack. However, with DoS protection, this might imply that it is not easy to get into the target in the first place. SIP messages are still in plain text and not encrypted. Therefore Confidentiality Impact is partial since attackers may not be able to break confidentiality if they cannot get access to the stream. With IPSec ESP implementation in place, SIP media is protected but SIP signalling can still be compromised. Hence, IPSec ESP increases confidentiality and integrity of the media in SIP service but it still compromises on confidentiality of the signal. SIP service depends on the internet availability,

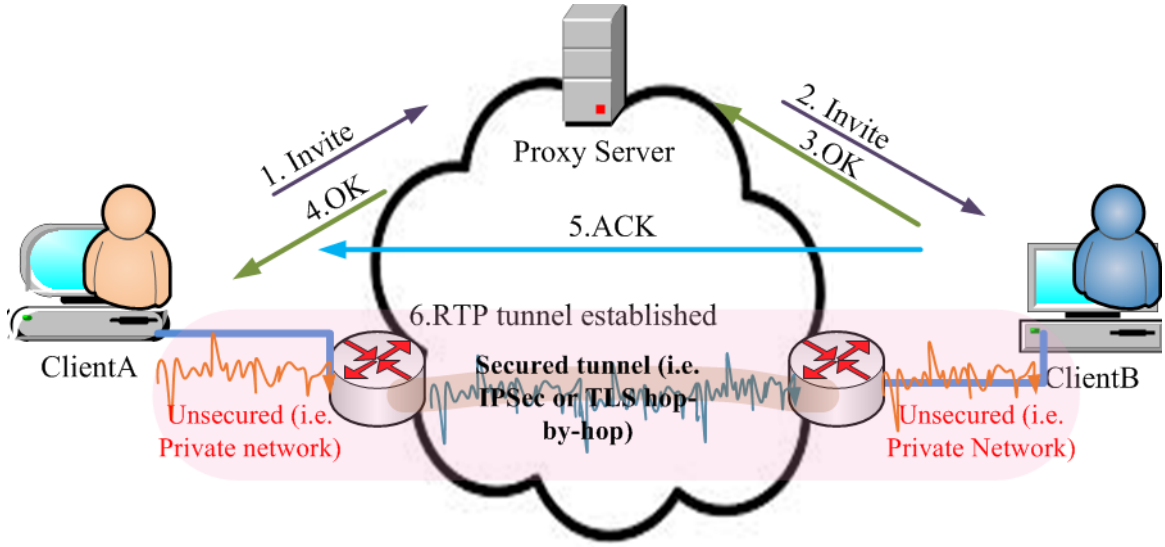


Figure 5.27: Media security

hence we concluded that the availability impact is partial for all scenarios.

Google Talk server is in the internet cloud. Google Talk Access complexity is low although there is a Simple Authentication Security Layer (SASL) in place. XMPP message is in a human readable format hence the confidentiality impact is partial. There is an integrity check therefore the integrity impact might be none. The availability is partial due to the internet best effort. With IPSec ESP implementation in place, the access complexity is still low on the signal but the audio streams are protected. IPSec ESP increases confidentiality and integrity of the media in Google Talk service but it still compromises on the confidentiality of the signal.

Skype has a decentralised user directory. Skype audio is encrypted end-to-end. The problem is, Skype can bypass firewall and NAT. Skype supernode also increase risk to its user. Therefore Skype access complexity is low, authentication is for a single instance, the confidentiality impact is partial and the integrity impact is none. With IPSec ESP implementation, Skype access complexity is medium since Skype signal is still vulnerable but the audio is protected by the IPSec tunnel. Hence, IPSec ESP increases confidentiality and integrity of the media in Skype service and lower the vulnerability score. However, it still compromises on confidentiality of the signal. Since Skype service depends on the internet availability, hence we decided that the availability impact is partial.

In all cases, OpenVPN TLS protects the payload from eavesdropping, tampering and message forgery. However, it still compromises on confidentiality of the signals. Therefore, in terms of media protection, both IPSec ESP and OpenVPN are at par. However, OpenVPN TLS or IPSec is more scalable and secure in term of key exchange mechanism if it uses public key cryptography instead of a pre-shared secret, refer to Section 4.1 on page 88 on the method used in this experiment. Another way to protect the payload is by using the ZRTP protocol

[189]. For example, Zfone could be used to encrypt phone calls over the Internet [188]. Zfone uses ZRTP protocol and it supports SIP and Google Talk but not Skype. Skype does not need this extra protection because its media is already been encrypted. If ZRTP is implemented, the media parts of SIP and Google Talk can also be protected but the signalling parts for both services can still be compromised. However, the Application Layer security is beyond the scope of our project. Hence, it is not included in this experiment.

Figure 5.26 on page 129 shows the base score estimations for the conditions that have been described above. Generally, services aim for low vulnerability scores. SIP with IPSec ESP and OpenVPN TLS implementations have the lowest scores. From the result, it shows that the vulnerability scores improve with the implementation of IPSec ESP and OpenVPN TLS. IPSec ESP and OpenVPN TLS are a few protocols that can be used to reduce the vulnerability on the media part of a VoIP service. The protection of the signal part of any public VoIP service is feasible. However, it is not scalable neither practical except for Internet Service Provider (ISP) that uses an Internet Telephony Service Provider (ITSP) connection to PSTN [147]. The service allows internet users to place calls in the PSTN. In this scenario the ISP already has a security relationship with its customer. For example, a SIP proxy server in the ISP network will be configured as the default outbound proxy server for the SIP clients in the ISP network. This proxy server forwards calls to the ITSP proxy server, which will select and contact the more appropriate SIP gateway. Other than that, a company can opt for a private VoIP service to have a full protection, just like a banking system is doing on the e-banking.

5.3 Bandwidth Utilisation

Bandwidth describes how much information can be transmitted over a connection per unit of time. For VoIP services to function properly, an end-to-end connectivity with an adequate bandwidth must be able to carry enough information to sustain the succession of audio signals. In a connectionless network where data can be routed through any paths from source to destination, the actual bandwidth is as good as the last miles bandwidth. This is because an access network normally has the lowest bandwidth capacity than to core networks that belong to ISP. Throughput, also known as bandwidth utilisation, is the amount of data successfully sent per unit of time. Throughput fluctuates even within the same session and it would not exceed bandwidth limit of a link. The type of link and the codec in used would determine the bandwidth requirement and throughput value. Subsubsection 2.4.2.1 on page 61 explains the method used to calculate the bandwidth requirement for a single call. Table 5.2 on page 132 shows the bandwidth and sample packet size in kbps that are calculated using this method.

Google Talk, Express Talk and Skype negotiate the codec type during calls setup. Apparently, Google Talk supports iLBC, Speex, G.722 and G.711 [57]. On the other hand, Express

Talk supports G.711u, G.711a and GSM with echo cancellation and noise reduction [167]. Express Talk might be using a new extension of ITU-T G.711 to cater for a wideband coding scheme known as the ITU-T wideband extension (G.711.1). The G.711.1 operates at 64, 80, and 96 kbps, and is designed to achieve very short delay and low complexity [69]. Lapierre et al. present a noise shaping scheme so that a multi-rate codec is interoperable with the legacy narrow-band codec [96]. As a result, it increases the intelligibility and naturalness of speech and offer a better face-to-face experience to the end users. Another Express Talk supported codec, GSM is less relevant for our research since it is used to support a cellular telephony service. Skype supports iLBC, G.711, G.729 and Speex [154]. In addition, Skype has several proprietary codecs like SILK[2], a super wideband speech coding standard, developed by Skype in the year 2009 and SVOPC[3], a lossy speech compression codec designed specifically towards communication channels suffering from packet loss. SVOPC coder is inherently robust to packet loss but uses more bandwidth than the best bandwidth-optimised codecs [101]. These codecs are either CBR or VBR , refer to Subsection 2.2.7 on page 40 for explanation.

Table 5.2: VoIP per call bandwidth per CODEC type

Codec type	Standard	VBR/ CBR	Sample period	No. of frame per packet	Packet per second	Codec bit rate	Total packet size (bits)	Bandwidth per call
G.711 (PCM)	ITU-T	CBR (Narrowband)	20ms	1	50 pps	64 kbps	1904 bits	95.2 kbps
G.711.1 (PCM)	ITU-T	CBR (Wideband)	20ms	1	50 pps	80 kbps	2224 bits	111.2 kbps
G.711.1 (PCM)	ITU-T	CBR (Wideband)	20ms	1	50 pps	96 kbps	2544 bits	127.2 kbps
G.729A (CS-CELP)	ITU-T	CBR	10ms	2	50 pps	8 kbps	784 bits	39.2 kbps
G.729	ITU-T	CBR	10ms	2	50 pps	8 kbps	224 bits	11.2 kbps
G.723.1A (ACELP)	ITU-T	CBR	30ms	1	33.3 pps	5.3 kbps	784 bits	26.1 kbps
iLBC	IETF (RFC 3951)	CBR	20ms	1	50 pps	15.2 kbps	928 bits	46.4 kbps
iLBC	IETF (RFC 3951	CBR	30ms	1	33.3 pps	13.33 kbps	1024 bits	34.1 kbps
Speex	Open Source	VBR	20ms	1	50 pps	4 kbps to 44.2 kbps	704 bits to 1512 bits	35.2kbps to 75.4kbps

In theory, VoIP over ethernet with legacy G.711 codec requires 95.2kbps bandwidth in

[2]Super Wideband Audio Codec
[3]Sinusoidal Voice Over Packet Coder

order to sustain the succession of audio signals in the VoIP service. There is no data compression for this codec [119]. However, there would be some trade-off in term of voice quality and bandwidth requirement. Other codec types would have some form of data compression in them. Hence, less bandwidth is required to sustain the succession of audio signals in the VoIP service. However, the voice quality would decrease due to the data compression. The payload size for the G.711 codec with 64kbps bandwidth and 20ms sample period is 160 bytes. On top of that there is a fixed IP overhead of 40 bytes and fixed Ethernet overhead of 38 bytes. Hence, the total IP packet size is 238 bytes. Since normal speech consists of talkspurts that last for a few hundred milliseconds and silence periods that occur within spoken words and between words, hence, ideally the packet size for each talkspurt would be around 1904 bits.

On the other hand, VoIP over ethernet with iLBC codec requires 46.4kbps bandwidth in order to sustain the succession of audio signals in the VoIP service. The iLBC codec enables graceful speech quality degradation in the case of lost frames, which occurs in connection with lost or delayed IP packets. The payload size for iLBC[4] codec with 15.2kbps bandwidth and 20ms sample period is 38 bytes. On top of that there is a fixed IP overhead of 40 bytes and fixed Ethernet overhead of 38 bytes. Hence, the total IP packet size is 116 bytes. Hence, the packet size for each talkspurt would be around 928 bits, an estimated size for any VoIP services with iLBC. VoIP over ethernet with G.729A codec requires 39.2kbps bandwidth in order to sustain the succession of audio signals in the VoIP service. The payload size for G.729A codec with 8kbps bandwidth, 10ms sample period and 2 frame/packet is 20 bytes. On top of that, there is a fixed IP overhead of 40 bytes and fixed Ethernet overhead of 38 bytes. Hence, the total IP packet size is 98 bytes. The packet size for each talkspurt would be around 784 bits.

Another type of codec supported by Skype, Google Talk and Express Talk is Speex which is an Open Source Software patent-free audio compression format designed for speech. The Speex Project aims to lower the barrier of entry for voice applications by providing a free alternative to expensive proprietary speech codecs. Moreover, Speex is well-adapted to Internet applications and provides useful features that are not present in most other codecs, for examples it is robust to corruption at the bit level, as found on wireless networks, embeds narrowband bitstreams in wideband bitstreams and variable bit rate (VBR) [182]. In particular, Speex codec uses multirate, and supports ultra-wideband (i.e. 32 kHz sampling rate), wideband (i.e. 16 kHz sampling rate) and narrowband (i.e. telephone quality, 8 kHz sampling rate). Multirate capability allows the codec to change bitrate dynamically, at any moment. The payload size for Speex wideband codec with bandwidth between 4kbps to 44.2kbps, 20ms sample period is between 10 to 111 bytes. There is also a fixed IP overhead of 40 bytes and fixed Eth-

[4]There is another one with 13.33 kbps with an encoding frame length of 30ms

ernet overhead of 38 bytes. Therefore, VoIP over ethernet with Speex wideband codec requires bandwidth between 35.2kbps to 75.4kbps in order to sustain the succession of audio signals in the VoIP service. The packet size for each talkspurt would be between 704 bits to 1508 bits.

In our experiment, the throughput for each VoIP service is measured every 10 seconds, (i.e. data transfer time T is set to 10 seconds, refer to page 102). The throughput for each service is represented in the form of time versus throughput graph. Subsection C.1.2 on page xxxiii in Appendix C shows the steps on how we draw the graphs of throughput from the .pcap data that we collect using the method as described in the Subsection 4.6.2 on page 108. For examples, the graphs on page 135 show the throughput for SIP clients, the graphs on page 141 show the throughput for Skype clients and the graphs on page 144 show the throughput for Google Talk clients. There are many different flavour of SIP clients. In this experiment we use Express Talk as our SIP client. We begin our analysis on the Express Talk implementation in the IPIP tunnel. We want to find out the effect of $delay$, plr and $jitter$ on throughput value based on the graphs shown on page 135. The graphs, refer to Figure 5.28, Figure 5.29, Figure 5.30 and Figure 5.31, on $delay$ show that $delay$ influences the amount of throughput passes through the network from a sender to a receiver. However, the maximum amount of throughput for each talkspurt remains constant throughout the experiment. We believe Express Talk uses a CBR type of CODEC like G.711.1 as mentioned in [167]. The graphs also demonstrate that the throughputs are sensitive to changes and disturbance in the network. For example, excessive $delay$ might result in packets loss during transmission or drop at the gateway, refer to the graph of Figure 5.31 for $delay$ between 300ms to 400ms, on page 135. There are more time gaps between the adjacent packets in this graph as compared to the graph in Figure 5.28 for $delay$ between 0ms to 90ms on page 135. Another condition that creates wider gaps between adjacent packets are when there are jitters in the network as shown by the graph in Figure 5.35 on page 135.

Figure 5.32, Figure 5.33 and Figure 5.34 show that plr values also influence the amount of throughput passes through the network from a sender to a receiver. Based on Figure 5.35 on page 135, it seems that the maximum throughput remains constant throughout the experiment. However, there are more obvious gaps in the graph. Throughputs may also affect the PESQ-WB scores. We believe that there is a correlation between the throughput and the available network resources. However, the maximum throughput of any CBR codec is a constant value and is never beyond a sample packet size travels every seconds. From the graphs, we notice that the maximum packet travels in one second for each condition is around 1400 bits. Subsubsection 2.4.2.1 on page 61 explains on how to calculate a bandwidth requirement for a single call for a different codec type. Referring to Table 5.2 on page 132, the packet size for VoIP/UDP/IP/Ethernet is about 1904 bits. This value is higher than the experimental value because in the experiment we extract IP packets, hence our script exclude any Layer 2 frame

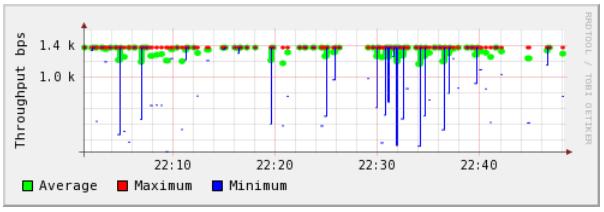


Figure 5.28: Express Talk: Delay 0ms to 90ms

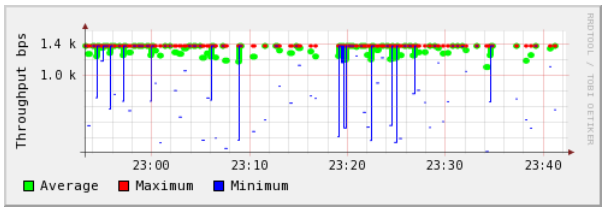


Figure 5.29: Express Talk: Delay 100ms to 190ms

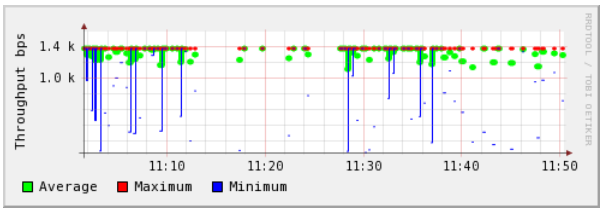


Figure 5.30: Express Talk: Delay 200ms to 290ms

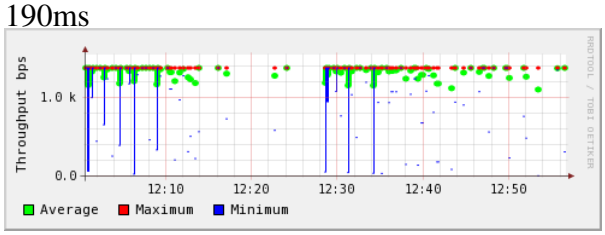


Figure 5.31: Express Talk: Delay 300ms to 400ms

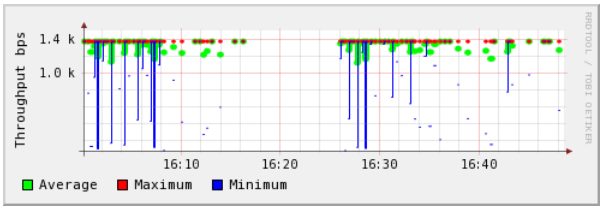


Figure 5.32: Express Talk: PLR 0.0% to 0.9%

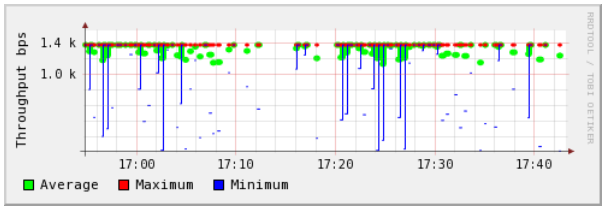


Figure 5.33: Express Talk: PLR 1.0% to 1.9%

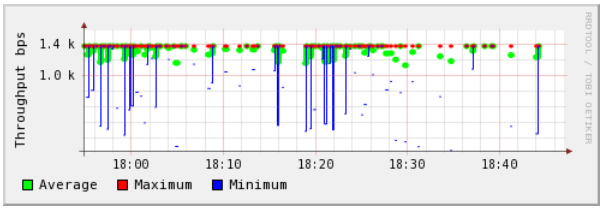


Figure 5.34: Express Talk: PLR 2.0% to 3.0%

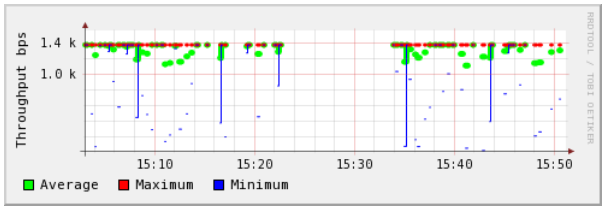


Figure 5.35: Express Talk: 9 Jitter profiles

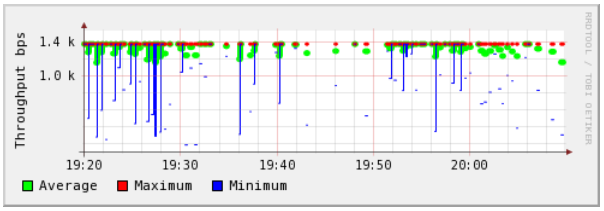


Figure 5.36: Express Talk-IPSec: Delay 0ms to 90ms

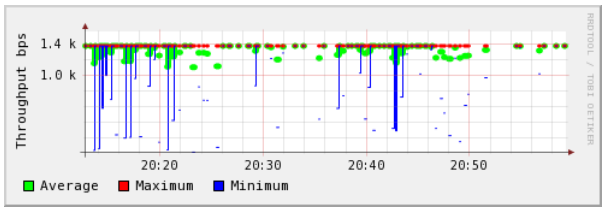


Figure 5.37: Express Talk-IPSec: Delay 100ms to 190ms

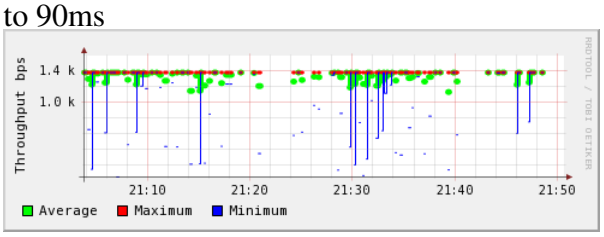


Figure 5.38: Express Talk-IPSec: Delay 200ms to 290ms

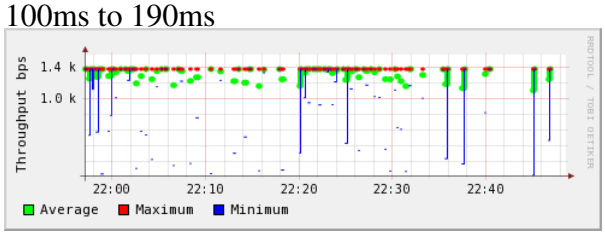


Figure 5.39: Express Talk-IPSec: Delay 300ms to 400ms

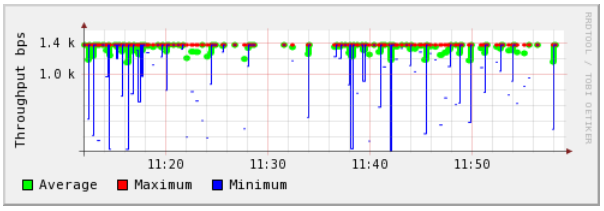


Figure 5.40: Express Talk-IPSec: PLR 0.0% to 0.9%

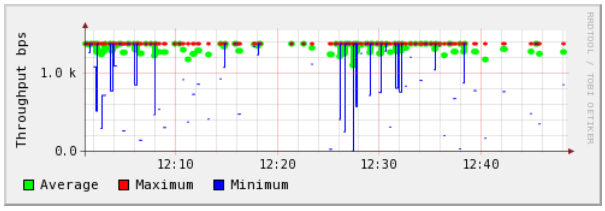


Figure 5.41: Express Talk-IPSec: PLR 1.0% to 1.9%

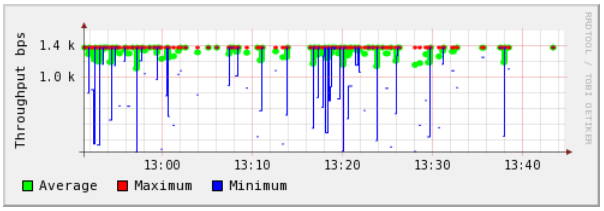


Figure 5.42: Express Talk-IPSec: PLR 2.0% to 3.0%

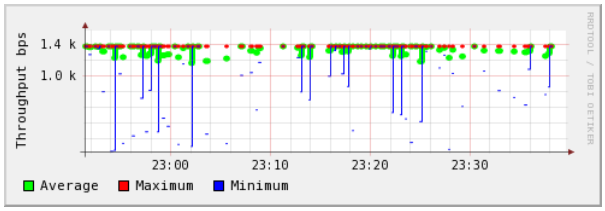


Figure 5.43: Express Talk-IPSec: 9 Jitter profiles

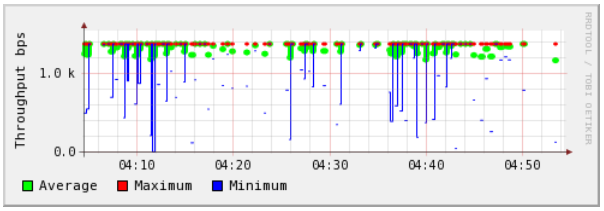


Figure 5.44: Express Talk-TLS:Delay 0ms to 90ms

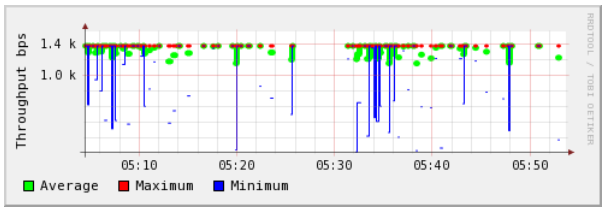


Figure 5.45: Express Talk-TLS:Delay 100ms to 190ms

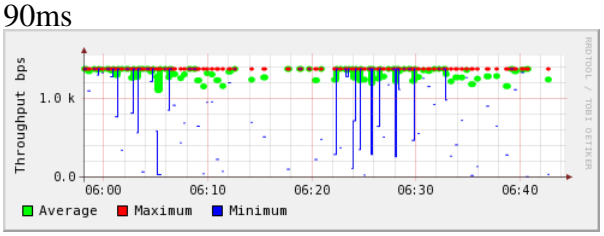


Figure 5.46: Express Talk-TLS:Delay 200ms to 290ms

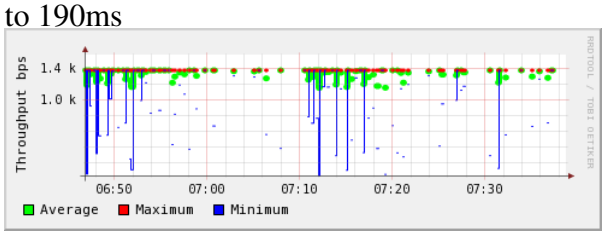


Figure 5.47: Express Talk-TLS:Delay 300ms to 400ms

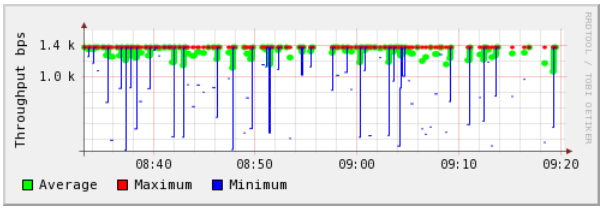


Figure 5.48: Express Talk-TLS:PLR 0.0% to 0.9%

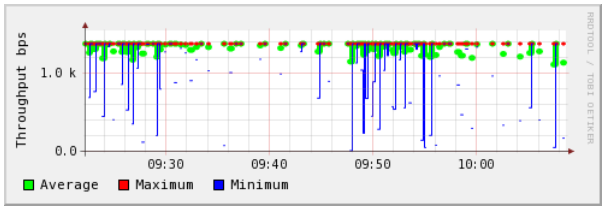


Figure 5.49: Express Talk-TLS:PLR 1.0% to 1.9%

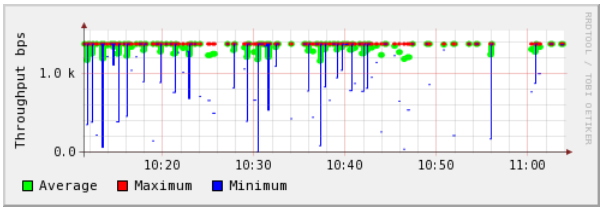


Figure 5.50: Express Talk-TLS:PLR 2.0% to 3.0%

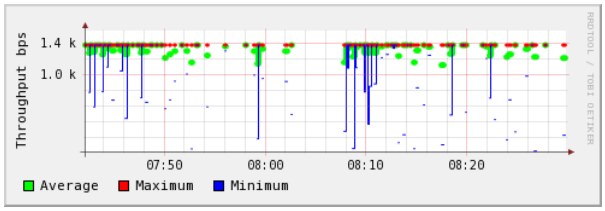


Figure 5.51: Express Talk-TLS:9 Jitter profiles

header, refer to Subsection C.1.2 on page xxxiii in Appendix .

Once we have analysed the throughputs in the IPIP tunnel we continue our analysis on throughputs in IPSec tunnel. From our observation, Express Talk in the IPSec reveals similar results, refer to the graphs on page 136. We believe IPSec has some impact on the amount of throughput transmitted from a sender to a receiver. However, since the .pcap data was collected at the gateway, hence we could not see the difference. At the gateway the packets were decapsulated by the IPSec tunnel. However, the size of the throughput confirms that VoIP packet's size is very small. In fact the total throughput is smaller than the calculated bandwidth required to make a single VoIP call. Express Talk in the TLS, refer to the graphs on page 137, reveals similar pattern to the IPIP and IPSec tunnels. These experiments show that Express Talk drops packets or decreases throughput during resources constraint.

Next, we analyse the amount of throughput produce while running Skype. We scrutinize Skype throughput on the IPIP tunnel. The graphs on $delay$ on page 141 show that Skype produced less throughput than Express Talk, refer to the graphs on page 135, and Express Talk PESQ-WB scores are better than Skype, refer to Figure 5.8 on page 116. However, within the Skype, it seems to suggest that there is a direct correlation between the throughput and the PESQ-WB. For example Figure 5.52 on page 141 and Figure 5.6 on page 114 seem to show that there are correlations between high throughput (i.e. at about 15:15 hour and 15:20 hour) and high PESQ-WB (i.e. between 10ms to 50ms $delay$) score and low throughput (i.e. after 15:20 hour) and low PESQ-WB score (i.e. after 50ms $delay$). We consider the PESQ-WB is high when the score is equal or above 2.5 and low if the score is less than 2.5. The PESQ-WB scores are low when there are packet losses. This is evidenced by the low PESQ-WB scores when there are very few throughput between 16:20 hour and 16:30 hour (refer to Figure 5.54 on page 141). The PESQ-WB scores of Skype reduce gracefully as compared to Google Talk, refer to to Figure 5.8 on page 116. This might suggest that Skype uses a codec that adapts to a congested network. It employs embedded or layered coding codec which allows the speech quality graceful degradation in the congested network [101]. This type of codec does not rely on interframe coding techniques for frame-erasure channel instead on a sinusoidal voice over packet coder (i.e. SVOPC) to avoid interframe erasure from propagate over several consecutive frames [101]. The graphs on plr on page 141 seem to show that Skype would adjust on the types of codecs use, refer to Subsection 2.2.7 on page 40, depending on the amount of available resources. In most cases the throughput is about 0.6 kbps. The throughput is not constant. There are sudden burst of throughput from time to time. However, these sudden burst of packets do not influence the PESQ-WB scores, refer to Figure 5.14 on page 120. Figure 5.18 on page 122 shows the PESQ-WB scores for Skype, Express Talk and Google Talk in the IPIP network for several jitter profiles, refer to Table 5.1 on page 123. Skype generates less throughput than Express Talk, refer to Figure 5.20 on page 124, except during start up. The

high throughput during start up might be due to the extra processing for Skype to adapt to the available resources and negotiates on the coder type between a sender and a receiver. The graphs on $delay$ in the IPSec for Skype reveals similar behaviour, refer to the figures 5.60 to 5.63 on page 142. However, Figure 5.61 for $delay$ between 100ms to 190ms seems to suggest that when there are network resource constraints, Skype waits until there are available network resources for the next call to succeed. This is evidenced by the long gap in the graph before the next call being made. In other situation Skype might drop the packets to conserve resources as shown in Figure 5.62 for $delay$ between 200ms to 290ms for the time between 17:10 hour and 17:20 hour. Due to that Skype PESQ-WB scores drop, refer to Figure 5.9 on page 116. IPSec introduces extra processing in Skype which affects Skype performance. However, Figure 5.12 on page 118 seems to suggest that Skype PESQ-WB scores are better in the IPSec as compared to the IPIP. The difference between the minimum throughput and the maximum throughput are small, refer to Figure 5.64 on page 142. There are also sudden burst of packets for plr between 1.0% and 1.9%. In which case, the PESQ-WB score is better than the adjacent plr values. For example when there is a burst at 18:47 hour, the PESQ-WB also increase, refer to Figure 5.65 on page 142 and Figure 5.12 on page 118. Skype throughput for $jitter$ in the IPSec is quite similar to those in the IPIP. This suggests that Skype PESQ-WB scores in the IPSec are random but the throughput remain the same. Similarly, the throughput values are higher during start up. Skype throughput values in the TLS also reveal almost similar patterns as in the IPIP and IPSec except there are no packet losses and the throughput during start up is much higher than the subsequent periods for $delay$ between 200ms to 290ms, refer to Figure 5.69 on page 143. Another interesting observation is for Skype in the TLS for plr between 0.0% to 0.9%, refer to Figure 5.72 on page 143. The throughput produced is quite similar to the throughput produced in the IPIP network, refer to Figure 5.56 on page 141. However, for $jitter$ the throughput resembles to the throughput produced under IPSec, refer to Figure 5.75 on page 143 and Figure 5.59 on page 141. Last but not least, we analyse the throughput produced by Google Talk. We start with the throughput produced in the IPIP tunnel, refer to the graphs on $delay$ on page 144. The throughput fluctuates between 0kbps and 1.2kbps for $delay$ between 0ms to 90ms, refer to Figure 5.76 on page 144. The amount of throughput produced is reflected by high PESQ-WB score, refer to Figure 5.7 on page 115. The throughput produced decreases for $delay$ that is between 100ms and 190ms, refer to Figure 5.77 on page 144. Consequently the PESQ-WB score is also reduced, refer to Figure 5.7 on page 115. The throughput produced seems to suggest that Google Talk uses a variable bit rate codec such as Speex or iLBC. There is a correlation between the throughput and the PESQ-WB score. For example the PESQ-WB scores are low for the $delay$ between 300ms to 400ms, refer to 5.79 on page 144, due to bandwidth constraint and packet losses. For plr, refer to Figure 5.80 on page 144 and Figure 5.13 on page 119, Google Talk waits until there

is enough network resources available to initiate another call. This is evidenced by the gap between the 12:00 hour and 12:10 hour of Figure 5.80 and the PESQ-WB scores are higher than 2.5 for the plr range. Another example is on Figure 5.82 on page 144, there is a gap before a successful call being made at about 13:36 hour. We believe that Google Talk will initiate a call when there is enough network resources otherwise the call is dropped. However, Google Talk produces less throughput when there are jitters in the network. This is evidenced by the low throughput and low PESQ-WB score for $jitter$, refer to Figure 5.83 and Figure 5.21 on page 144 and page 124 respectively. We see similar patterns appear for Google Talk in the IPSec and TLS implementations. The graphs for Google Talk in the IPSec implementation is on page 145 and the graphs for Google Talk in the TLS implementation is on page 146 respectively.

In summary, the experiment shows that Express Talk, Skype and Google Talk generate small amount of throughputs. This is as expected since VoIP payloads are small generally, i.e. approximately 160 bytes and the sizes are smaller for any VoIP codecs with compression. IPSec and TLS do not increase the size of the throughput produced. This is because the VoIP payload is encapsulated within the IPSec or TLS headers when transmitted within the security tunnel and is decapsulated at the gateway before entering the end user device. We have set the tcpdump to monitor the packets travelling from a sender to a receiver by mirroring the sender and receiver PCs and relevent router ports into the DELL switch, refer to the network setup on Section 4.1 on page 88. In which case tcpdump fails to monitor the packets when they are in either IPSec or TLS tunnels. As such we set the tcpdump to sniff the voice packets at the gateway that is closer to the receiver end. In addition, the reason for the long time gaps in the time versus throughput graphs might be due to the VoIP mechanism to overcome any network resources constraint whereby during any disturbance, the overall end-to-end delay is prolong or some packets are dropped. The overall effect of the mechanism would be to reduce the quality of the VoIP services.

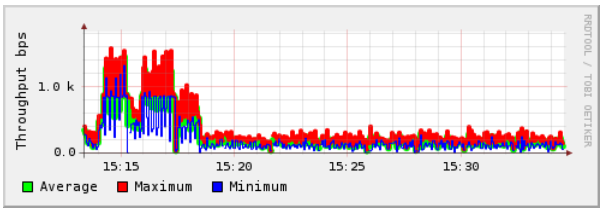


Figure 5.52: Skype: Delay 0ms to 90ms

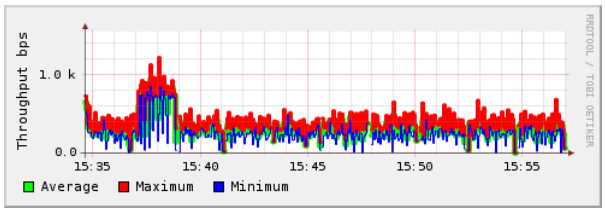


Figure 5.53: Skype: Delay 100ms to 190ms

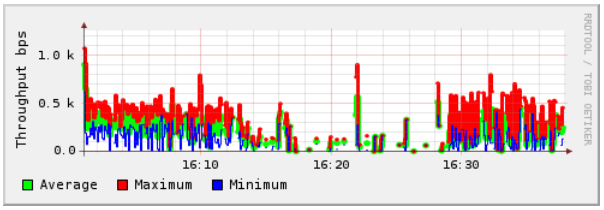


Figure 5.54: Skype: Delay 200ms to 290ms

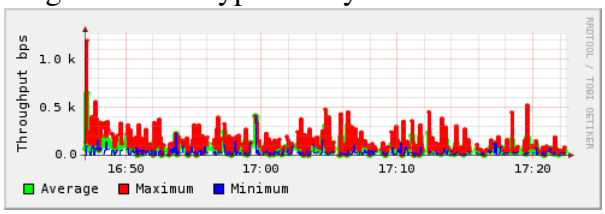


Figure 5.55: Skype: Delay 300ms to 400ms

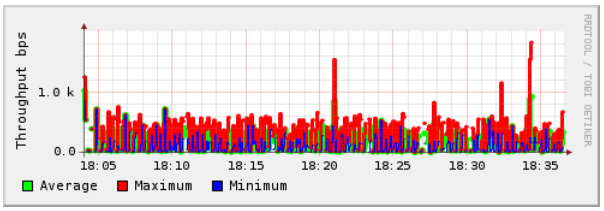


Figure 5.56: Skype: PLR 0.0% to 0.9%

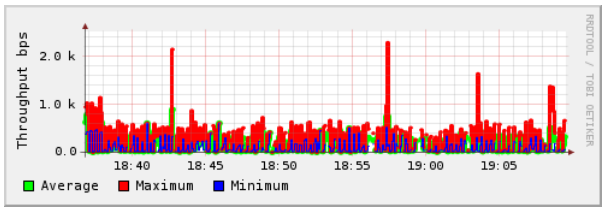


Figure 5.57: Skype: PLR 1.0% to 1.9%

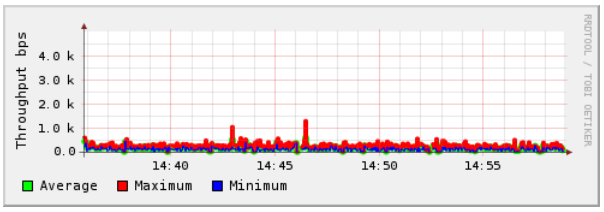


Figure 5.58: Skype: PLR 2.0% to 3.0%

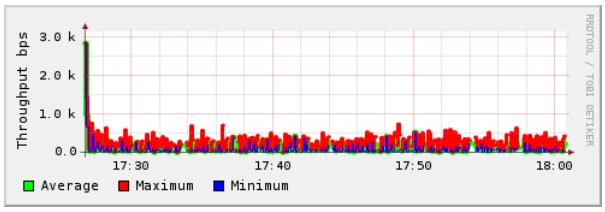


Figure 5.59: Skype: 9 Jitter profiles

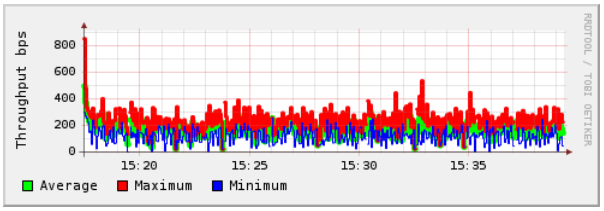


Figure 5.60: Skype-IPSec: Delay 0ms to 90ms

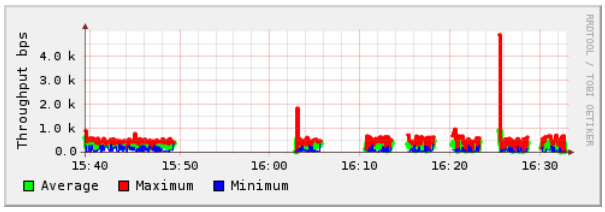


Figure 5.61: Skype-IPSec: Delay 100ms to 190ms

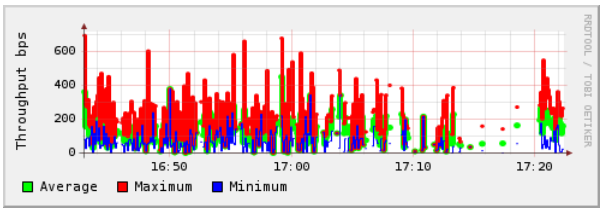


Figure 5.62: Skype-IPSec: Delay 200ms to 290ms

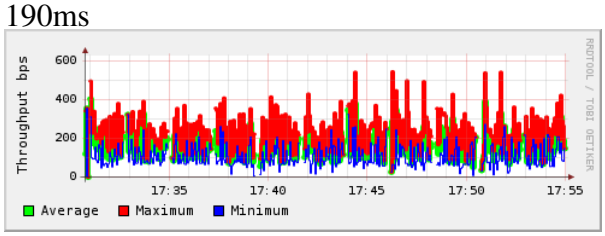


Figure 5.63: Skype-IPSec: Delay 300ms to 400ms

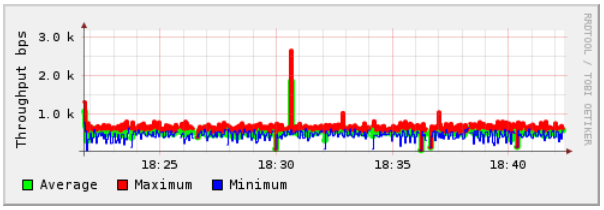


Figure 5.64: Skype-IPSec: PLR 0.0% to 0.9%

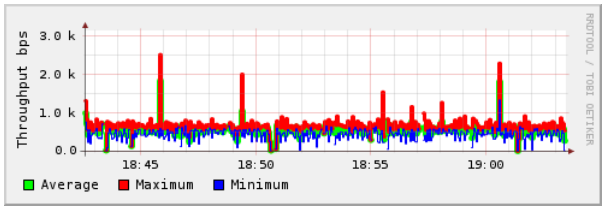


Figure 5.65: Skype-IPSec: PLR 1.0% to 1.9%

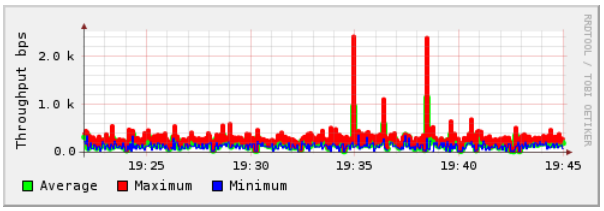


Figure 5.66: Skype-IPSec: PLR 2.0% to 3.0%

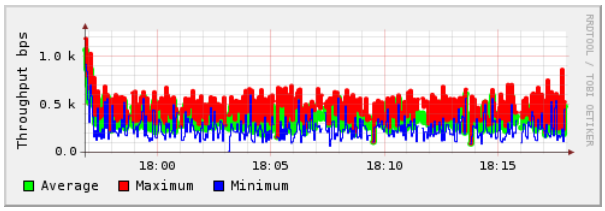


Figure 5.67: Skype-IPSec: 9 Jitter profiles

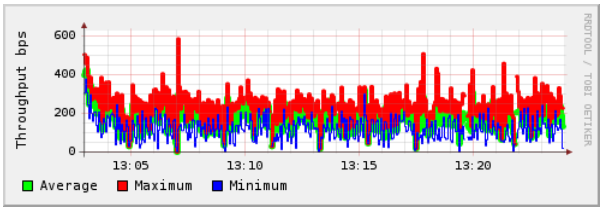


Figure 5.68: Skype-TLS: Delay 0ms to 90ms

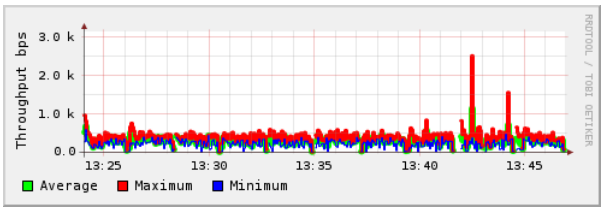


Figure 5.69: Skype-TLS: Delay 100ms to 190ms

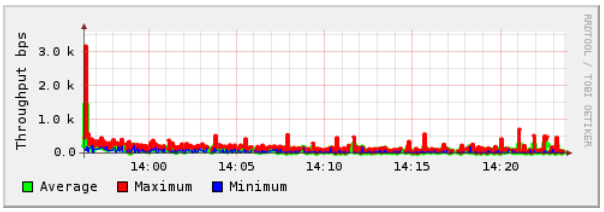


Figure 5.70: Skype-TLS: Delay 200ms to 290ms

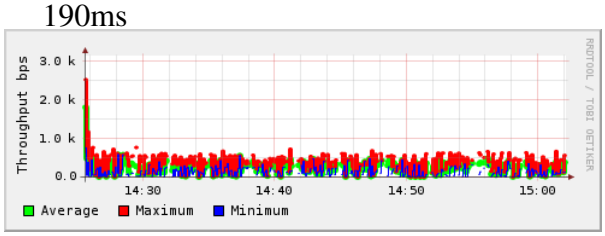


Figure 5.71: Skype-TLS: Delay 300ms to 400ms

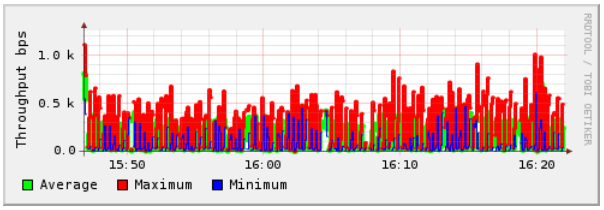


Figure 5.72: Skype-TLS: PLR 0.0% to 0.9%

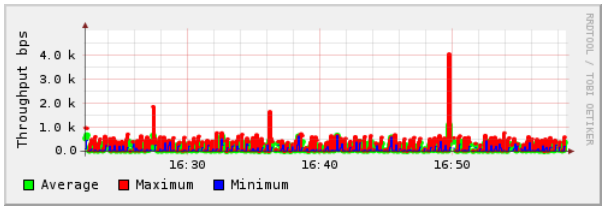


Figure 5.73: Skype-TLS: PLR 1.0% to 1.9%

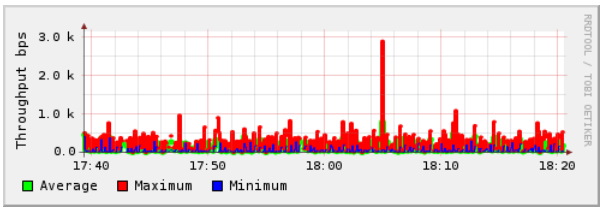


Figure 5.74: Skype-TLS: PLR 2.0% to 3.0%

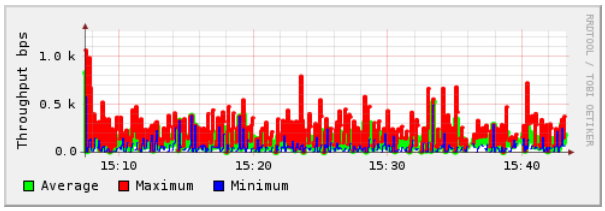


Figure 5.75: Skype-TLS: 9 Jitter profiles

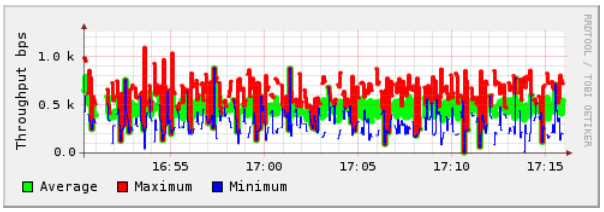


Figure 5.76: GTalk: Delay 0ms to 90ms

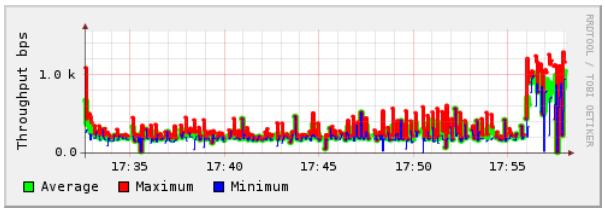


Figure 5.77: GTalk: Delay 100ms to 190ms

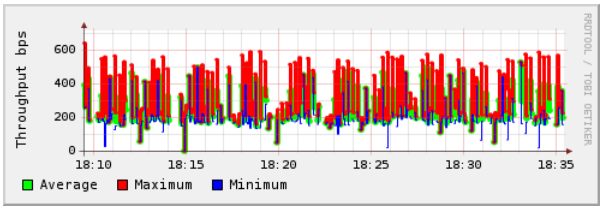


Figure 5.78: GTalk: Delay 200ms to 290ms

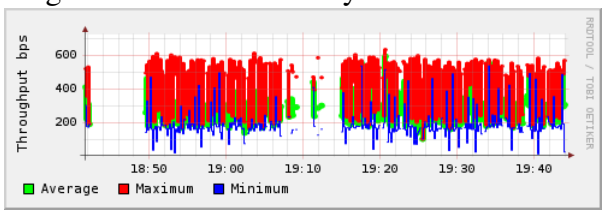


Figure 5.79: GTalk: Delay 300ms to 400ms

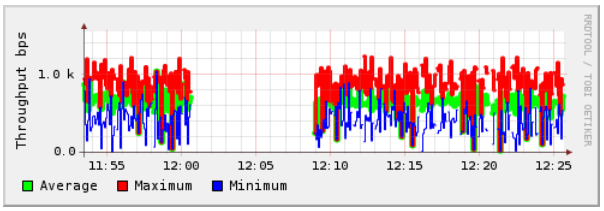


Figure 5.80: GTalk: PLR 0.0% to 0.9%

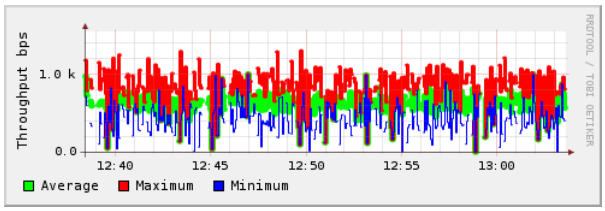


Figure 5.81: GTalk: PLR 1.0% to 1.9%

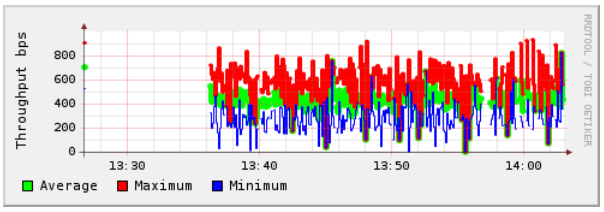


Figure 5.82: GTalk: PLR 2.0% to 3.0%

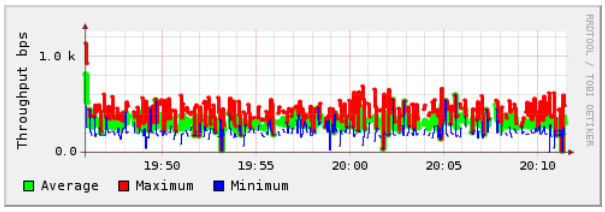


Figure 5.83: GTalk: 9 Jitter profiles

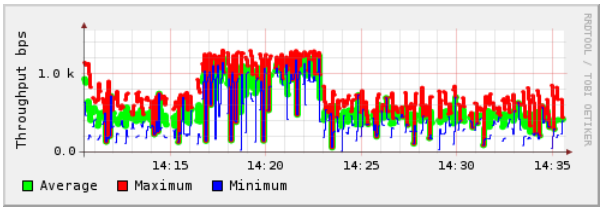


Figure 5.84: Gtalk-IPSec:Delay 0ms to 90ms

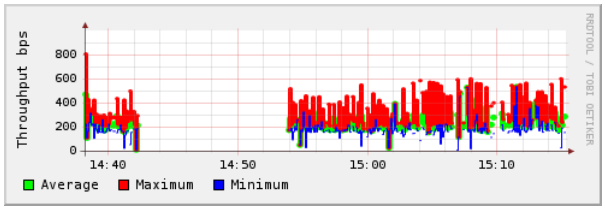


Figure 5.85: Gtalk-IPSec:Delay 100ms to 190ms

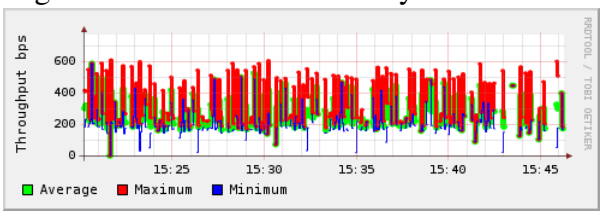


Figure 5.86: Gtalk-IPSec:Delay 200ms to 290ms

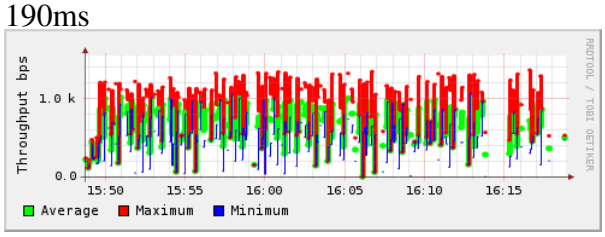


Figure 5.87: Gtalk-IPSec:Delay 300ms to 400ms

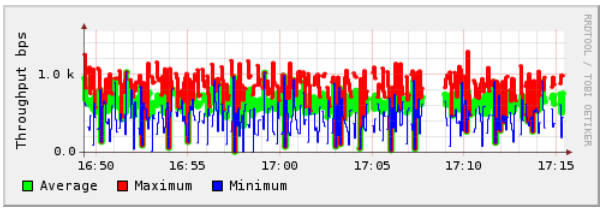


Figure 5.88: Gtalk-IPSec:PLR 0.0% to 0.9%

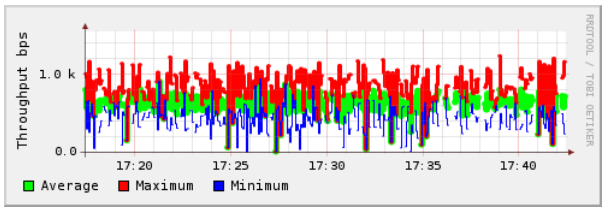


Figure 5.89: Gtalk-IPSec:PLR 1.0% to 1.9%

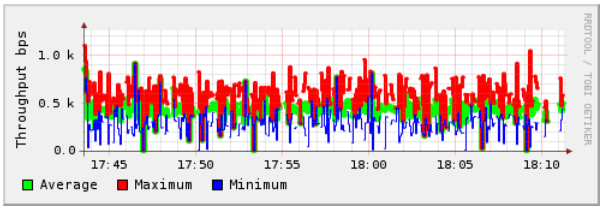


Figure 5.90: Gtalk-IPSec:PLR 2.0% to 3.0%

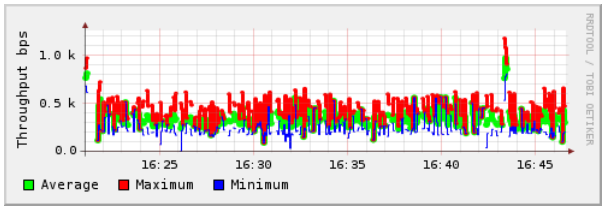


Figure 5.91: Gtalk-IPSec:9 Jitter profiles

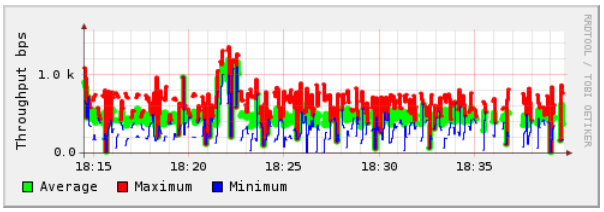


Figure 5.92: Gtalk-TLS:Delay 0ms to 90ms

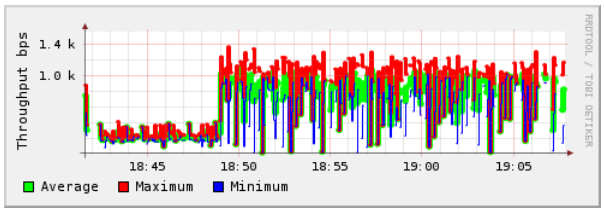


Figure 5.93: Gtalk-TLS:Delay 100ms to 190ms

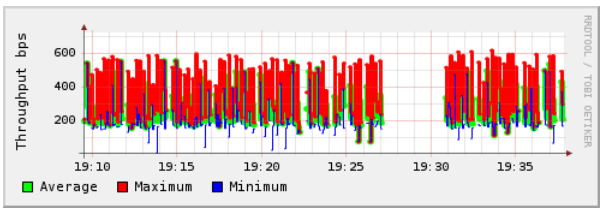


Figure 5.94: Gtalk-TLS:Delay 200ms to 290ms

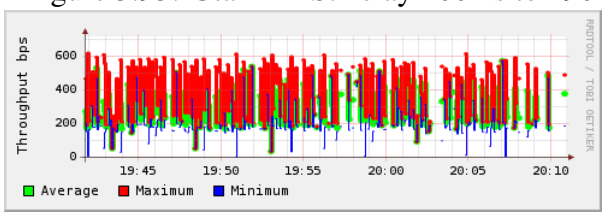


Figure 5.95: Gtalk-TLS:Delay 300ms to 400ms

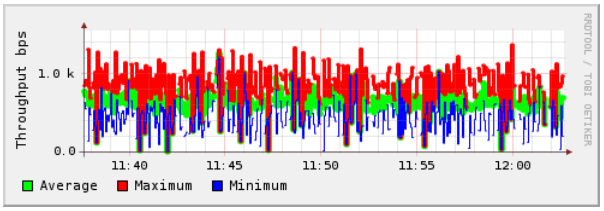


Figure 5.96: Gtalk-TLS:PLR 0.0% to 0.9%

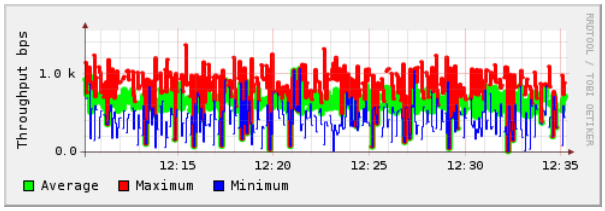


Figure 5.97: Gtalk-TLS:PLR 1.0% to 1.9%

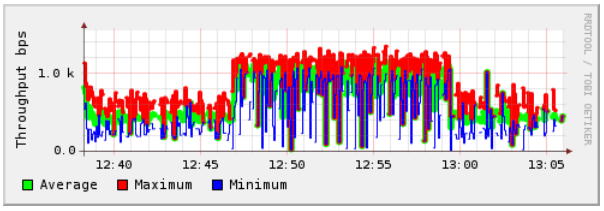


Figure 5.98: Gtalk-TLS:PLR 2.0% to 3.0%

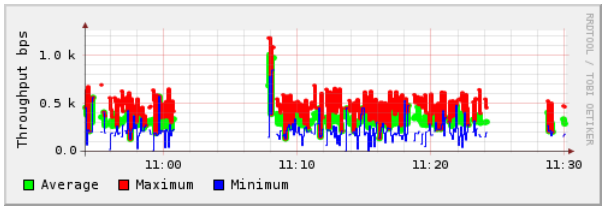


Figure 5.99: Gtalk-TLS:9 Jitter profiles

5.4 Overall Scores

Table 5.3 on page 148 shows the overall performance scores for Express Talk, Skype and Google Talk. There are several conclusions that can be derived from this research:

1. The three VoIP services under test are more sensitive to the impairments due to plr and $jitter$ rather than to the impairment due to $delay$.

2. Bandwidth and other resources like a de-jitter buffer and a gateway's CPU and memory are important in order to produce a good quality VoIP service. The lack of these resources would result in several packets lost before they reach the destination or the packets arrive too late to join the other packets in the de-jitter buffer at the destination gateway. The gateway would drop these packets if the de-jitter buffer is full or not enough memory or CPU powers to process the packets.

3. The gateway closer to the receiver end decapsulates IPSec or TLS packets. The gateway also decodes the voice packets before the packets entering the receiver machine.

4. High throughputs do not imply high PESQ-WB scores. The throughput size is determined by the codec type and the security features that are implemented on the infrastructure.

5. Google Talk PESQ-WB scores are better than Skype for any $delay$ less than 150ms and plr that is less than 1%, those factors are within the recommended ITU-T Recommendation values.

6. Most of the Skype PESQ-WB scores due to plr impairments are below 2.5 which indicate poor voice quality than acceptable telephony quality.

7. In most cases, Express Talk PESQ-WB scores are quite high which indicates good voice quality, generally.

8. Application Layer security, Internetworking Layer security and Transport Layer security reduce the vulnerabilities score for the system. For example the CVSS for SIP in the IPIP is 5.4 and it is reduced to 1.8 in the IPSec and TLS implementations, refer to the CVSS scores on Table 5.3.

Table 5.3: Overall Performance

Categories	Parameters	Express Talk	Skype	Google Talk
Voice Quality	$Delay$ between 0ms and 150ms	Good PESQ-WB scores. The PESQ-WB scores are above 2.5, refer to Figure 5.5 on page 114	PESQ-WB is between 2.5 and 3. The scores gradually decrease, refer to Figure 5.6 on page 114.	PESQ-WB linearly decrease. PESQ-WB score is between 2.3 and 3, refer to Figure 5.7 on page 115.
	$Delay$ between 150ms and 400ms	PESQ-WB scores are above 2.5.	PESQ-WB score is below 2.5	PESQ-WB score is below 2.5
	plr between 0% and 1%	The PESQ-WB scores are above 2.5	No clear patterns. Most of the PESQ-WB scores are below 2.5	The PESQ-WB scores are above 2.5
	plr between 1% and 2%	Majority of the PESQ-WB scores are above 2.5 , refer to Figure 5.11 on page 118	No clear patterns. Most of the PESQ-WB scores are below 2.5, refer to Figure 5.12 on page 118	Most of the PESQ-WB scores are above 2.5, refer to Figure 5.13 on page 119
	plr between 2% and 3%	Several of the PESQ-WB scores are below 2.5	No clear patterns. Most of the PESQ-WB scores are below 2.5	Most of the PESQ-WB scores are above 2.5
	Jitter	no clear patterns, refer to the box and whisker plots on page 122. The scores spread between the good and bad quality.	no clear patterns, refer to the box and whisker plots on page 124. The PESQ-WB scores are mostly low but the scores spread is smaller than Express Talk. There are more outliers on the sample data. The scores skew to the left, which means most of the scores are lower than 2.5	more stable than Skype or Express Talk, refer to the box and whisker plots on page 126. However, the scores are lower than Express Talk and better than Skype. The scores spread is the smallest.
CVSS Score (i.e. between 0 and 10, lower is better)	IPIP	5.4	5.5	5.5
	IPSec	1.8	2.8	2.8
	TLS	1.8	2.8	2.8
Bandwidth Utilisation	Throughput	Throughtputs are almost constant. Maximum throughputs are the same in all conditions. Most probably CBR type of codecs win the negotiation.	Throughput size depends on the codec that wins the negotiation. IPSec and TLS did not increase the throughput	TLS and IPSec did not increase the throughputs. The throughput values depend on the codec that wins the negotiation.
	Codec	G7.11.a, G.7.11.u and GSM with echo cancellation and noise reduction	iLBC, Speex, G.729, G.711, SILK and SVOPC	iLBC, Speex, G.722 and G7.11.

5.5 Summary

This chapter discusses the experimental results and analysis on the results. The chapter is divided into three different sections. The first section discuss the results and analysis on voice quality based on PESQ-WB scores for Express Talk, Skype and Google Talk. The second section looks into the security aspect of the performance and the third section evaluate the throughput for all test cases that were discussed earlier. Next, Chapter 6 explains on how the results gained from this project could be used by the end users.

Chapter 6

Knowledge Presentation And Distribution

'So indeed with hardship is ease. Indeed with hardship is ease.."

Al-Quran:Chapter 94 verses 5-6

This chapter discusses on how the information gained from this project could help end users to choose VoIP services that suit their needs. The information obtained should be disseminated to users in a form that is easily understood by them. In most cases these two scoring systems are being presented as two separate entities. However, it would be more meaningful to present the two scores together. In which case, users can easily comprehend the meaning of the scores.

6.1 Information Convergence

We present our data in a graphical form of CVSS versus PESQ-WB scores. We set some limits to the CVSS and PESQ-WB scores before drawing this graph . Basically a low CVSS value (i.e. value nearer to 0) indicates less vulnerability in the network as compared to the high value (i.e. value nearer to 10) of CVSS. A high PESQ-WB score (i.e. value nearer to 5) indicates better voice quality . We present an algorithm to combine the two sets of scores as in the Subsection 6.1.1. The overall performance of any VoIP services could be determined by these two scoring schemes.

6.1.1 Algorithm

The main purpose of our algorithm is to combine the two metrics that measure VoIP clients' QoS and security so that a user on the work could determine the suitability of a VoIP platform for a particular VoIP client. In order to achieve this objective, we design our solution using Set theory. We start with the descriptions of our independent and dependent variables. There are three important independent variables. They are the security metric, the QoS metric and the

client type. In addition, there is one dependent variable, i.e. the overall performance. From the three independent variables we could determine the status of the dependent variable, whether it is in good or not satisfactory or bad condition. In which case our algorithm is as follows:

Suppose that $x \in C$ and $y \in M$,
where $C = [0..10]$, $M = [0..5]$ and performance β ={ Good, Not Satisfactory, Bad }
then $\forall z$ whereby $z \in V = \{Skype, GoogleTalk, ExpressTalk\}$
then the performance β of z is good if $x \leq 5$ and $y \geq 2.5$ (i.e. GREEN ZONE)
else if $x \geq 5$ and $y \geq 2.5$ then the performance β of z is Not Satisfactory (i.e. BLUE ZONE)
else if $x \leq 5$ and $y \leq 2.5$ the performance β of z is Not Satisfactory (i.e. YELLOW ZONE)
else if $x \geq 5$ and $y \leq 2.5$ the performance β of z is Bad (i.e. RED ZONE)

In this case, C represents the CVSS scores and M represents the PESQ-WB scores. Based on this algorithm we map each z performance, β, into CVSS versus PESQ-WB scores chart. The charts are depicted in Figure 6.1, Figure 6.2, Figure 6.3, Figure 6.4 and Figure 6.5. With regards to the voice QoS, a score $2.5 < M < 3.5$ could only be considered satisfactory, but in our case we consider anything where $M > 2.5$ as "good". In practice an operator may set the threshold for M as nearer 3.5. Our results show lower MOS scores than we might expect, possibly due to a PESQ mapping that has not been calibrated. However, this improved mapping would only be possible with a large subjective test data corpus that is not openly available.

6.1.1.1 Four Coloured Zones

We have divided the overall client statuses into four different zones. The Green zone means that z has good CVSS score and good voice quality. The Red zone means that z is in a danger area whereby both CVSS and PESQ-WB scores are bad and not acceptable. These are the two extreme zones. VoIP users should choose z that is in the Green zone and not otherwise. The Blue zone means that z has good voice quality but it is vulnerable to security threats and attacks. The Yellow zone means that the voice quality of z is not acceptable and might contribute to the conversational difficulty though the security vulnerability is less.

6.1.1.2 Network Impairments

The PESQ-WB scores depend on the impairments induced on the test-bed. The factors that influence the network condition are one-way delay, packet loss rate and network security features like IPSec and TLS tunnelling. Jitters exist due to the inconsistent time taken for voice packets to travel to their destination device from their source device. Part of the problem lies on the TCP/IP protocol itself, refer to Subsubsection 2.4.1.1. By default, the IP protocol

offers a best effort connectivity and the TCP transport protocol of the TCP/IP stack offers reliability. However in the effort of providing the reliability, the overall one-way delay might increase which can be detrimental to any time-sensitive applications. From our observation, a voice quality y, is greater or equal to 2.5 when the one-way delay is between 0ms to 150ms $\forall z, z \in \{Skype, GoogleTalk, ExpressTalk\}$. IPSec and TLS reduce y value a little but the performance β improves in x value because the value reduces with the implementation on IPSec or TLS. Our results show some similarity with the research done by Ranganathan [134]. We have summarised his result on page 79. In which case, there is no significant different on the voice quality for a VoIP service in a secure or an unsecure network. Figure 6.1 displays the position of each VoIP client in the CVSS versus PESQ-WB chart for delay between 0ms and 150ms. The delay is within the acceptable delays for voice application as recommended by ITU-T, G.114 standard.

The other observation is for the packet loss rate between 0% and 1%. We believe that Skype, Google Talk and Express Talk have good voice qualities eventhough Skype might potentially fall into the Yellow and Red zones. Section 5.4 describes the overall effects of CVSS and PESQ-WB scores due to the impairments that have been mentioned before. However, the effects of the two scoring methods are presented separately in the section. Among the three VoIP services, Skype is most likely susceptible to a high packet loss rate. This is evidenced by the score of y below 2.5 for plr between 0% to 1%, refer to Figure 6.3.

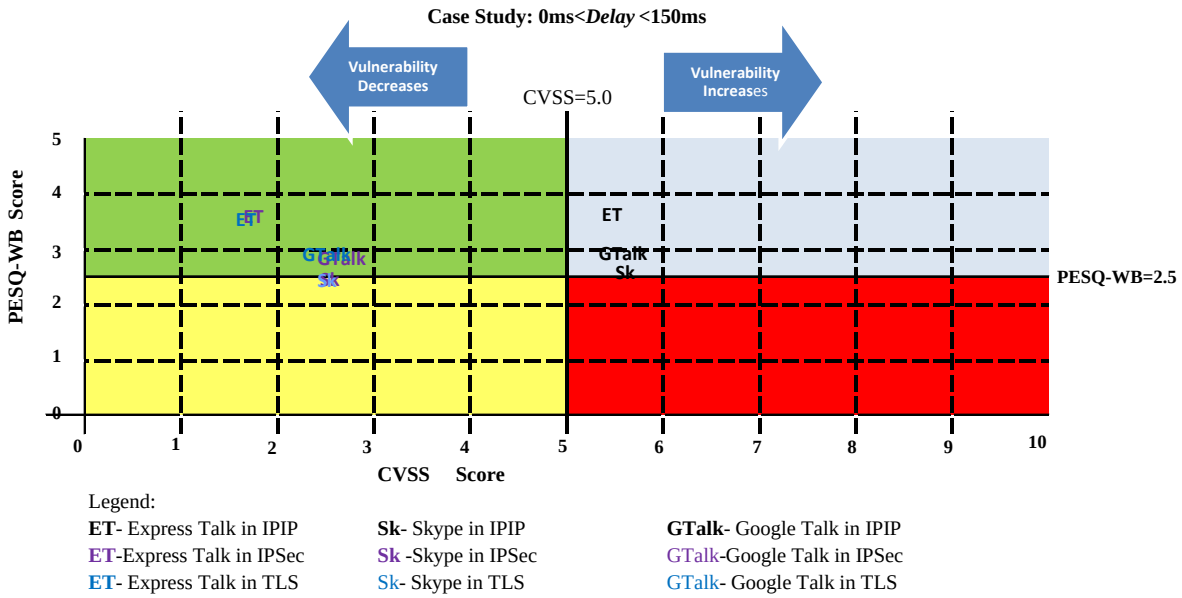


Figure 6.1: CVSS to PESQ-WD chart for Delay between 0ms and 150ms

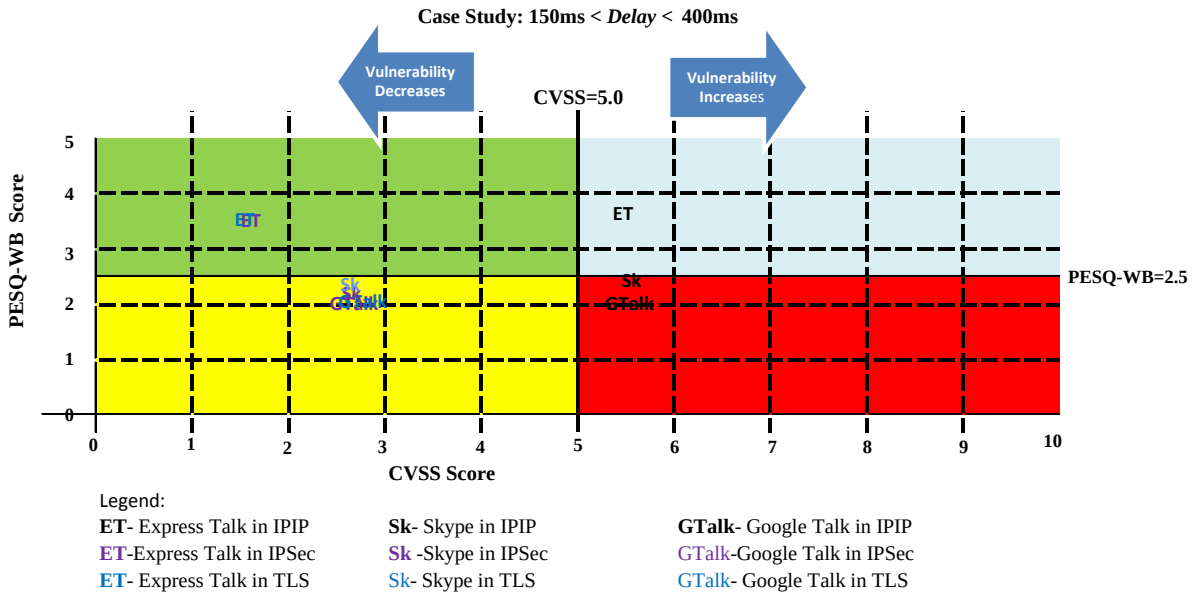


Figure 6.2: CVSS to PESQ-WD chart for Delay between 150ms and 400ms

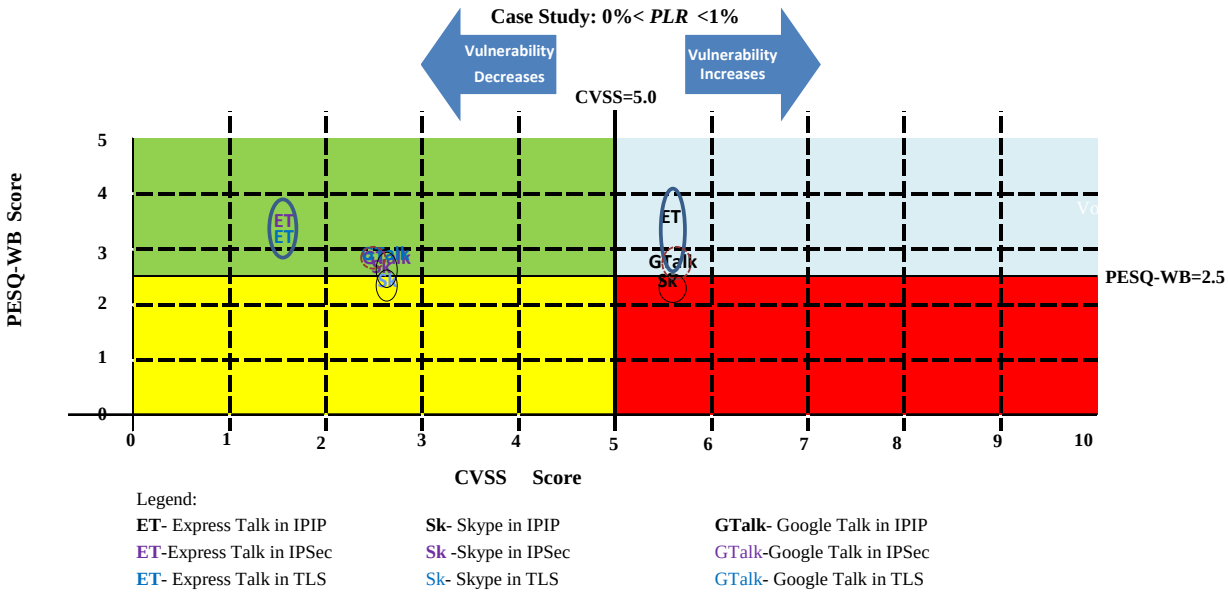


Figure 6.3: CVSS to PESQ-WD chart for PLR between 0% and 1%

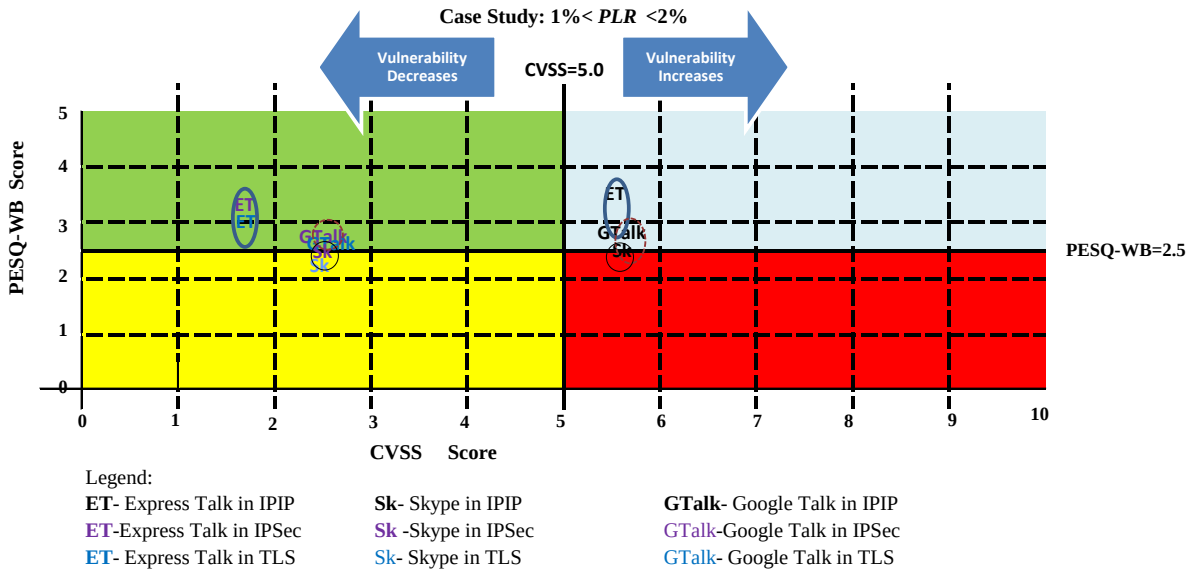


Figure 6.4: CVSS to PESQ-WD chart for PLR betweeen 1% and 2%

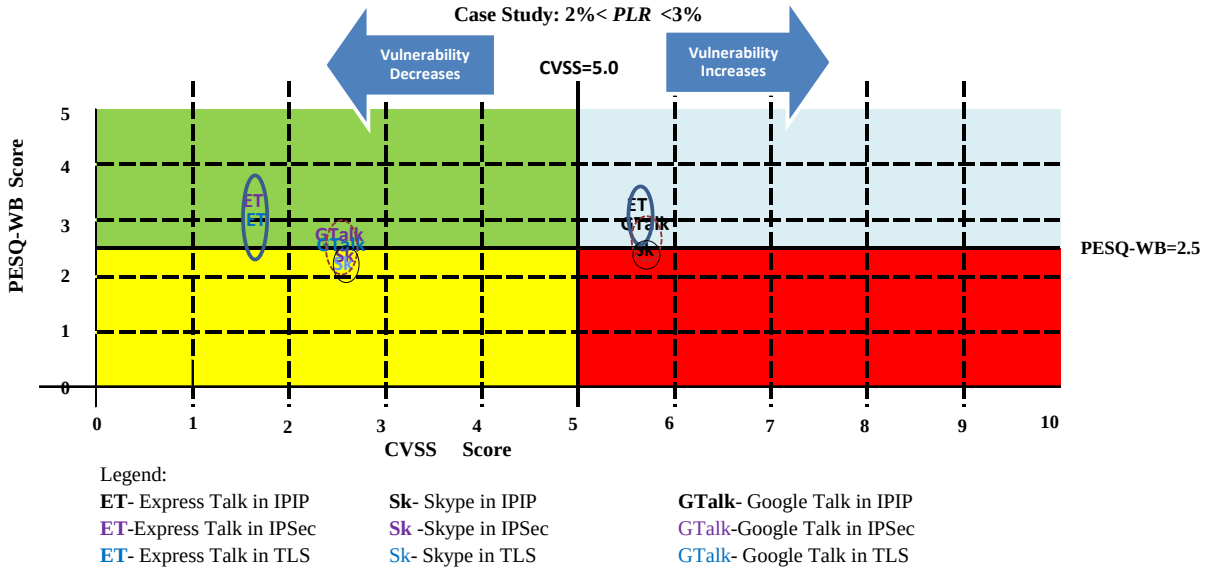


Figure 6.5: CVSS to PESQ-WD chart for PLR betweeen 2% and 3%

6.1.2 CPU Power and Operating System Type

Researches performed by other researchers showed that different types of CPUs speed, RAMs sizes and operating systems can influenced the end and intermediate devices performance. We highlight a few important points as follows:

i. We use quite low capacity of CPUs and RAMs with Windows XP operating systems on our end devices and Linux operating systems on our routers, refer to Table 4.1 on page 93. The preliminary studies conducted by Ahmed and Mansor showed that the number of concurrent calls was mostly dependent on the CPU powers of the end and intermediate devices [2]. However, we only execute a single call at any one time.

ii. An experiment conducted by Gaspary et al. suggested that a CPU was busy at 80% to 97% for messages up to 1024 bytes long due to the fact that, for smaller messages, a larger time fraction was consumed in preparing and dispatching data through the TCP/IP stack, hence, the sending rates in messages tended to be higher. For messages larger than 2048 bytes, the average CPU consumption was lower. The CPU load decreased further when messages larger than 4096 bytes were used [51]. Based on this theory, we can deduced that VoIP clients consume 80% to 97% CPU because their packet sizes are less than 1024 bytes. Table 2.2 shows the VoIP per call bandwidth per codec type. For example, we can see in the table that the total packet size in bytes for G.711 codec is less that 1024 bytes. The total packet size for other codec types is lower.

iii. Another study that was performed by Salah and Hamawi on the impact of running CPU-

bound applications on the performance of IP packet forwarding in Linux and Windows XP showed that Linux IP forwarding was not affected by the CPU-bounded applications, whereas Windows XP network performance was degraded in terms of throughput, packet loss and delay [145]. Balen et al. in their studies showed that a network traffic with smaller packet sizes would benefit from Windows Vista and Windows 7 enhancement to overcome any shortcoming of Windows XP. However, any network traffic with bigger packet sizes still has the best performance in Windows XP [7]. Balen et al. device specifications are quite similar to our end devices except that their RAMs sizes are bigger.

iv. Since the overall one-way delay, refer to Subsection 6.1.3, depends on the end device processor speed and the intermediate network equipments' capabilities hence β might be improved in term of y, the voice quality score, if users can afford higher capability computers' CPUs and RAMs or using an upgraded version of Windows Operating Systems. In general, heavy loaded CPUs and RAMs of end devices and network equipments are not good for any VoIP services.

6.1.3 Source of Delays

Users should understand the source of delays and the detrimental effect of the one-way delay, packet loss rate and jitters to a VoIP voice quality if not properly control. Once they understood the source of these delays, they would consider the right technologies, architectures and topologies to be deployed into their VoIP networks. We highlight a few important points about delays:

i. There are two distint types of delay that contribute to the overall one-way delay. They are the fixed delay and the variable delay. In theory the fixed delay is due to the Coder delay, Packetization and Depacketization delays and Serialization delay [62, 93]. By definition, the Coder delay is the time taken by the digital signal processor (DSP) to compress a block of PCM samples and the value varies with the voice coder used and the device processor speed. The Packetization delay is the time taken to fill a packet payload with encoded or compressed speech and the Depacketization delay is the reversed of it. Both Packetization and Depacketization delays depend on the CPU processing power and the RAM size [51]. The Serialization delay is the fixed delay required to clock a voice or data frame onto a network interface. It is directly related to the clock rate on a trunk. On relatively slow links, such as WAN connections, large data packets can take a long time to send onto the wire. When these large packets are mixed with smaller voice packets, the excessive transmission time can lead to both one-way delay and jitters. One

solution to overcome the problem is to separate the voice and data networks or in a unified network to give higher priority to voice packets.

ii. The other variable delay is due to the queueing delays of the intermediate and end devices. Variable delays or jitters are not good to VoIP services because they cause voice packets to reach the destination device at unpredictable time. The side effect of jitters is packets being drop and loss during transmission. A de-jitter buffer is used to overcome the problem due to the variable delay at the receiving router or gateway. However, the de-jitter buffer increases the overall one-way delay and might affect the voice quality of a call. Subsubsection 2.4.1.2 discusses the idle size for a de-jitter buffer as to reduce the number of packets loss or being drop while maintaining the one-way delay within the acceptable ITU-T standard limit. Another source of delay is the Network switching delay that contribute to the largest delays for any voice connections. The Network switching delay is the most difficult to quantify. It depends of the type of equipments and technology in used by the carrier networks.

6.1.4 Packet Loss Rate

Express Talk uses Constant Bitrate (CBR) type of codec. It uses an extended version of G.711 codec, refer to Section 5.3. In theory G.711u and G.711a do not compress the voice payload. We believe that the y value might be influenced by the codec type. The y values for Express Talk are good in all network conditions.

Google Talk and Skype use Variable Bitrate (VBR) type of codecs. For VBR type of codecs, each packet size depends on the amount of available bandwidth and CPU and memory processing powers of the intermediate and end devices. Google Talk and Skype drop any packets that arrive too late to be placed into a de-jitter buffer. In the experiment we can see that Skype voice quality is slightly below the standard set by ITU-T, G.114 . The acceptable plr for a VoIP service is below 1%. By right a VoIP client should adjust its de-jitter buffer based on the current network condition [179]. The packet loss rate and latency could be reduced by setting the right size of de-jitter buffer depending whether it is a static buffer size or an adaptive buffer size. The de-jitter buffer size is important because a small buffer size would lead to packets overflow and more packets may be discarded and a large buffer size may reduced conversation intelligent because it increases the one-way delay. The de-jitter buffer is discussed in Subsubsection 2.4.1.2. When choosing VoIP services, users should also check on the type of codecs in used and the de-jitter buffer size. In addition they should also find out whether the size is adaptive to the network conditions.

6.1.5 Codec and Bandwidth Utilisation

We have discussed about each VoIP bandwidth utilisation in Section 5.3. The belief that high throughput implies good service quality is not hundred percent correct. Our observation shows that the payload size, hence, the throughput produced by each VoIP service is small as compare to other data applications. High throughputs do not imply high y scores, refer to Subsection 6.1.1. The throughput size is determined by the codec type, the topology and architecture of the infrastructure. The overall frame size is determined by the Packetization and the compression technique applied on the voice payload or encapsulation by other protocols. Our observation on Skype and Google Talk shows that high throughput can happen during call initiation and the services might wait for available resources before initiate another call, refer to Section 5.3. Our experiment shows that the throughput graphs produced by each VoIP client reflect the type of codec used by the client. For example the throughput graphs produced by Express Talk have constant maximum value but the graphs produced by Skype and Google Talk fluntuate with different maximum and minimum throughput values. If this is true then the shape of the throughput graphs give the first indication on the type of codec used by a VoIP client. Most VoIP clients automatically negotiate the best suitable codecs with their peers. However, some VoIP clients allow their users to choose the codecs for their VoIP clients. The information that we provide would allow the users to choose the suitable codecs based on their available resources. For example, users might opt for G.711 codec or the new extended version of it if they have high capacity bandwidth in their network. Our observation on Express Talk indicates that VoIP clients that use G.711 codec would produce better voice quality.

6.2 Summary

This chapter discusses the knowledge that can be deduced from the project that benefit users, system administrators and companies. Next, Chapter 7 is the conclusion of this project and the thesis.

Chapter 7

Conclusion

*"**H**e hath loosed the two seas. They meet.*
***T**here is a barrier between them. They encroach not (one upon the other).*
***W**hich is it, of the favours of your Lord, that ye deny?"*

Al-Quran:Chapter 55 verses 19-21

7.1 Process Flow

In this project, research had been conducted on three selected VoIP clients. They were Express Talk, Skype and Google Talk. Our experiment was conducted in the laboratory. The experiment was executed on a hybrid testbed because the approach was closer to the real world problem. The experiment gathered the information regarding secure VoIP services. Several researchers conducted experiments on SIP-based systems or Skype [27, 8, 14, 23]. However, none of them tried to figure out the differences and similarities among SIP-based application, Skype and Google Talk, particularly on their performances with the effect of the Network Layer security or Transport Layer security implementations even though these services were heavily used by VoIP users. A sytematic approach was followed in order to gain understanding on the subject. The steps that a network administrator and an end user had to undergo to setup secure VoIP services were emulated. The research started with a feasibility study on the subject matter in order to find the possible gap in the area. There were researchers who tried to correlate the VoIP security with VoIP performance, but their research was more confined to theoretical approaches and deduced result from tests conducted in laboratory using simulations. In this research, the hybrid testbed was designed. From this design the method was expanded onto the experimental network. An E-Model calculator was used as the tool to calculate the R-factor in order to gauge the readiness of the testbed. The R-factors were mapped to MOS values and the graphs delay versus MOS and packet loss probability versus

MOS were drawn, example graphs are on page 112. These graphs gave a rough idea on how the experimental results would turn out.

Next, the test rig was built and calibrated. The calibration of the test rig took longer time than we anticipated to cater for three different networks setup, i.e. IPIP, IPSec and OpenVPN TLS. Other time consuming tasks were data collection and analysis. Therefore, data collection was performed with less human intervention. Scripts were written in order to generate and collect the degraded voice calls and sniff traffics that travel from sender to receiver computers in batches. In addition, CACTI was configured to monitor the health of the test equipments and whether they were connected or disconnected. PESQ software was used to evaluate the voice qualities of the selected VoIP services. Several graphs were generated and analysed in order to compare the performance of each VoIP service. The comparison was not confined to voice quality but on the vulnerability score and the amount of throughput generated for each test case. The findings were documented in Chapter 5 of this thesis.

7.2 Potential Research Works

This research had some limitations, hence, the results obtained were subjected to the test cases of this project alone. First, the result for SIP-based service might be different if different types of SIP clients were used in the experiment. Second, the result was valid for the specific Skype and Google Talk versions. Third, the transit router TR* was in the same physical location but different subnet to Router1 and Router2[1]. The research was unable to conclude whether the result would be different if the TR* router was placed at several different locations. Fourth, the research could not determine how much noise was in the data since the experiment was conducted on live testbed and internet offers only best effort connectivity. Location and time might influence the results of this research. Therefore, the scores might be influenced by the time of the day the data was collected especially during peak hours. Fifth, the experiment could only determine which codec was used for each VoIP service based on the time versus throughput graphs, refer to Section 5.3 in Chapter 5 for detail but we think that Express Talk uses CBR type of codec and Google Talk and Skype use VBR type of codec. Sixth, CVSS only predicted the vulnerability score of each VoIP service in the different networks. There was no attempt to hack the system to see how hard it was to compromise the system. Seven, the research only obtained the throughput of each VoIP service for every experiments conducted. Other resources like CPU and memory were not obtained. However, the overall readings of CPU and memory were available in the CACTI report[2]. This data might be obsolete overtime since rrd database was updated every day, therefore the data was not included in the result.

[1] refer to figure 4.1 on page 90
[2] refer to D.1.5 on page lxvii

The research limitations have opened up for some opportunity on the research area. First, the experiment could be extended for other SIP-based clients, another version of Skype and Google Talk or other VoIP clients. Second, to locate TR* at a new location and perform an experiment to validate the previous result. Third is to apply an Application layer security, using an open source application like ZRTP on our topology and architecture. The survey done by Keromytis shows the possibility of using ZRTP to protect and encrypt voice at Application layer for a peer to peer communication [84, 86]. Fourth is to investigate whether the VoIP media was really protected by the IPSec ESP or OpenVPN TLS or ZRTP. Fifth is to investigate the effect of VoIP calls to the network equipments, CPU and memory. Subsequent researchers might also want to duplicate the test rig to confirm the results of this research. The research method could also be used for performance measurement of wireless network, mobile ad-hoc network and other interactive and non-interactive real-time applications.

7.3 Different Technologies under Test

We have included several technologies into our test cases concerning VoIP clients, VoIP protocols, codec types, securities and quality of services. We have tested on three different VoIP clients that are built on different architectures, i.e. Skype is based on proprietary protocol, Google Talk is based on XMPP protocol and Express Talk is based on SIP protocol. They adopt different VoIP signalling protocols. The applications also have different built-in securities. For example, Skype encrypts any voice message travels from sender to receiver but the current Google Talk and Express Talk do not. Google Talk, however, has built-in SASL (i.e. a framework for authentication and data security in Internet protocols) to provide mutual authentication and integrity-checking and Express Talk has embedded DoS prevention, user-to-user authentication and user-to-proxy authentication security features. On top of that, we have tested on the effect of TLS and IPSec on the voice quality. We have discussed the different CVSS scores due to these differences in Section 5.2 on page 128. We also look at several different codec types. Most of the codecs compress the voice packet at one end of the transmission channel to conserve the available bandwidth and then decompress the packet at the other end. This is because the bandwidth is shared with other applications on the Internet. However, the voice quality degraded due to the extra processing that contributes to the overall end-to-end delay. Our experiments show that VoIP clients would select which codecs to be used based on the condition of the network at the time of the transmission. For example, Skype would adjust on the codec type depending on the network traffic. Skype would use SVOPC, a lossy speech compression codec when the communication channels suffer from packet loss. We have discussed this issue in Section 5.3 on page 131. There are also open source codecs like Speex or iLBC which are categorised as variable bit rate codecs used by Google Talk and

Skype in a lossy network condition. G.711 is an example of the codecs under ITU-T standard. G.711 does not compress the payload hence requires more bandwidth to sustain the succession of any audio signal. Express Talk uses the extended version of G.711. However, ITU-T also have codecs that compress audio signal like G.729a, G.723 and G.726, refer to figure 3.2 on page 81. A few of these codecs are supported by the VoIP clients. We used PESQ software to measure the voice quality of each VoIP client. We can conclude that the quality of these services are not the same. It depends on the available bandwidth, codec type, security features and the network traffic conditions. We discussed the effect of one-way delay, jitter and packet loss to the voice quality in section 5.1 on page 113. In most cases, as long as the one-way delay or packet loss is within the recommended ITU-T standard range, i.e. one-way delay less than or equal to 150ms for wired network and packet loss rate less than 1%, then users would not noticed any conversation difficulty. However, there was no standard range for jitter. We have designed our own jitter profiles and analysed their effects on the VoIP clients. The result is tabled in Subsection 5.1.3 on page 121. If time-sensitive applications like VoIP clients ride on UDP instead of TCP then there would be no error checking and no retransmission for any packet losses during transmission. Generally, Skype, Google Talk and Express Talk would transmit the voice packets when there are adequate network resources and drop the packet when there is a network contention.

7.4 Summary

This chapter concludes the research and this thesis. It summaries the processes involved before, during and after the experiment has been conducted. It also highlights the research limitations and the potential related researches that could come out from this research. In addition we also highlight the different technologies under test. In conclusion, the technique used in this research could also be used for other performance measurement researches especially those related to network systems that provide best effort services like wireless network, mobile network and ad-hoc network. Those systems carry time-sensitive interactive and non-interactive applications.

References

[1] H. Abdelnur, V. Cridlig, R. State, and O. Festor. VoIP Security Assessment : Methods and Tools. In *Proceedings of the 1st IEEE workshop on VoIP Management and Security*, volume VoIP Mase, pages 29–34. IEEE, April 2006.

[2] Mohiuddin Ahmed and Abdul Malik Mansor. CPU dimensioning on performance of Asterisk VoIP PBX. In *Proceedings of the 11th communications and networking simulation symposium*, CNS '08, pages 139–146, New York, NY, USA, 2008.

[3] S. Ahson and M. Ilyas. *VoIP Handbook: Applications, Technologies, Reliability, and Security*. Boca Raton: CRC Press, 2009.

[4] M. Aida, N. Miyoshi, and K. Ishibashi. A scalable and lightweight QoS monitoring technique combining passive and active approaches. In *Proc. INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, volume 1, pages 125–133 vol.1, 2003.

[5] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581, IETF, April 1999.

[6] F. Andreasen and B. Foster. Media Gateway Control Protocol Version 1.0. RFC 3435, IETF, January 2003.

[7] J. Balen, G. Martinovic, and Z. Hocenski. Network performance evaluation of latest windows operating . In *Software, Telecommunications and Computer Networks (SoftCOM), 2012 20th International Conference on*, pages 1–6, 2012.

[8] R. Barbieri, D. Bruschi, and E. Rosti. Voice over IPsec: analysis and solutions. In *Proc. 18th Annual Computer Security Applications Conference*, pages 261–270, 2002.

[9] Patrick Battistello, Joaquin Garcia-Alfaro, and Cyril Delétré. Transaction-based authentication and key agreement protocol for inter-domain VoIP. *Journal of Network and Computer Applications*, 35(5):1579–1597, 2012.

[10] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. The Secure Real-time Transport Protocol (SRTP). RFC 3711, IETF, March 2004.

[11] J. G. Beerends, A. P. Hekstra, A. W. Rix, and M. P. Hollier. Perceptual Evaluation of Speech Quality (PESQ) The New ITU Standard for End-to-End Speech Quality Assessment Part II: Psychoacoustic Model. *J.AES (Audio Engineering Society)*, 50(10):765–778, October 2002.

[12] Y. Boger. Fine-tuning Voice over Packet services. Whitepaper, RADCOM Ltd, 2008.

[13] D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca, and D. Rossi. Tracking Down Skype Traffic. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 261–265, 2008.

[14] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli. Revealing Skype traffic: when randomness plays with you. *SIGCOMM Comput. Commun. Rev.*, 37(4):37–48, 2007.

[15] A. Bremler-Barr, R. Halachmi-Bekel, and J. Kangasharju. Unregister Attacks in SIP. In *Proc. 2nd IEEE Workshop on Secure Network Protocols*, pages 32–37, 2006.

[16] E. Brosh, S.A. Baset, V. Misra, D. Rubenstein, and H. Schulzrinne. The delay-friendliness of TCP for real-time traffic. *Networking, IEEE/ACM Transactions on*, 18(5):1478–1491, 2010.

[17] J.F.M. Bross and C. Meinel. Can VoIP Live up to the QoS Standards of Traditional Wireline Telephony? In *Telecommunications, 2008. AICT '08. Fourth Advanced International Conference on*, pages 126 –132, June 2008.

[18] D. Butcher, X. Li, and J. Guo. Security Challenge and Defense in VoIP Infrastructures. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions*, 37(6):1152–1162, 2007.

[19] E. Byres and J. Lowe. The Myths and Facts behind Cyber Security Risks for Industrial Control System. Technical report, PA Consulting Group, Tech. Rep., 2004.

[20] Cacti. Cacti Manual 0.8.7. http://docs.cacti.net/manual:087, Last visited December 2010.

[21] Cacti. What is Cacti? http://www.cacti.net/what_is_cacti.php, Last visited March 2010.

[22] Cacti. Cacti the complete rrdtool-based graphing solution. http://www.cacti.net/, Last visited September 2008.

[23] F. Cao and S. Malik. Security analysis and solutions for deploying IP telephony in the critical infrastructure. In *Proc. Workshop of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks*, pages 171–180, 2005.

[24] L. Carvalho, E. Mota, R. Aguiar, A. F. Lima, and J. N. de Souza. An E-model implementation for speech quality evaluation in VoIP systems. In *Proc. 10th IEEE Symposium on Computers and Communications ISCC 2005*, pages 933–938, 2005.

[25] E. Cha, H. Choi, and S. Cho. Evaluation of Security Protocols for the Session Initiation Protocol. In *Proc. 16th International Conference on Computer Communications and Networks ICCCN 2007*, pages 611–616, 13–16 Aug. 2007.

[26] E.Y. Chen. Detecting DoS attacks on SIP systems. In *VoIP Management and Security, 2006. 1st IEEE Workshop on*, pages 53 – 58, April 2006.

[27] K. Chen, C. Huang, and C. Lei. Quantifying Skype User Satisfaction. In *SIGCOMM'06*, Pisa,Italy, September 2006.

[28] Wen-Hui Chiang, Wei-Cheng Xiao, and Cheng-Fu Chou. A Performance Study of VoIP Applications: MSN vs. Skype. In *IEEE ICC MultiComm wokshop First Multimedia Communications Workshop*, Jun 2006.

[29] Soo-Hyun. Choi and M. Handley. Designing TCP-Friendly Window-based Congestion Control for Real-time Multimedia Applications. In *In Inproceedings of PFLDNeT*, 2009.

[30] W. Chou. Strategies to Keep Your VoIP Network Secure. *IT Professional*, 9(5):42–46, September-October 2007.

[31] Cisco. Voice Quality: Voice Over IP - Per Call Bandwidth Consumption. http://www.cisco.com/en/US/tech/tk652/tk698/technologies_tech_note09186a0080094ae2.shtml, Last visited October 2012.

[32] Reuven Cohen and Liran Katzir. The Effect of Packetization Time and Silence Suppression on the Schedulability of Voice Packets in a Shared Medium Wireless Access Network. CiteseerX, 2004.

[33] R. Colda, T. Palade, I. Vermešan, A. Moldovan, and E. Puşchiţă. Link adaptation in mobile WiMAX systems under the ITU-R mix of channels. In *Proceedings of the 14th WSEAS international conference on Communications*, ICCOM'10, pages 242–247, Stevens Point, Wisconsin, USA, 2010.

[34] William Conner and Klara Nahrstedt. Protecting sip proxy servers from ringing-based denial-of-service attacks. In *Multimedia, 2008. ISM 2008. Tenth IEEE International Symposium on*, pages 340–347. IEEE, 2008.

[35] F. Cuervo, N. Greene, A. Rayhan, C. Huitema, B. Rosen, and J. Segers. Megaco Protocol Version 1.0. RFC 3015, IETF, November 2000.

[36] I. Dalgic and H. Fang. Comparison of H.323 and SIP for IP telephony signaling. In *Proc. of Photonics East, (Boston, Massachusetts), SPIE*, September 1999.

[37] M. Day, S. Aggarwal, G. Mohr, and Vincent J. Instant Messaging / Presence Protocol Requirements. Xmpp, IETF, February 2000.

[38] A. F. M. de Lima, L. S. G. de Carvalho, J. N. de Souza, and E. de Souza Mota. A framework for network quality monitoring in the VoIP environment. *International Journal of Network Management*, 17(4):263–274, 2006.

[39] F. De Rango, M. Tropea, P .Fazio, and S. Marano. Overview on VoIP: Subjective and Objective Measurement Methods. *International Journal of Computer Science and Network Security*, 6(1B):140–153, January 2006.

[40] K-O. Detken. VoIP Security regarding the Open Source Software Asterisk. In *International Multi-Conference on Engineering and Technological Innovation (IMETI); 30. Juni bis 02. Juli; Orlando (Florida)*, 2008. Orlando (Florida) 2008.

[41] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, IETF, August 2008.

[42] B. Dixon. OpenCVSS source codes. http://www.dueyesterday.net/system/files/openCVSS.py.txt, Last visited June 2010.

[43] D. Dlaka and M. Kapov. VoIP/PSTN Networks Planning with PlanVoIP Application. Professional paper, IFAC, 2006.

[44] J. Dowling, J. Sacha, and S. Haridi. Improving ICE Service Selection in a P2P System using the Gradient Topology. *Self-Adaptive and Self-Organizing Systems, International Conference on*, 0:285–288, 2007.

[45] S. Ehlert and S. Petgang. Analysis and Signature of Skype VoIP Session Traffic. Technical report, Franunhofer FOKUS Technical Report NGNISKYPE-06b, Berlin, Germany, July 2006.

[46] Ekiga. Download Ekiga Binaries or Source Code. http://ekiga.org/download-ekiga-binaries-or-source-code, Last visited March 2010.

[47] D. Endler and M. Collier. *Hacking VoIP Exposed :Voice over IP Security Secrets and Solutions*. Mc Graw Hill, 2007.

[48] S. Farraposo, K. Boudaoud, L. Gallon, and P. Owezarski. Some issues raised by DoS attacks and the TCP/IP suite. Technical report, In SAR 2005, Batz-sur-mer, France, June 2005.

[49] FIRST. Forum of Incident Response and Security Teams : Common Vulnerability Scoring System (CVSS-SIG), Last visited October 2009.

[50] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[51] Luciano Paschoal Gaspary, Marinho P. Barcellos, André Detsch, and Rodolfo S. Antunes. Flexible security in peer-to-peer applications: Enabling new opportunities beyond file sharing. *Comput. Netw.*, 51(17):4797–4815, December 2007.

[52] D. Geneiatakis, G. Kambourakis, T. Dagiuklas, C. Lambrinoudakis, and S. Gritzalis. SIP Security Mechanisms: A state-of-the-art review. In *Proceedings of the Fifth International Network Conference (INC 2005), Samos, Greece, July 2005.*, 2005.

[53] Qipeng Gong and Peter Kabal. Quality-based playout buffering with FEC for conversational VoIP. In Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura, editors, *INTERSPEECH*, pages 2402–2405. ISCA, September 2010.

[54] Qipeng Gong and Peter Kabal. Improved Quality for Conversational VoIP Using Path Diversity. In *INTERSPEECH*, pages 2549–2552, 2011.

[55] B. Goode. Voice over Internet protocol (VoIP). *Proceedings of The IEEE*, 90(9):1495–1517, September 2002.

[56] B. Goode. Voice Over Internet Protocol (VoIP). In *Proceedings of The IEEE*, volume 90. IEEE, September 2002.

[57] Google. Google Talk for Developers. http://code.google.com/apis/talk/libjingle/, Last visited March 2010.

[58] G. Goth. VoIP Security Gets More Visible. In *Computer Society*, volume 1089, pages 8–10. IEEE, November-December 2006.

[59] N. Greene, M. Ramalho, and B. Rosen. Media Gateway Control protocol Requirements. RFC 2805, IETF, April 2000.

[60] C. Groves, M. Pantaleo, T. Anderson, and T. Taylor. Gateway Control Protocol. RFC 3525, IETF, June 2003.

[61] T.A. Hall. Objective speech quality measures for Internet telephony in Voice over IP (VoIP) Technology. In *Proceedings of SPIE*, volume 4522, Denver, CO, USA, 2001.

[62] Michal HALS. Mathematical Representation of VoIP Connection Delay. *Radioengineering*, 16(3):77–85, September 2007.

[63] F. Hammer, P. Reichl, and T. Ziegler. Where packet traces meet speech samples: an instrumental approach to perceptual QoS evaluation of VoIP. In *Proc. Twelfth IEEE International Workshop on Quality of Service IWQOS 2004*, pages 273–280, 2004.

[64] M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC 2327, IETF, 1998.

[65] R. Hao, David. Lee, Rakesh. K. Sinha, and Nancy. D. Griffeth. Integrated system interoperability testing with applications to VoIP. *IEEE/ACM Trans. Netw.*, pages 823–836, 2004.

[66] W.C. Hardy. *VoIP Service Quality: Measuring and Evaluating Packet-Switched Voice*. McGraw-Hill Networking, 2003.

[67] Helpbytes. Yahoo! Connection Problems. http://www.helpbytes.co.uk/ yconnect.php, Last visited September 2008.

[68] Kirk Hermann, Mark Mattei, and Steve Regini. Enterprise Network Assessment for VoIP Call Quality - Is it Adequate? A capstone paper submitted as partial fulfillment of the requirements for the degree of Masters in Interdisplinary Telecommunications at the University of Colorado, Boulder, December 2002.

[69] Yusuke Hiwasaki and Hitoshi Ohmuro. ITU-T G.711.1: extending G.711 to higher-quality wideband speech. *Comm. Mag.*, 47(10):110–116, October 2009.

[70] P.C.K. Hung and M.V. Martin. Security Issues in VoIP Applications. In *Proc. Canadian Conference on Electrical and Computer Engineering CCECE '06*, pages 2361–2364, 2006.

[71] InfoTech. Strategies for IP Telephony Evaluation and Migration: Best Practice Considerations for Deploying IPT in the Enterprise. White Paper, April 2005. An Executive Briefing Paper.

[72] ITU-T. App. II, Objective Quality Measurement of Telephone-band (300-3400 Hz) speech codecs using measuring normalizing blocks(MNB), 1998.

[73] ITU-T. P.862: Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. ITU-T Recommendation P.862, February 2001.

[74] ITU-T. ITU-T Recommendation - One-way Transmission Time. ITU-T G-Series Recommendation, 2003.

[75] ITU-T. Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals. ITU-T Recommendation H.235 August 2005, August 2005.

[76] ITU-T. The E-model, a computational model for use in transmission planning. ITU-T Recommendation G.107 March 2005, March 2005.

[77] ITU-T. Perceptual Objective Listening Quality Assessment (POLQA), 2011.

[78] ITU-T. E-model Tutorial. http://www.itu.int/ITU-T/2005-2008/com12/emodelv1/tut.htm, Last October visited 2010.

[79] Bennett J. AutoIt v3. http://www.autoitscript.com/autoit3/index.shtml, Last visited June 2010.

[80] J. Janssen, D. De Vleeschauwer, and Guido H. Petit. Delay and Distortion Bounds For Packetized Voice Calls of Traditional PSTN Quality. In *Proceedings of the 1st IP-Telephony Workshop (IPTel 2000)*, pages 105–110, Berlin, Germany, 12-13 April 2000.

[81] Wenyu Jiang, K. Koguchi, and H. Schulzrinne. QoS evaluation of VoIP end-points. In *Proc. IEEE International Conference on Communications ICC '03*, volume 3, pages 1917–1921 vol.3, 2003.

[82] A.B. Johnston and D.M. Piscitello. *Understanding Voice Over IP Security*. Artech House INC, 2006.

[83] Tobias Jung, Sylvain Martin, Damien Ernst, and Guy Leduc. SPRT for SPIT: using the sequential probability ratio test for SPAM in VoIP prevention. In *Dependable Networks and Services*, pages 74–85. Springer, 2012.

[84] A. Keromytis. A Survey of Voice over IP Security Research. In Atul Prakash and Indranil Sen Gupta, editors, *Information Systems Security*, volume 5905 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2009.

[85] A.D. Keromytis. Voice-over-IP Security: Research and Practice. *Security Privacy, IEEE*, 8(2):76 –78, March-April 2010.

[86] A.D. Keromytis. *Voice over IP Security: A Comprehensive Survey of Vulnerabilities and Academic Research*. SpringerBriefs in Computer Science. Springer, 2011.

[87] Angelos D. Keromytis. Voice over IP: Risks, Threats and Vulnerabilities. In *In: Proceedings of the Cyber Infrastructure Protection (CIP) Conference*, 2009.

[88] W. Kho, S. A. Baset, and H. Schulzrinne. Skype relay calls: Measurements and experiments. In *Proc. INFOCOM Computer Communications Workshops IEEE Conference on*, pages 1–6, 13–18 April 2008.

[89] ChinChol. Kim, SangChul. Shin, Sang. Yong. Ha, SunYoung. Han, and YoungJae. Kim. End-to-End QoS Monitoring Tool Development and Performance Analysis for NGN. In *APNOMS*, pages 332–341, 2006.

[90] A. Kos, B. Klepec, and S. Tomazic. Techniques for performance improvement of VoIP applications. In *Proc. 11th Mediterranean Electrotechnical Conference MELECON 2002*, pages 250–254, 2002.

[91] D.R. Kuhn, T.J. Walsh, and S. Fries. Security Considerations for Voice Over IP Systems. Special publication 800-58, The National Institute of Standards and Technology (NIST), January 2005.

[92] J.F. Kurose and K.W. Ross. *Computer Networking A Top-Down Approach*. Addison-Wesley, 4th edition edition, 2008.

[93] B. Kyrbashov, I. Baronak, M. Kovacik, and V. Janata. Evaluation and Investigation of the Delay in VoIP Networks. *Radioengineering*, 20(2):540–547, 2011.

[94] A. Lakaniemi, J. Rosti, and V.I. Raisanen. Subjective VoIP speech quality evaluation based on network measurements. In *Proc. IEEE International Conference on Communications ICC 2001*, volume 3, pages 748–752 vol.3, 2001.

[95] L. Lambrinos and P. Kirstein. Integrating Voice over IP Services in IPv4 and IPv6 Networks. In *Proceedings of the International Multi-Conference on Computing in the Global Information Technology, IEEE Computer Society*, pages 54–, Washington, DC, USA, 2007.

[96] Jimmy Lapierre, Roch Lefebvre, Bruno Bessette, Vladimir Malenovsky, and Redwan Salami. Noise Shapping in an ITU-T G.711-Interoperable Embedded CODEC. *16th European Signal Processing Conference (EUSIPCO 2008)*, 2008.

[97] Christopher Leckie. CPU-based DoS attacks against SIP servers. *NOMS 2008 2008 IEEE Network Operations and Management Symposium*, pages 41–48, 2008.

[98] R. Lehtinen. *Computer Security Basics, 2nd Edition*. OReilly, 2006.

[99] C. Leung and Y. Chan. Network Forensic on Encrypted Peer-to-Peer VoIP Traffics and The Detection, Blocking, and Prioritization of Skype Traffics. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2007. WETICE 2007. 16th IEEE International Workshops*, pages 401–408. IEEE, June 2007.

[100] S. Li, J. Liu, and Q. Zhang. Research on H.323 Communications Passing Through NAT/Firewall. In *Proc. 6th International Conference on ITS Telecommunications*, pages 482–485, June 2006.

[101] Jonas Lindblom. A Sinusoidal Voice Over Packet Coder Tailored for the Frame-Erasure Channel. *IEEE Transactions on Speech and Audio Processing*, 13(5-2):787–798, 2005.

[102] Malden Electronic Ltd. Speech Quality Assessment: Background Information For DSLA and MultiDSLA Users with PESQv2.2. Technical report, Malden Electronic Ltd, 2004.

[103] Rudolf H. Mak. A taxonomy of maximally elastic buffers. *Technical Report CS-Report 04-26 Technische Universiteit Eindhoven*, pages 1–37, September 2004.

[104] V. Manral. Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH). Technical report, IETF, April 2007.

[105] A.P. Markopoulou, F.A. Tobagi, and M.J. Karam. Assessing the quality of voice communications over Internet backbones. *IEEE/ACM Transactions on Networking*, 11(5):747–760, October 2003.

[106] K. Mase, Y. Toyama, A. A. Bilhaj, and Y. Suda. QoS management for VoIP networks with edge-to-edge admission control. In *Proc. IEEE Global Telecommunications Conference GLOBECOM '01*, volume 4, pages 2556–2560 vol.4, 2001.

[107] W. Mazurczyk and Z. Kotulski. New Security and control protocol for VoIP based on steganography and digital watermarking. In *SECTIO AI INFORMATICA*, volume V, page 43, 2006.

[108] Ahmed Meddahi and Hossam Afifi. Packet-E-Model: e-model for VoIP quality evaluation. *Comput. Netw.*, 50(15):2659–2675, 2006.

[109] M. A. Mehmood, T. M. Jadoon, and N. M. Sheikh. Assessment of VoIP quality over access networks. In *Proc. First IEEE and IFIP International Conference in Central Asia on Internet*, 2005.

[110] H. Melvin and L. Murphy. Time synchronization for VoIP quality of service. In *Internet Computing*, volume 6, pages 57–63. IEEE, May -June 2002.

[111] Luis MG. tcpdump/libpcap public repository. http://www.tcpdump.org/, Last visited March 2010.

[112] F. Michaut and F. Lepage. Application-oriented network metrology: metrics and active measurement tools. *IEEE Communications Surveys & Tutorials*, 7(2):2–24, 2005.

[113] Kapila Moon, MM Moon, and BB Meshram. Securing VoIP networks via signaling protocol layer. In *Radar, Communication and Computing (ICRCC), 2012 International Conference on*, pages 6–10. IEEE, 2012.

[114] J Muller and Michael Massoth. Defense against direct SPAM over internet telephony by caller pre-validation. In *Telecommunications (AICT), 2010 Sixth Advanced International Conference on*, pages 172–177. IEEE, 2010.

[115] J.K. Muppala, T. Bancherdvanich, and A. Tyagi. VoIP performance on differentiated services enabled network. In *Proc. IEEE International Conference on Networks (ICON 2000)*, pages 419–423, 2000.

[116] muzychenko.net. Virtual Audio Cable. http://software.muzychenko.net/eng/vac.htm, Last visited May 2012.

[117] M. Narbutt and M. Davis. Experimental investigation on VoIP performance and the resource utilization in 802.11b WLANs. In *Proc. 31st IEEE Conference on Local Computer Networks*, pages 397–403, 2006.

[118] Mohamed Nassar, Sylvain Martin, Guy Leduc, and Olivier Festor. Using decision trees for generating adaptive spit signatures. In *Proceedings of the 4th international conference on Security of information and networks*, pages 13–20. ACM, 2011.

[119] Newport. VoIP Bandwidth Calculator. http://kambing.ui.ac.id/onnopurbo/library/library-ref-eng/ref-eng-3/physical/voip/52-VoIP-Bandwidth.pdf, Last visited January 2012.

[120] O. Nhway. An investigation into the effect of security on reliability and voice recognition system in a VoIP network. In *13th International Conference on Advanced Communication Technology (ICACT)*, pages 1293 –1297, 2011.

[121] Tobias Oetikar. About RRDtool. http://oss.oetiker.ch/rrdtool/, Last visited March 2010.

[122] Tobias Oetiker. RRDTOOL Logging and Graphing: rrdgraph. http://oss.oetiker.ch/rrdtool/doc/rrdgraph.en.html, Last visited November 2011.

[123] S. Osborn, R. Sandhu, and Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Trans. Inf. Syst. Secur.*, 3(2):85–106, 2000.

[124] ITU-T Rec. P.861. Objective quality measurement of telephone-band (300 - 3400 Hz) speech codecs. International Telecommunication Union, Geneva, Switzerland, August 1996.

[125] V. Paulsamy and S. Chatterjee. Network convergence and the NAT/Firewall problems. In *Proc. 36th Annual Hawaii International Conference on System Sciences*, page 10, 2003.

[126] E.W.C. Peh, W.K.G. Seah, Y.H. Chew, and Y. Ge. Experimental Study of Voice over IP Services over Broadband Wireless Networks. In *Proc. 22nd International Conference on Advanced Information Networking and Applications AINA 2008*, pages 834–839, 2008.

[127] T. Peng, C. Leckie, and K. Ramamohanarao. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Comput. Surv.*, 39(1):3, 2007.

[128] R. Perlman. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols. 2nd Edition.* Addison-Wesley, 1999.

[129] Yusuf Perwej and Firoj Parwej. Perceptual Evaluation of Playout Buffer Algorithm for Enhancing Perceived Quality Of Voice Transmission Over IP Network. In *International Journal of Mobile Network Communications & Telematics (IJMNCT)*, volume 2, April 2012.

[130] J.M. Pitts, X. Wang, Q. Yang, and J.A. Schormans. Excess-rate queuing theory for M/M/1/RED with application to VoIP QoS. *Electronics Letters*, 42(20):1188–1189, 2006.

[131] M. Zubair Rafique, M. Ali Akbar, and Muddassar Farooq. Evaluating DOS attacks against SIP-based VoIP systems. In *Proceedings of the 28th IEEE conference on Global telecommunications*, GLOBECOM'09, pages 6130–6135, Piscataway, NJ, USA, 2009. IEEE Press.

[132] K. Ramesh and A. Salman. *Cisco IP Telephony: Planning, Design, Implementation, Operation, and Optimization*. Cisco Press, 2005.

[133] M. R. Randazzo, M. Keeney, E. Kowalski, D. Cappelli, and A. Moore. Insider Threat Study: Illicit Cyber Activity in the Banking and Finance Sector. Technical report, Carnegie Mellon Univ., Software Eng. Inst., 2004.

[134] M.K. Ranganathan and L. Kilmartin. Performance analysis of secure session initiation protocol based VoIP networks. *Computer Communications*, 26(6):552–565, 2003.

[135] R. J. B. Reynolds and A. W. Rix. Quality VoIP - An Engineering Challenge. *BT Technology Journal*, 19:23–32, April 2001.

[136] A. Rix. Advances in objective quality assessment of speech over analogue and packet based networks. In *Proc. IEE Colloquium on Data Compression: Methods and Implementations (Ref. No. 1999/150)*, pages 10/1–10/8, 1999.

[137] A. Rix, R. Reynolds, and M. Hollier. Robust perceptual assessment of end-to-end audio quality. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 39–42, 1999.

[138] A. W. Rix, M. P. Hollier, A. P. Hekstra, and J. G. Beerends. Perceptual Evaluation of Speech Quality (PESQ) The New ITU Standard for End-to-End Speech Quality Assessment Part I–Time-Delay Compensation. In *J.AES(Audio Engineering Society)*, volume 50, pages 755–764, October 2002.

[139] A.W. Rix, J.G. Beerends, M.P. Hollier, and A.P. Hekstra. Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, volume 2, pages 749–752 vol.2, 2001.

[140] A.W. Rix, J.G. Beerends, D.-S. Kim, P. Kroon, and O. Ghitza. Objective Assessment of Speech and Audio Quality:Technology and Applications. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(6):1890–1901, 2006.

[141] L. Rizzo. Dummynet: A simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27:31–41, 1997.

[142] Luigi Rizzo. Dummynet. http://info.iet.unipi.it/ luigi/dummynet/, Last visited March 2010.

[143] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, IETF, June 2002.

[144] K. Salah and M. Almashari. An Analytical Tool to Assess Readiness of Existing Networks for Deploying IP Telephony. In *ISCC '06. Proceedings. 11th IEEE Symposium Computers and Communications*, volume iscc, pages 301–305. IEEE, June 2006.

[145] Khaled Salah and Mohamed Hamawi. Impact of CPU-bound Processes on IP Forwarding of Linux and Windows XP. *J. UCS*, pages 3299–3313, 2010.

[146] A.B. Salman and H. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. Technical report, Columbia University, 2004.

[147] S. Salsano, L. Veltri, and D. Papalilo. SIP security issues: the SIP authentication procedure and its processing load. *Network IEEE*, 16(6):38–44, 2002.

[148] C. Sanders. *Practical Packet Analysis: Using Wireshark to solve real-world network problems*. No Starch Press, 2007.

[149] R.F. Sari and P.L.P. Wirya. Performance Analysis of Session Initiation Protocol on Emulation Network using NIST NET. In *Proc. 9th International Conference on Advanced Communication Technology*, volume 1, pages 506–510, 12–14 Feb. 2007.

[150] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, IETF, July 2003.

[151] Hemant Sengar, Haining Wang, Duminda Wijesekera, and Sushil Jajodia. Detecting VoIP Floods Using the Hellinger Distance. *IEEE Transactions on Parallel and Distributed Systems*, 19:794–805, 2008.

[152] Charles Shen, Erich Nahum, Henning Schulzrinne, and Charles Wright. The impact of TLS on SIP server performance. In *Principles, Systems and Applications of IP Telecommunications*, pages 59–70. ACM, 2010.

[153] Skype. Business version of Skype. http://www.skype.com/intl/en-gb/business/download/, Last visited January 2010.

[154] Skype. Current Codecs. http://forum.skype.com/index.php?showtopic=176491, Last visited June 2011.

[155] NCH Software. Express Talk VoIP Softphone. http://www.nch.com.au/talk/index.html, Last visited May 2012.

[156] M. Spencer, B. Capouch, E. Guy, F. Miller, and K. Shumard. IAX2: Inter-Asterisk eXchange Version 2. Internet-draft april 2007, IETF, April 2007.

[157] S. Spinsante, E. Gambi, and E. Bottegoni. Security solutions in VoIP applications: State of the art and impact on quality. In *Proc. IEEE International Symposium on Consumer Electronics ISCE 2008*, pages 1–4, 2008.

[158] Internet World Stats. The Internet Big Picture , World Internet Users and Population Stats. Internet Usage Statistics, Last visited December 2011.

[159] L. Sun and E.C. Ifeachor. Voice quality prediction models and their application in VoIP networks. *IEEE Transactions on Multimedia*, 8(4):809–820, 2006.

[160] Tim Szigeti and Christina Hattingh. *End-to-End QoS Network Design: Quality of Service in LANs, WANs, and VPNs (Networking Technology)*. Cisco Press, 2004.

[161] A. Takahashi, A. Kurashima, and H. Yoshino. Objective Assessment Methodology for Estimating Conversational Quality in VoIP. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(6):1984–1993, 2006.

[162] TMCnet. Solving the Network Crime Through Forensics. http://www.tmcnet.com/voip/0107/itspecial-focus-forensic-voip-analysis-0107.htm, Last visited December 2007.

[163] Ubuntu. Package: bridge-utils. http://packages.ubuntu.com/maverick/bridge-utils, Last visited March 2010.

[164] U. Varshney, A. Snow, M. McGivern, and C. Howard. Voice over IP. *Commun. ACM*, 45(1):89–96, 2002.

[165] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos. Conceptual Modeling for ETL Processes. In *In Proc. 5th ACM Intl. Workshop on Data Warehousing and OLAP (DOLAP)*, page 1421. McLean, Virginia, USA, 2002.

[166] VoIPSA. VoIP Security and Privacy Threat Taxonomy. Public release 1.0, VoIP Security Alliance, October 2005.

[167] VoIPThink. VoIP Telephones. http://www.voipforo.com/en/Telephones/softphones.php, Last Visited July 2012.

[168] VoIPTroubleshooter. VoIP Quality and Bandwidth Calculator. http://www.voip troubleshooter.com/diagnosis/emodel.html, Last visited September 2008.

[169] S. Voran. Estimation of perceived speech quality using measuring normalizing blocks. In *Proc. IEEE Workshop on Speech Coding For Telecommunications Proceeding*, pages 83–84, 1997.

[170] S. Voran. Objective estimation of perceived speech quality. I. Development of the measuring normalizing block technique. *Speech and Audio Processing, IEEE Transactions on*, 7(4):371 –382, July 1999.

[171] S. Voran. Objective estimation of perceived speech quality .II. Evaluation of the measuring normalizing block technique. *IEEE Transactions on Speech and Audio Processing*, 7(4):383–390, 1999.

[172] Miroslav Voznak, NE Mastorakis, V Mladenov, Z Bojkovic, S Kartalopoulos, A Varonides, M Jha, and D Simian. Speech bandwith requirements in IPsec and TLS environment. In *WSEAS International Conference. Proceedings. Recent Advances in Computer Engineering*, number 13. WSEAS, 2009.

[173] VyattaSystem. VPN. Reference Guide VC5 v03, Vyatta.com, February 2009.

[174] T.J. Walsh and D.R. Kuhn. Challenges in securing voice over IP. *IEEE Security & Privacy*, 3(3):44–49, 2005.

[175] A. J. A. Wang, M. Xia, and F. Zhang. Metrics For Information Security Vulnerabilities. In *International Handbook of Academic Research and Teaching Proceedings of Intellectbase International Consortium,*, volume 1, pages 284–294, 2007.

[176] Ning Wang, Kin-Hon Ho, George Pavlou, and Michael P. Howarth. An overview of routing optimization for internet traffic engineering. *IEEE Communications Surveys and Tutorials*, 1:36–56, 2008.

[177] Y. Wang, Z. Lu, and J. Gu. Research on Symmetric NAT Traversal in P2P applications. In *Proc. International Multi-Conference on Computing in the Global Information Technology ICCGI '06*, pages 59–59, Aug. 2006.

[178] Pubudu Eroshan Weerathunga, Jagath Samarabandu, and Tarlochan Sidhu. Implementation of IPSec in substation gateways. In *Information and Automation for Sustainability (ICIAfS), 2012 IEEE 6th International Conference on*, pages 327–331. IEEE, 2012.

[179] Chen-Chi Wu, Kuan-Ta Chen, Chun-Ying Huang, and Chin-Laung Lei. An empirical evaluation of VoIP playout buffer dimensioning in Skype, Google talk, and MSN Messenger. In *Proceedings of the 18th international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '09, pages 97–102, New York, NY, USA, 2009.

[180] H. Xia and J.C. Brustoloni. Hardening Web browsers against man-in-the-middle and eavesdropping attacks. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 489–498, New York, NY, USA, 2005.

[181] J. Xia, Y. Ge, and C.K. Chang. An empirical performance study for validating a performance analysis approach: PSIM. In *Proc. 29th Annual International Computer Software and Applications Conference COMPSAC 2005*, volume 1, pages 307–312 Vol. 2, 2005.

[182] Xiph.org. Speex: A Free Codec For Free Speech. http://www.speex.org/, Last visited October 2011.

[183] Yahoo. Yahoo! Developer Network. http://developer.yahoo.com/, Last visited December 2007.

[184] J. Yan, W. Muhlbauer, and B. Plattner. Analytical Framework for Streaming over TCP. In *TIK REPORT*, number 333, 2010.

[185] Y. Yeryomin, F. Evers, and J. Seitz. Solving the firewall and NAT traversal issues for SIP-based VoIP. In *Proc. International Conference on Telecommunications ICT 2008*, pages 1–6, 16–19 June 2008.

[186] Hongbin Yim, Jinwoo Hyun, Hwankuk Kim, and Jaeil Jung. SIP based service QoS parameters impact analysis according to DoS based attack rate. In *ACS'08: Proceedings of the 8th conference on Applied computer scince*, pages 259–264, Stevens Point, Wisconsin, USA, 2008.

[187] Houda Zarhouni, Ghizlane Orhanou, Said El Hajji, and Redouane Benaini. Traffic Engineering and Optimization Routing for VoIP Traffic in Wireless Mesh Networks. In *Proceedings of the World Congress on Engineering 2012*, volume Vol II, pages 1–5, London, U.K., July 4 - 6 2012.

[188] Zfone. The Zfone Project. http://zfoneproject.com/, Last visited May 2010.

[189] P. Zimmermann, A. Ed. Johnston, and J. Callas. ZRTP: Media Path Key Agreement for Unicast Secure RTP draft-zimmermann-avt-zrtp-22. Internet-draft, IETF, June 2010.

Appendix A

A.1 Network Configuration and Monitoring

A.1.1 Routers' Scripts

A.1.1.1 Source File: R3_config

```
interfaces {
    ethernet eth0 {
        address 158.125.253.17/30
        description Router3-TR
        duplex auto
        hw-id 00:0e:0c:63:bf:2e
        speed auto
    }
    ethernet eth1 {
        address 158.125.253.1/30
        description Router3-Router1
        duplex auto
        hw-id 00:07:e9:96:78:4b
        speed auto
    }
    ethernet eth4 {
        address 158.125.253.9/30
        description Router3-Router2
        duplex auto
        hw-id 00:04:23:bd:ab:e5
        speed auto
    }
    ethernet eth5 {
        address 172.16.1.2/30
        description Router3-HP
        duplex auto
        hw-id 00:04:23:bd:ab:e4
        speed auto
    }
}
protocols {
    ospf {
        area 100 {
```

```
            network 158.125.253.0/30
            network 158.125.253.16/30
            network 158.125.253.8/30
        }
        log-adjacency-changes {
        }
    }
    snmp {
        community com-router3 {
            authorization ro
            network 172.16.1.0/30
            network 158.125.253.0/27
        }
        contact ASaad
        description "Router3-FH community"
        location FH
    }
    static {
        route 0.0.0.0/0 {
            next-hop 172.16.1.1 {
            }
        }
        route 131.231.127.0/24 {
            next-hop 172.16.1.1 {
            }
        }
    }
}
service {
    ssh {
        allow-root false
        port 22
        protocol-version v2
    }
}
system {
    host-name Router3
    login {
        user root {
            authentication {
                encrypted-password $1$wtXXi9t6$pskG.qE07fDz3nIJq.Tg7.
            }
            level admin
        }
        user vyatta {
            authentication {
                encrypted-password $1$gEv7VFem$VltvrVNNbJByp0OO2kCsE/
            }
            level admin
        }
    }
    ntp-server 69.59.150.135
    package {
        auto-sync 1
        repository community {
            components main
```

```
            distribution stable
            password ""
            url http://packages.vyatta.com/vyatta
            username ""
        }
    }
    time-zone GMT
}


/* Warning: Do not remove the following line. */
/* === vyatta-config-version: "ipsec@1:serial@1:vrrp@1:nat@2:dhcp-relay@1:cluster@1:dhcp-server@3:quagga@1
/* Release version: VC4.1.4 */
```

A.1.1.2 Source File: R2_config:IPIP tunnel

```
interfaces {
    ethernet eth0 {
        address 158.125.253.13/30
        description "Router2 to sgment 158.125.253.12/30-PC2"
        duplex auto
        hw-id 00:0e:0c:6d:08:9a
        smp_affinity auto
        speed auto
    }
    ethernet eth1 {
        address 158.125.253.10/30
        description "Router2 to Router3"
        duplex auto
        hw-id 00:0e:0c:6d:07:d7
        smp_affinity auto
        speed auto
    }
    ethernet eth2 {
        address 192.168.1.1/24
        description "To VLAN management"
        duplex auto
        hw-id 00:07:e9:92:b4:a5
        smp_affinity auto
        speed auto
    }
    loopback lo {
        address 10.0.5.10/24
        address 10.0.11.10/24
        description net5-TR-net11-TR-TJ
    }
    tunnel tun0 {
        address 10.0.3.10/24
        description IPIP-TR
        encapsulation ipip
        local-ip 10.0.5.10
        multicast disable
        remote-ip 10.0.1.18
        ttl 255
```

```
    }
    tunnel tun1 {
        address 10.0.7.10/24
        description IPIP-TR-FJ
        disable
        encapsulation ipip
        local-ip 10.0.11.10
        multicast disable
        remote-ip 10.0.9.249
        ttl 255
    }
}
protocols {
    ospf {
        area 100 {
            network 158.125.253.12/30
            network 192.168.1.0/24
            network 10.0.3.0/24
            network 158.125.253.8/30
            network 10.0.7.0/24
        }
        log-adjacency-changes {
        }
    }
    static {
        route 0.0.0.0/0 {
            next-hop 158.125.253.9 {
                distance 120
            }
        }
        route 131.231.127.0/24 {
            next-hop 158.125.253.9 {
                distance 115
            }
        }
    }
}
service {
    snmp {
        community com-router2 {
            authorization ro
            network 158.125.253.0/27
        }
        contact ASaad
        description Rtr2
        location Hzgrave
    }
    ssh {
        port 22
        protocol-version v2
    }
    telnet {
        port 23
    }
}
system {
```

```
    host-name Router2
    login {
        user root {
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
        user vyatta {
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
    }
    package {
        auto-sync 1
        repository community {
            components main
            distribution stable
            password ""
            url http://packages.vyatta.com/vyatta
            username ""
        }
    }
    time-zone GMT
}


/* Warning: Do not remove the following line. */
/* === vyatta-config-version: "cluster@1:config-management@1:conntrack-sync@1:content-inspection
@2:dhcp-relay@1:dhcp-server@4:firewall@4:ipsec@2:nat@3:qos@1:quagga@2:system@4:vrrp@
1:wanloadbalance@2:webgui@1:webproxy@1:zone-policy@1" === */
```

A.1.1.3 Source File: R2_config:IPSec tunnel

```
interfaces {
    ethernet eth0 {
        address 158.125.253.13/30
        description "Router2 to sgment 158.125.253.12/30-PC2"
        duplex auto
        hw-id 00:0e:0c:6d:08:9a
        speed auto
    }
    ethernet eth1 {
        address 158.125.253.10/30
        description "Router2 to Router3"
        duplex auto
        hw-id 00:0e:0c:6d:07:d7
        speed auto
    }
    ethernet eth2 {
        address 192.168.1.1/24
        description "To VLAN management"
```

```
        duplex auto
        hw-id 00:07:e9:92:b4:a5
        speed auto
    }
    loopback lo {
        address 10.0.5.10/24
        address 10.0.11.10/24
        description net5-TR-net11-TR-TJ
    }
    tunnel tun0 {
        address 10.0.3.10/24
        description IPIP-TR
        encapsulation ipip
        local-ip 10.0.5.10
        remote-ip 10.0.1.18
        ttl 255
    }
    tunnel tun1 {
        address 10.0.7.10/24
        description IPIP-TR-FJ
        disable
        encapsulation ipip
        local-ip 10.0.11.10
        remote-ip 10.0.9.249
        ttl 255
    }
}
protocols {
    ospf {
        area 100 {
            network 158.125.253.12/30
            network 192.168.1.0/24
            network 10.0.3.0/24
            network 158.125.253.8/30
            network 10.0.7.0/24
        }
        log-adjacency-changes {
        }
    }
    snmp {
        community com-router2 {
            authorization ro
            network 158.125.253.0/27
        }
        contact pg-asaad
        description "router2 FH community"
        location FH
    }
    static {
        route 0.0.0.0/0 {
            next-hop 158.125.253.9 {
                distance 120
            }
        }
        route 131.231.127.0/24 {
            next-hop 158.125.253.9 {
```

```
                distance 115
            }
        }
    }
}
service {
    ssh {
        allow-root false
        port 22
        protocol-version v2
    }
    telnet {
        allow-root false
        port 23
    }
}
system {
    host-name Router2
    login {
        user root {
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
        user vyatta {
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
    }
    ntp-server 69.59.150.135
    package {
        auto-sync 1
        repository community {
            components main
            distribution stable
            password ""
            url http://packages.vyatta.com/vyatta
            username ""
        }
    }
    time-zone GMT
}
vpn {
    ipsec {
        copy-tos disable
        esp-group ESP-IPIP {
            compression disable
            lifetime 3600
            mode tunnel
            pfs enable
            proposal 1 {
                encryption aes128
                hash sha1
```

```
            }
        }
        ike-group IKE-IPIP {
            aggressive-mode disable
            dead-peer-detection {
                action hold
                interval 15
                timeout 120
            }
            lifetime 28800
            proposal 1 {
                dh-group 5
                encryption aes128
                hash sha1
            }
        }
        ipsec-interfaces {
            interface eth1
        }
        site-to-site {
            peer 158.125.253.2 {
                authentication {
                    mode pre-shared-secret
                    pre-shared-secret Allahuakhbar
                }
                ike-group IKE-IPIP
                local-ip 158.125.253.10
                tunnel 1 {
                    allow-nat-networks disable
                    allow-public-networks disable
                    esp-group ESP-IPIP
                    local-subnet 10.0.5.10/32
                    remote-subnet 10.0.6.2/32
                }
            }
        }
    }
}


/* Warning: Do not remove the following line. */
/* === vyatta-config-version: "dhcp-server@3:wanloadbalance@1:cluster@1:nat@2:quagga@1:vrrp@1:
firewall@3:ipsec@1:dhcp-relay@1:serial@1:webgui@1" === */
/* Release version: VC4.1.4 */
```

A.1.1.4 Source File: R2_config:OpenVPN TLS

```
interfaces {
    ethernet eth0 {
        address 158.125.253.13/30
        description "Router2 to sgment 158.125.253.12/30-PC2"
        duplex auto
        hw-id 00:0e:0c:6d:08:9a
        smp_affinity auto
```

```
        speed auto
    }
    ethernet eth1 {
        address 158.125.253.10/30
        description "Router2 to Router3"
        duplex auto
        hw-id 00:0e:0c:6d:07:d7
        smp_affinity auto
        speed auto
    }
    ethernet eth2 {
        address 192.168.1.1/24
        description "To VLAN management"
        duplex auto
        hw-id 00:07:e9:92:b4:a5
        smp_affinity auto
        speed auto
    }
    loopback lo {
        address 10.0.5.10/24
        address 10.0.11.10/24
        description net5-TR-net11-TR-TJ
    }
    openvpn vtun0 {
        local-address 10.18.1.2
        mode site-to-site
        remote-address 10.18.1.1
        remote-host 158.125.253.2
        tls {
            ca-cert-file /root/keys/ca.crt
            cert-file /root/keys/RRHzgrave.crt
            key-file /root/keys/RRHzgrave.key
            role active
        }
    }
    tunnel tun0 {
        address 10.0.3.10/24
        description IPIP-TR
        encapsulation ipip
        local-ip 10.0.5.10
        multicast disable
        remote-ip 10.0.1.18
        ttl 255
    }
    tunnel tun1 {
        address 10.0.7.10/24
        description IPIP-TR-FJ
        disable
        encapsulation ipip
        local-ip 10.0.11.10
        multicast disable
        remote-ip 10.0.9.249
        ttl 255
    }
}
protocols {
```

```
    ospf {
        area 100 {
            network 158.125.253.12/30
            network 192.168.1.0/24
            network 10.0.3.0/24
            network 158.125.253.8/30
            network 10.0.7.0/24
        }
        log-adjacency-changes {
        }
    }
    static {
        interface-route 158.125.253.12/30 {
            next-hop-interface vtun0 {
            }
        }
        route 0.0.0.0/0 {
            next-hop 158.125.253.9 {
                distance 120
            }
        }
        route 131.231.127.0/24 {
            next-hop 158.125.253.9 {
                distance 115
            }
        }
    }
}
service {
    snmp {
        community com-router2 {
            authorization ro
            network 158.125.253.0/27
        }
        contact ASaad
        description Rtr2
        location Hzgrave
    }
    ssh {
        port 22
        protocol-version v2
    }
    telnet {
        port 23
    }
}
system {
    host-name Router2
    login {
        user root {
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
        user vyatta {
```

```
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
    }
    package {
        auto-sync 1
        repository community {
            components main
            distribution stable
            password ""
            url http://packages.vyatta.com/vyatta
            username ""
        }
    }
    time-zone GMT
}


/* Warning: Do not remove the following line. */
/* === vyatta-config-version: "cluster@1:config-management@1:conntrack-sync@1:content-inspection@2:
dhcp-relay@1:dhcp-server@4:firewall@4:ipsec@2:nat@3:qos@1:quagga@2:system@
4:vrrp@1:wanloadbalance@2:webgui@1:webproxy@1:zone-policy@1" === */
```

A.1.1.5 Source File: R1_config:IPIP Tunnel

```
interfaces {
    ethernet eth0 {
        address 158.125.253.5/30
        description "Router1 to segment 158.125.253.4/30-PC1"
        duplex auto
        hw-id 00:02:b3:bf:6a:d2
        smp_affinity auto
        speed auto
    }
    ethernet eth1 {
        address 158.125.253.2/30
        description "Router1 to Router3"
        duplex auto
        hw-id 00:02:b3:bf:6a:d6
        smp_affinity auto
        speed auto
    }
    ethernet eth2 {
        address 158.125.253.25/29
        description VLAN4
        duplex auto
        hw-id 00:07:e9:92:b4:09
        smp_affinity auto
        speed auto
    }
    loopback lo {
        address 10.0.6.2/24
```

```
        address 10.0.12.2/24
        description net6-TR-net12-TR-TJ
    }
    tunnel tun0 {
        address 10.0.8.2/24
        description IPIP-TR-FJ
        disable
        encapsulation ipip
        local-ip 10.0.12.2
        multicast disable
        remote-ip 10.0.10.249
        ttl 255
    }
    tunnel tun1 {
        address 10.0.4.2/24
        description IPIP-TR
        encapsulation ipip
        local-ip 10.0.6.2
        multicast disable
        remote-ip 10.0.2.18
        ttl 255
    }
}
protocols {
    ospf {
        area 100 {
            network 158.125.253.4/30
            network 158.125.253.24/29
            network 10.0.4.0/24
            network 158.125.253.0/30
            network 10.0.8.0/24
        }
        log-adjacency-changes {
        }
    }
    static {
        route 0.0.0.0/0 {
            next-hop 158.125.253.1 {
                distance 120
            }
        }
        route 131.231.127.0/24 {
            next-hop 158.125.253.1 {
                distance 115
            }
        }
    }
}
service {
    snmp {
        community com-router1 {
            authorization ro
            network 158.125.253.0/27
        }
        contact ASaad
        description Rtr1
```

```
        location Hzgrave
    }
    ssh {
        port 22
        protocol-version v2
    }
    telnet {
        port 23
    }
}
system {
    host-name Router1
    login {
        user root {
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
        user vyatta {
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
    }
    package {
        auto-sync 1
        repository community {
            components main
            distribution stable
            password ""
            url http://packages.vyatta.com/vyatta
            username ""
        }
    }
    time-zone GMT
}


/* Warning: Do not remove the following line. */
/* === vyatta-config-version: "cluster@1:config-management@1:conntrack-sync@1:content-inspection@2:
dhcp-relay@1:dhcp-server@4:firewall@4:ipsec@2:nat@3:qos@1:quagga@2:system@4:
vrrp@1:wanloadbalance@2:webgui@1:webproxy@1:zone-policy@1" === */
```

A.1.1.6 Source File: R1_config:IPSec Tunnel

```
interfaces {
    ethernet eth0 {
        address 158.125.253.5/30
        description "Router1 to segment 158.125.253.4/30-PC1"
        duplex auto
        hw-id 00:02:b3:bf:6a:d2
        speed auto
```

```
    }
    ethernet eth1 {
        address 158.125.253.2/30
        description "Router1 to Router3"
        duplex auto
        hw-id 00:02:b3:bf:6a:d6
        speed auto
    }
    ethernet eth2 {
        address 158.125.253.25/29
        description VLAN4
        duplex auto
        hw-id 00:07:e9:92:b4:09
        speed auto
    }
    loopback lo {
        address 10.0.6.2/24
        address 10.0.12.2/24
        description net6-TR-net12-TR-TJ
    }
    tunnel tun0 {
        address 10.0.8.2/24
        description IPIP-TR-FJ
        disable
        encapsulation ipip
        local-ip 10.0.12.2
        remote-ip 10.0.10.249
        ttl 255
    }
    tunnel tun1 {
        address 10.0.4.2/24
        description IPIP-TR
        encapsulation ipip
        local-ip 10.0.6.2
        remote-ip 10.0.2.18
        ttl 255
    }
}
protocols {
    ospf {
        area 100 {
            network 158.125.253.4/30
            network 158.125.253.24/29
            network 10.0.4.0/24
            network 158.125.253.0/30
            network 10.0.8.0/24
        }
        log-adjacency-changes {
        }
    }
    snmp {
        community com-router1 {
            authorization ro
            network 158.125.253.0/27
        }
        contact pg-asaad
```

```
        description "router1 FH community"
        location FH
    }
    static {
        route 0.0.0.0/0 {
            next-hop 158.125.253.1 {
                distance 120
            }
        }
        route 131.231.127.0/24 {
            next-hop 158.125.253.1 {
                distance 115
            }
        }
    }
}
service {
    ssh {
        allow-root false
        port 22
        protocol-version v2
    }
    telnet {
        allow-root false
        port 23
    }
}
system {
    host-name Router1
    login {
        user root {
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
        user vyatta {
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
    }
    ntp-server 69.59.150.135
    package {
        auto-sync 1
        repository community {
            components main
            distribution stable
            password ""
            url http://packages.vyatta.com/vyatta
            username ""
        }
    }
    time-zone GMT
}
```

```
vpn {
    ipsec {
        copy-tos disable
        esp-group ESP-IPIP {
            compression disable
            lifetime 3600
            mode tunnel
            pfs enable
            proposal 1 {
                encryption aes128
                hash sha1
            }
        }
        ike-group IKE-IPIP {
            aggressive-mode disable
            dead-peer-detection {
                action hold
                interval 30
                timeout 120
            }
            lifetime 28800
            proposal 1 {
                dh-group 5
                encryption aes128
                hash sha1
            }
        }
        ipsec-interfaces {
            interface eth1
        }
        site-to-site {
            peer 158.125.253.10 {
                authentication {
                    mode pre-shared-secret
                    pre-shared-secret Allahuakhbar
                }
                ike-group IKE-IPIP
                local-ip 158.125.253.2
                tunnel 1 {
                    allow-nat-networks disable
                    allow-public-networks disable
                    esp-group ESP-IPIP
                    local-subnet 10.0.6.2/32
                    remote-subnet 10.0.5.2/32
                }
            }
        }
    }
}


/* Warning: Do not remove the following line. */
/* === vyatta-config-version: "cluster@1:nat@2:vrrp@1:serial@1:firewall@3:wanloadbalance@
1:dhcp-relay@1:quagga@1:ipsec@1:webgui@1:dhcp-server@3" === */
/* Release version: VC4.1.4 */
```

A.1.1.7 Source File: R1_config:OpenVPN TLS

```
interfaces {
    ethernet eth0 {
        address 158.125.253.5/30
        description "Router1 to segment 158.125.253.4/30-PC1"
        duplex auto
        hw-id 00:02:b3:bf:6a:d2
        smp_affinity auto
        speed auto
    }
    ethernet eth1 {
        address 158.125.253.2/30
        description "Router1 to Router3"
        duplex auto
        hw-id 00:02:b3:bf:6a:d6
        smp_affinity auto
        speed auto
    }
    ethernet eth2 {
        address 158.125.253.25/29
        description VLAN4
        duplex auto
        hw-id 00:07:e9:92:b4:09
        smp_affinity auto
        speed auto
    }
    loopback lo {
        address 10.0.6.2/24
        address 10.0.12.2/24
        description net6-TR-net12-TR-TJ
    }
    openvpn vtun0 {
        local-address 10.18.1.1
        mode site-to-site
        remote-address 10.18.1.2
        remote-host 158.125.253.10
        tls {
            ca-cert-file /root/keys/ca.crt
            cert-file /root/keys/RHzgrave.crt
            dh-file /root/keys/dh1024.pem
            key-file /root/keys/RHzgrave.key
            role passive
        }
    }
    tunnel tun0 {
        address 10.0.8.2/24
        description IPIP-TR-FJ
        disable
        encapsulation ipip
        local-ip 10.0.12.2
        multicast disable
        remote-ip 10.0.10.249
        ttl 255
    }
    tunnel tun1 {
```

```
        address 10.0.4.2/24
        description IPIP-TR
        encapsulation ipip
        local-ip 10.0.6.2
        multicast disable
        remote-ip 10.0.2.18
        ttl 255
    }
}
protocols {
    ospf {
        area 100 {
            network 158.125.253.4/30
            network 158.125.253.24/29
            network 10.0.4.0/24
            network 158.125.253.0/30
            network 10.0.8.0/24
        }
        log-adjacency-changes {
        }
    }
    static {
        interface-route 158.125.253.4/30 {
            next-hop-interface vtun0 {
            }
        }
        route 0.0.0.0/0 {
            next-hop 158.125.253.1 {
                distance 120
            }
        }
        route 131.231.127.0/24 {
            next-hop 158.125.253.1 {
                distance 115
            }
        }
    }
}
service {
    snmp {
        community com-router1 {
            authorization ro
            network 158.125.253.0/27
        }
        contact ASaad
        description Rtr1
        location Hzgrave
    }
    ssh {
        port 22
        protocol-version v2
    }
    telnet {
        port 23
    }
}
```

```
system {
    host-name Router1
    login {
        user root {
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
        user vyatta {
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
    }
    package {
        auto-sync 1
        repository community {
            components main
            distribution stable
            password ""
            url http://packages.vyatta.com/vyatta
            username ""
        }
    }
    time-zone GMT
}


/* Warning: Do not remove the following line. */
/* === vyatta-config-version: "cluster@1:config-management@1:conntrack-sync@1:content-inspection@2:
dhcp-relay@1:dhcp-server@4:firewall@4:ipsec@2:nat@3:qos@1:quagga@2:system@4:vrrp@
1:wanloadbalance@2:webgui@1:webproxy@1:zone-policy@1" === */
```

A.1.1.8 Source File: TR_config: IPIP Tunnel

```
interfaces {
    ethernet eth0 {
        address 158.125.253.18/30
        description TR-R3
        duplex auto
        hw-id 00:12:3f:6f:17:e5
        smp_affinity auto
        speed auto
    }
    loopback lo {
        address 10.0.1.18/24
        address 10.0.2.18/24
        description net1-R2-net2-R1
    }
    tunnel tun0 {
        address 10.0.3.18/24
        description IPIP-tunnel-Router2
```

```
        encapsulation ipip
        local-ip 10.0.1.18
        multicast disable
        remote-ip 10.0.5.10
        ttl 255
    }
    tunnel tun1 {
        address 10.0.4.18/24
        description IPIP-tunnel-Router1
        encapsulation ipip
        local-ip 10.0.2.18
        multicast disable
        remote-ip 10.0.6.2
        ttl 255
    }
}
protocols {
    ospf {
        area 100 {
            network 10.0.3.0/24
            network 10.0.4.0/24
        }
        log-adjacency-changes {
        }
    }
    static {
        route 0.0.0.0/0 {
            next-hop 158.125.253.17 {
            }
        }
        route 158.125.253.24/29 {
            next-hop 158.125.253.17 {
            }
        }
    }
}
service {
    snmp {
        community TR {
            authorization ro
            network 158.125.253.0/27
        }
        contact ASaad
        description TR
        location Hzgrave
    }
    ssh {
        port 22
        protocol-version v2
    }
    telnet {
        port 23
    }
}
system {
    host-name TR
```

```
    login {
        user root {
            authentication {
                encrypted-password $1$YGew3aZJ$7ZIT/bjI0VErFPG5MUATI.
            }
            level admin
        }
        user vyatta {
            authentication {
                encrypted-password $1$oHkhPnB9$8RLQ/cgVzvK9EKA6EVLX81
            }
            level admin
        }
    }
    package {
        auto-sync 1
        repository community {
            components main
            distribution stable
            password ""
            url http://packages.vyatta.com/vyatta
            username ""
        }
    }
    time-zone GMT
}


/* Warning: Do not remove the following line. */
/* === vyatta-config-version: "cluster@1:config-management@1:conntrack-sync@1:content-inspection@2:
dhcp-relay@1:dhcp-server@4:firewall@4:ipsec@2:nat@3:qos@1:quagga@2:system@4:vrrp@
1:wanloadbalance@2:webgui@1:webproxy@1:zone-policy@1" === */
```

A.1.2 Switch's Script

A.1.2.1 Source File: switch

```
interface range ethernet g(3,16-17,20)
flowcontrol on
exit
vlan database
vlan 2-4
exit
interface range ethernet g(13-16)
switchport access vlan 2
exit
interface range ethernet g(17-20)
switchport access vlan 3
exit
interface range ethernet g(7-10)
switchport access vlan 4
exit
interface vlan 2
```

```
name client1
exit
interface vlan 3
name client2
exit
interface vlan 4
name monitor
exit
voice vlan oui-table add 0001e3 Siemens_AG_phone________
voice vlan oui-table add 00036b Cisco_phone______________
voice vlan oui-table add 00096e Avaya____________________
voice vlan oui-table add 000fe2 H3C_Aolynk_______________
voice vlan oui-table add 0060b9 Philips_and_NEC_AG_phone
voice vlan oui-table add 00d01e Pingtel_phone___________
voice vlan oui-table add 00e075 Polycom/Veritel_phone___
voice vlan oui-table add 00e0bb 3Com_phone_______________
interface ethernet g3
port monitor g13
port monitor g20
exit
iscsi target port 860 address 0.0.0.0
iscsi target port 3260 address 0.0.0.0
interface vlan 1
ip address 192.168.1.2 255.255.255.0
exit
ip default-gateway 192.168.1.1
hostname switch1
aaa authentication enable default line

aaa authentication login default line

line telnet
password 20b6978f602cf1c18b02e923cadf2bf9 encrypted
exit
line ssh
password 2f36bb1b6e4f736466c80b0a580f2a98 encrypted
exit
line console
password 23b27718e46d81b32d2d37cc72db1ad6 encrypted
exit
username admin password 39d0bb87972d85973ebb8918d58f104e level 15 encrypted
snmp-server engineID local 800002a203001ec9932896
snmp-server location FH
snmp-server contact ASaad
snmp-server community Dell_Network_Manager rw view DefaultSuper
clock summer-time recurring eu
clock source sntp
no ip domain-lookup
```

A.1.3 DummyNet's Scripts

A.1.3.1 Bridging and Firewalling

Source File: bridgescript.sh

```
#!/bin/bash
#==================================================================#
#                                                                  #
#                 FILE: bridgescript.sh
#                 USAGE: create bridge for client A and setup ipfw
#                 DESCRIPTION:
#
#
#                 OPTION:
#                 REQUIREMENT:
#                 BUGS:
#                 NOTES:
#                 AUTHOR: Amna Saad
#                 VERSION: 1
#                 CREATED: 3.3.2010
#                 REVISION:
#
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
echo on
echo "setting up bridge"
 ifconfig eth1 0.0.0.0
 ifconfig eth3 0.0.0.0
 brctl addbr mybridge
 brctl addif mybridge eth1
 brctl addif mybridge eth3
 brctl stp mybridge on
 ifconfig mybridge up
 echo "setting up bridge is completed"
 echo "setting ipfw"
 cd /lib/modules/2.6.32-14-generic-pae/kernel/net/netfilter
 ls -al
 insmod /home/pratik/myscript/ipfw3/dummynet2/ipfw_mod.ko
 echo off
```

A.1.3.2 Delay

Source File: ipfw-delayxx.sh

```
#!/bin/bash
#==============================================================#
#
#                 FILE: ipfwrules.sh
#                 USAGE:  ipfwrule - delay xxms
#                 DESCRIPTION:
#
#
#                 OPTION:
#                 REQUIREMENT:
#                 BUGS:
#                 NOTES:
#                 AUTHOR: Amna Saad
#                 VERSION: 1
#                 CREATED: 4.8.2010
#                 REVISION:
#
```

```
################ Start of IPFW rules file #####################
# Flush out the list before we begin.
ipfw -q -f flush
# Set rules command prefix
cmd="ipfw -q add"
##############################################################
# No restrictions on Inside LAN Interface for private network
# Not needed unless you have LAN.
# Change xl0 to your LAN NIC interface name
##############################################################
ipfw pipe 100 config delay xxms
$cmd 200 pipe 100 ip from any to any
##############################################################
# Allow the packet through if it has previous been added to the
# the "dynamic" rules table by a allow keep-state statement.
##############################################################
$cmd 400 check-state
################ End of IPFW rules file #######################
```

A.1.3.3 Jitter

Source File: ipfw_jitter.sh

```
#!/bin/bash
#==============================================================#
#
#                 FILE: ipfwrules.sh
#                 USAGE:  ipfwrule - jitter
#                 DESCRIPTION:
#
#
#                 OPTION:
#                 REQUIREMENT:
#                 BUGS:
#                 NOTES:
#                 AUTHOR: Amna Saad
#                 VERSION: 1
#                 CREATED: 4.3.2010
#                 REVISION:
#
################ Start of IPFW rules file #####################
# Flush out the list before we begin.
ipfw -q -f flush
# Set rules command prefix
cmd="ipfw -q add"
##############################################################
# No restrictions on Inside LAN Interface for private network
# Not needed unless you have LAN.
# Change xl0 to your LAN NIC interface name
##############################################################
ipfw pipe 100 config delay 100ms
ipfw pipe 200 config delay 200ms
ipfw pipe 300 config delay 300ms
$cmd 100 prob 0.330000 pipe 100 ip from any to any
$cmd 200 prob 0.330000 pipe 200 ip from any to any
```

```
$cmd 300 prob 0.330000 pipe 300 ip from any to any
##############################################################
# Allow the packet through if it has previous been added to the
# the "dynamic" rules table by a allow keep-state statement.
##############################################################
$cmd 400 check-state
################ End of IPFW rules file ########################
```

A.1.3.4 Packet Loss Rate

Source File: ipfw-delay0-plr-xx.sh

```
#!/bin/bash
#============================================================#
#                                                            #
#                 FILE: ipfwrules.sh
#                 USAGE:  ipfwrule - delay 0 plr xx
#                 DESCRIPTION:
#
#
#                 OPTION:
#                 REQUIREMENT:
#                 BUGS:
#                 NOTES:
#                 AUTHOR: Amna Saad
#                 VERSION: 1
#                 CREATED: 4.3.2010
#                 REVISION:
#
################ Start of IPFW rules file #####################
# Flush out the list before we begin.
ipfw -q -f flush
# Set rules command prefix
cmd="ipfw -q add"
##############################################################
# No restrictions on Inside LAN Interface for private network
# Not needed unless you have LAN.
# Change xl0 to your LAN NIC interface name
##############################################################
ipfw pipe 100 config delay 0ms plr xx
$cmd 100 pipe 100 ip from any to any
##############################################################
# Allow the packet through if it has previous been added to the
# the "dynamic" rules table by a allow keep-state statement.
##############################################################
$cmd 400 check-state
################ End of IPFW rules file ########################
```

Appendix B

B.1 Semi-automated Scripts

Note:All codes are initiated from Client 1 computer unless stated otherwise.

B.1.1 Common Files

B.1.1.1 Source File: playsox.bat

```
@Echo off
Echo Test AUDIODRIVER
rem Define AUDIODRIVER or AUDIODEV TYPES
set SIGMA="SigmaTel Audio"
set MICIN="Microsoft Sound Mapper - Input"
set MICOUT="Microsoft Sound Mapper - Output"
set LOGITECH="Logitech USB Headset"
set VAC1="Virtual Cable 1"
set VAC2="Virtual Cable 2"
set VAC3="Virtual Cable 3"
set VAC4="Virtual Cable 4"
set VAC5="Virtual Cable 5"
rem Play - sox infile -t waveaudio AUDIODEVout
set sox="C:\sox-14.3.1\sox.exe"
set infile="C:\Documents and Settings\coas8\Desktop\test_data\sample1234.wav"
rem %sox% %infile% -t waveaudio %VAC2% trim 0 10
%sox% %infile% -t waveaudio %VAC3%

Echo Print today's date
date /T
```

B.1.1.2 Source File: recsox.bat

```
@Echo off
Echo Test AUDIODRIVER
rem Define AUDIODRIVER or AUDIODEV TYPES
set SIGMA="SigmaTel Audio"
set MICIN="Microsoft Sound Mapper - Input"
set MICOUT="Microsoft Sound Mapper - Output"
```

```
set LOGITECH="Logitech USB Headset"
set VAC1="Virtual Cable 1"
set VAC2="Virtual Cable 2"
set VAC3="Virtual Cable 3"
set VAC4="Virtual Cable 4"
set VAC5="Virtual Cable 5"
rem rec - sox -t waveaudio AUDIODEVin outfile
set sox="C:\sox-14.3.1\sox.exe"
rem cut off fractional seconds
set t=%time:~0,8%
rem replace colons with dashes
set t=%t::=-%
set Year=%Date:~-4%
set Month=%Date:~-10,2%
set Day=%Date:~-7,2%
set Prefix= %1
set FileName=%Prefix%-%Year%-%Month%-%Day%-%t%.wav
set outfile=%FileName%
Echo %outfile% >> D:\amna\logdir\loglist
%sox% -r 96000 -t waveaudio %SIGMA% %outfile% trim 0 10
Echo Print today's date
date /T
```

B.1.1.3 Source File: flush-ipfwrule-remotely.bat

```
@Echo off
set dummynet=158.125.253.27
ssh root@%dummynet% /root/flushout.sh
```

B.1.1.4 Source File: setipfwrule-remotely.bat

```
@Echo off
Echo start run setipfw-remotely.bat
rem to run setipfw-remotely ipfwconfig
set dummynet=158.125.253.27
set ipfwconfig = %1
ssh root@%dummynet% /root/%ipfwconfig%
```

B.1.1.5 Source File: flushout.sh

This code is run in Dummynet computer.

```
#!/bin/bash
#====================================================#
#                                                    #
#                 FILE: ipfwrules.sh                 #
#                 USAGE:  ipfwrule -flush out        #
#                 DESCRIPTION:                     #
#                             #
#                 OPTION:                          #
#                 REQUIREMENT:                     #
```

```
################ Start of IPFW rules file ###########
# Flush out the list before we begin.
ipfw -q -f flush
ipfw list

##############################################################
# Allow the packet through if it has previous been added to the#
# the "dynamic" rules table by a allow keep-state statement.   #
##############################################################
$cmd 400 check-state
################ End of IPFW rules file ########################
```

B.1.1.6 Source file: runtcpdump.sh

This code is run from TCPdump computer.

```
#!/bin/sh
wait=$1
shift 1
tcpdump $* &
pid=$!
sleep $wait
kill $pid
```

B.1.2 Skype

B.1.2.1 Source File: datacollection.bat

```
@Echo off
set t=%time:~0,8%
rem replace colons with dashes
set t=%t::=-%
set Year=%Date:~-4%
set Day=%Date:~-10,2%
set Month=%Date:~-7,2%
set tcpdump=158.125.253.28
set Capturedfile=Skype-IPIP-%Year%-%Month%-%Day%-%t%.pcap
set file=/home/coas8/datacollection/TR/SKYPEPCAP/%Capturedfile%
Echo %Capturedfile% >> D:\amna\logdir\loglistpcap
Echo start run runtcpdump
START /B ssh root@%tcpdump% /root/runtcpdump.sh 15 -i eth0 -w %file%
Echo start run callclient
C:\myscript\latestscript\SkypeA\skypeClientAattach.pl
ECHO end
```

B.1.2.2 Source File: recandplay.bat

```
ECHO start run recsox.bat
ECHO THEN start run playsox.bat
START /B C:\myscript\latestscript\common\recsox.bat D:\AMNA\SkypeA\outaudio\temp\Skype-IPIP
```

```
call C:\myscript\latestscript\common\playsox.bat
ECHO End
```

B.1.2.3 Source File: skypeClientAattach.pl

```
#!C:\strawberry\perl\bin\perl -w

#All libraries here
use Win32::Skype;
use String;

sub AttachThenCall
{
my $Skype = Win32::Skype->new;

#$Skype->start(1, 0);

$Skype->attach;

while (!$Skype->attach)
{
print "Client A available". $Skype->attach, "\n";
}

print $Skype->userGetFullName;

my $clientB='amna.saad';
#my $clientB='echo123';
while (!$Skype->userStatus($clientB))
         {
 print $Skype->userStatus($clientB),"\n";
 }
$Skype->call($clientB);
while(!$Skype->callStatus)
                       {
print $Skype->userSetCallNoAnswerTimeout(5), "\n";
$Skype->call($clientB);
}
system('"C:\myscript\latestscript\SkypeA\recandplay.bat"');

$Skype->endCall();
}
AttachThenCall;
```

B.1.3 SIP

B.1.3.1 Source File: datacollection.bat

```
@Echo off
set t=%time:~0,8%
rem replace colons with dashes
set t=%t::=-%
set Year=%Date:~-4%
```

```
set Day=%Date:~-10,2%
set Month=%Date:~-7,2%
set tcpdump=158.125.253.28
set Capturedfile=SIP-IPSEC-plr2to3-%Year%-%Month%-%Day%-%t%.pcap
set file=/home/coas8/datacollection/TR/SIPPCAP/PLR2to3/IPSEC/%Capturedfile%
Echo %Capturedfile% >> D:\amna\logdir\loglistpcap
ECHO start run runtcpdump
START /B ssh root@%tcpdump% /root/runtcpdump.sh 15 -i eth0 -w %file%
ECHO start run callclient
call C:\myscript\latestscript\SIPA\SIPRegisterNCall.bat
ECHO end
```

B.1.3.2 Source File: recandplay.bat

```
ECHO start run recsox.bat
ECHO THEN start run playsox.bat
sleep 5
START /B C:\myscript\latestscript\common\recsox.bat D:\AMNA\SIPA\outaudio\temp\ipsec-plr2to3
call C:\myscript\latestscript\common\playsox.bat
ECHO End
```

B.1.3.3 Source File: SIPRegisterNCall.bat

```
ECHO Register and call
START /B  C:\myscript\latestscript\SIPA\clientAcallclientB.exe
call C:\myscript\latestscript\SIPA\recandplay.bat
call C:\myscript\latestscript\SIPA\hangupcall.exe
rem call C:\myscript\latestscript\SIPA\closetray.exe
ECHO End
```

B.1.3.4 Source File: clientAcallclientB.au3

```
; clientAcallclientB

#Region --- Au3Recorder generated code Start ---
Run('C:\Program Files\NCH Software\Talk\talk.exe')
_WinWaitActivate("Express Talk","")
Send("coas8{ENTER}")
_WinWaitActivate("coas8@ekiga.net", "")
Send("{F11}")
#RunWait("C:\myscript\latestscript\SIPA\recandplay.bat")


#Region --- Internal functions Au3Recorder Start ---
Func _WinWaitActivate($title, $text, $timeout = 0)
WinWait($title, $text, $timeout)
If Not WinActive($title, $text) Then WinActivate($title, $text)
WinWaitActive($title, $text, $timeout)
EndFunc   ;==>_WinWaitActivate
#EndRegion --- Internal functions Au3Recorder Start ---
```

B.1.3.5 Source File:hangupcall.au3

```
; hangupcall

#Region --- Au3Recorder generated code Start ---
Send("{ALTDOWN}{F12}")

#Region --- Internal functions Au3Recorder Start ---
Func _WinWaitActivate($title, $text, $timeout = 0)
WinWait($title, $text, $timeout)
If Not WinActive($title, $text) Then WinActivate($title, $text)
WinWaitActive($title, $text, $timeout)
EndFunc   ;==>_WinWaitActivate
#EndRegion --- Internal functions Au3Recorder Start ---

#EndRegion --- Au3Recorder generated code Start ---

#Region --- Au3Recorder generated code Start ---
```

B.1.4 Google Talk

B.1.4.1 Source File: datacollection.bat

```
@Echo off
set t=%time:~0,8%
rem replace colons with dashes
set t=%t::=-%
set Year=%Date:~-4%
set Day=%Date:~-10,2%
set Month=%Date:~-7,2%
set tcpdump=158.125.253.28
set Capturedfile=Gtalk-ZRTP-%Year%-%Month%-%Day%-%t%.pcap
set file=/home/coas8/datacollection/TR/GTALKPCAP/%Capturedfile%
Echo %Capturedfile% >> D:\amna\logdir\loglistpcap
ECHO start run runtcpdump
START /B ssh root@%tcpdump% /root/runtcpdump.sh 15 -i eth0 -w %file%
ECHO start run callclient
call C:\myscript\latestscript\Gtalk\GtalkRegisterNCall.bat
ECHO end
```

B.1.4.2 Source File: recandplay.bat

```
ECHO start run recsox.bat
ECHO THEN start run playsox.bat
START /B C:\myscript\latestscript\common\recsox.bat D:\AMNA\Gtalk\outaudio\temp\Gtalk-ZRTP
call C:\myscript\latestscript\common\playsox.bat
ECHO End
```

B.1.4.3 Source File: clientAcallclientB.au3

```
; clientAcallclientB
```

```
#Region --- Au3Recorder generated code Start ---

Run('googletalk')
_WinWaitActivate("Google Talk", "")
Send("amnasaad20{ENTER}")
_WinWaitActivate("amnasaad20@gmail.com", "")
Send("{F11}")
Sleep(100)
RunWait("C:\myscript\latestscript\Gtalk\recandplay.bat")
Send("{F12}")

_WinWaitActivate("amnasaad20@gmail.com", "")
Send("{CTRLDOWN}{F4}{CTRLUP}")
_WinWaitActivate("Google Talk", "")
Send("{ALTDOWN}{F4}{ALTUP}")


#Region --- Internal functions Au3Recorder Start ---
Func _WinWaitActivate($title, $text, $timeout = 0)
WinWait($title, $text, $timeout)
If Not WinActive($title, $text) Then WinActivate($title, $text)
WinWaitActive($title, $text, $timeout)
EndFunc   ;==>_WinWaitActivate
#EndRegion --- Internal functions Au3Recorder Start ---
```

B.1.4.4 Source File: clientBacceptsClientA.au3

```
;clientBaccepts


#region --- Au3Recorder generated code Start ---

Run('googletalk')
_WinWaitActivate("azizahcoas8\40googlemail.com@gtalk.jabbin.com","Chat Links")
Send("{F11}")
_WinWaitActivate("azizahcoas8\40googlemail.com@gtalk.jabbin.com","",500)
Sleep (10000)
Send("{CTRLDOWN}{F4}{CTRLUP}")
_WinWaitActivate("Google Talk","")
Send("{CTRLDOWN}{F4}{CTRLUP}")



#region --- Internal functions Au3Recorder Start ---
Func _WinWaitActivate($title,$text,$timeout=0)
WinWait($title,$text,$timeout)
If Not WinActive($title,$text) Then WinActivate($title,$text)
WinWaitActive($title,$text,$timeout)
EndFunc
#endregion --- Internal functions Au3Recorder End ---
```

Appendix C

C.1 Software Tools

C.1.1 PESQ

C.1.1.1 Link To PESQ source codes

Note:PESQ codes is available from this website http://www.itu.int/rec/T-REC-P.862-200102-I/en [73].

C.1.1.2 Sample of PESQ run script

Source File: soundtest.bat

```
pesq    +  16000   sample96000.wav sample96000.wav
pesq    +  16000   sample96000.wav outtest3a-failed.wav
pesq    +  16000   sample96000.wav outtest3a-halfway.wav
pesq    +  16000   sample96000.wav outtest3a-halfway2.wav
pesq    +  16000   sample96000.wav outtest3a-toosoon.wav
pesq
```

C.1.1.3 Sample outout of PESQ run

Source File: soundtest.bat

```
REFERENCE      DEGRADED      PESQMOS        PESQMOS        SUBJMOS        COND      SAMPLE_FREQ        CRUDE_DELAY
sample96000.wav   sample96000.wav        SQValue=4.500     4.500     0.000     0     16000    0.0000
 sample96000.wav        outtest3a-failed.wav      SQValue=0.998     0.998     0.000     0     16000    -40.6800
 sample96000.wav        outtest3a-halfway.wav     SQValue=-0.186   -0.186   0.000     0     16000    0.1280
 sample96000.wav        outtest3a-halfway2.wav   SQValue=1.765     1.765     0.000     0     16000    -88.3680
 sample96000.wav        outtest3a-toosoon.wav     SQValue=0.523     0.523     0.000     0     16000    -2.9840
```

C.1.2 Pcap File Processing

C.1.2.1 Sample of Ipsumdump script and rrdtool script

```
STEP 1: Extract <epoch time> and <payload-len> for traffic travel from <src> to <dst>.
The payload-len is in byte.  The payload-len include IP packet length in the dump, not including
```

```
any link-level headers.
ipsumdump -r --no-headers -tL -f " dst 158.125.253.14  && src 158.125.253.6" *.pcap > samplerpt
vi samplerpt
 1307039254.818301 32
 1307039254.818411 32
 1307039254.832169 92
.....
.....
 1307039953.693428 172
 1307039953.693582 172
 1307039953.705148 172
 1307039953.705302 172
 1307039953.725657 172
```

Note: Interpretation of the script as follows:
Extract epoch time and payload-len by filtering only IP traffics travel
from source(158.125.253.6)
to destination(158.125.253.14).

STEP 2: Create rrd database, in this sample: plr2to3.rrd. Data is archived for every 1 second and 10 seconds.
Notice that the Data Source(DS) is of type GAUGE, to store the values that are measured directly as "they are".

```
rrdtool create plr2to3.rrd    \
--start 1307039254 --step 1   \
DS:payload-len:GAUGE:1:U:U   \
RRA:AVERAGE:0.5:1:46410  \
RRA:AVERAGE:0.5:10:46410\
RRA:MAX:0.5:1:46410    \
RRA:MAX:0.5:10:46410  \
RRA:MIN:0.5:1:46410  \
RRA:MIN:0.5:10:46410
```

How to interpret the script:
We created the round robin database called plr2to3.rrd which starts at 1307039254 epoch time. Our database holds one data source (DS) named "payload-len" that represents a GAUGE. This GAUGE is read every 1 seconds. In the same database two round robin archives (RRAs) are kept, one averages the data every time it is read and keeps 46410 samples. The other averages the payload every 10 seconds over 46410 samples.

STEP 3: Update rrd database with <epoch time> and <payload-len>. In this case there are about 46410 lines of data.
Before updating, ensure that there is no duplication in entry.

```
#rrdtool update test3.rrd <epoch time>:<value>
rrdtool update plr2to3.rrd 1307039254.818301:32
rrdtool update plr2to3.rrd 1307039254.818411:32
rrdtool update plr2to3.rrd 1307039254.832169:92
    .....
    .....
rrdtool update plr2to3.rrd 1307039953.693428:172
rrdtool update plr2to3.rrd 1307039953.693582:172
rrdtool update plr2to3.rrd 1307039953.705148:172
rrdtool update plr2to3.rrd 1307039953.705302:172
rrdtool update plr2to3.rrd 1307039953.725657:172
```

```
STEP 4: "Rrdtool fetch" is used to check that the data is update successfully.
rrdtool fetch plr2to3.rrd   AVERAGE --start 1307039254  --end 1307039954

STEP 5: Draw the average throughput, maximum throughput and minimum throughput.

rrdtool graph ipsecplr2to3.png        \
--start 1307039254  --end 1307039954          \
--vertical-label bps                          \
DEF:avepayload=plr2to3.rrd:payload-len:AVERAGE              \
DEF:maxpayload=plr2to3.rrd:payload-len:MAX              \
DEF:minpayload=plr2to3.rrd:payload-len:MIN              \
CDEF:avelen=avepayload,8,\*,0,+                 \
CDEF:maxlen=maxpayload,8,\*,0,+                    \
CDEF:minlen=minpayload,8,\*,0,+                   \
LINE5:avelen#00FF00:"Throughput"          \
LINE3:maxlen#FF0000:"Maximum"  \
LINE1:minlen#0000FF:"Minimum"
```

C.1.2.2 Sample of Ipaggcreate script

```
Note: Ipaggcreate can be used to find the total bytes transfer fro src to dst.
ipaggcreate -r -B -s -f "dst 158.125.253.14  && src 158.125.253.6" *.pcap > report
```

C.1.3 ICMP Ping

C.1.3.1 Sample of ICMP ping command

```
Note: pinging to a destination provides rtt and packet loss rate. In this example a 100 bytes
payload is sent with the "ping".
ping  -l 100 -n 10 158.125.253.14 > ICMPlog
```

C.1.4 E-model

Note:E-model tutorial available from this website http://www.itu.int/ITU-T/2005-2008/com12/emodelv1/tut.htm[78].

Source File: emodel.java

```
import java.io.*;
import java.text.DecimalFormat;

//*******************************//
// credit to Bill Gao             //
// voipcalculator.java            //
// http://www.billgao.net/?p=14   //
//*******************************//

public class emodel {
/**
 * This function returns R value calculated by using the passed in parameters
 * See detail of each parameter in ITU-T G.107
 * @param T
 * @param Ppl
```

```
 * @param SLR
 * @param RLR
 * @param Ds
 * @param STMR
 * @param Dr
 * @param TELR
 * @param WEPL
 * @param Ie
 * @param BPL
 * @param BurstR
 * @param A
 * @param Nc
 * @param Ps
 * @param Pr
 * @param qdu
 * @param Nfor
 * @return
 */
/*****************************************************************************/
public double roundTwoDecimals(double d) {
         DecimalFormat twoDForm = new DecimalFormat("#.##");
return Double.valueOf(twoDForm.format(d));
}

//*****************************************************************************//
//ReadingFloat
 public static double[] readFile(String file, String delimiter)
     throws Exception {
   return(readValues(new java.io.FileInputStream(file), delimiter));
   }

 public static double[] readURL(String url, String delimiter)
     throws Exception {
   java.net.URL addr = new java.net.URL(url);
   return(readValues(addr.openStream(), delimiter));
   }

 public static double[] readValues(java.io.InputStream in, String delimiter)
    throws  java.io.FileNotFoundException,
            java.io.IOException,
            java.lang.NumberFormatException {
   String thisLine;
   java.io.BufferedInputStream s = new java.io.BufferedInputStream(in);
   java.io.BufferedReader myInput = new java.io.BufferedReader
      (new java.io.InputStreamReader(s));
   int j = 0;
   double[] values = new double[144]; //change array size here

   while ((thisLine = myInput.readLine()) != null) {
     // scan it line by line
     java.util.StringTokenizer st =
          new java.util.StringTokenizer(thisLine, delimiter);
     while(st.hasMoreElements())
       values[j++] = Double.valueOf(st.nextToken()).doubleValue();
       }
    return(values);
```

```
    }

//*****************************************************************************//
private static double calRValue(double T, double Ppl, double SLR,
double RLR, double Ds, double STMR, double Dr, double TELR,
double WEPL, double Ie, double BPL, double BurstR, double A,
double Nc, double Ps, double Pr, double qdu, double Nfor) {
double LSTR = STMR + Dr,
Tr = 2*T,
Ta = T;
double Nfo = Nfor + RLR;
double OLR = SLR+RLR;
double Pre = Pr + 10*Math.log10(1+Math.pow(10,
((double)(10-LSTR))/(double)10));
double Nor = RLR - 121 +Pre
+ 0.008*Math.pow((Pre-35),2);
double Nos = Ps - SLR -Ds
- 100 +0.004*Math.pow((Ps-OLR -Ds - 14),2);
double No = 10*Math.log10((Math.pow(10, (double)Nc/(double)10))
+Math.pow(10, (double)Nos/(double)10)
+Math.pow(10, (double)Nor/(double)10)
+Math.pow(10, (double)Nfo/(double)10));
double Ro = 15 - 1.5*(SLR+No);

double Q = 37 - 15*(Math.log10(qdu));
double G = 1.07+0.258*Q+0.0602*Math.pow(Q, 2);
double Z = (double)46/(double)30-G/(double)40;
double Y = (double)(Ro-100)/(double)15+46/8.4-G/9;
double Iq = 15*Math.log10(1+Math.pow(10, Y)
+Math.pow(10, Z));
double STMRo = -10*Math.log10(Math.pow(10, -STMR/(double)10)
+ Math.exp(-T/(double)4)*Math.pow(10, -TELR/10));
double Ist = 12*Math.pow((1+Math.pow((STMRo-13)/(double)6, 8)),
1/(double)8)
-28*Math.pow((1+Math.pow((STMRo+1)/19.4, 35)),
1/(double)35)
-13*Math.pow((1+Math.pow((STMRo-3)/33, 13)),
1/(double)13)
+29;
double Xolr = OLR+0.2*(64+No-RLR);
double Iolr = 20* (Math.pow((1+Math.pow((Xolr/8),(double)8)),
((double)1/(double)8))-Xolr/(double)8);
double Is = Iolr + Ist + Iq;

double TERV = TELR - 40*Math.log10((1+T/10)/(1+T/150)
+6*Math.exp(-0.3*Math.pow(T,2)));
double Idd = 0;
if (Ta>100)
{
double X = Math.log10(Ta/(double)100)/Math.log10(2);
Idd = 25 * (Math.pow((1+Math.pow(X, 6)),
1/(double)6)-3*Math.pow((1+Math.pow(X/(double)3, 6)),
1/(double)6)+2);
}

if (STMR<9)
```

```
TERV = TERV + 0.5*Ist;

double Rle = 10.5 * (WEPL+7)*Math.pow(Tr+1, -0.25);
double Idle = (Ro-Rle)/(double)2
+ Math.sqrt(Math.pow(Ro-Rle, 2)/4+169);
double Roe = -1.5*(No-RLR);
double Re = 80+2.5*(TERV-14);
double Idte = ((Roe-Re)/2
+ Math.sqrt(Math.pow(Roe-Re, 2)/(double)4+100) -1)*(1-Math.exp(-T));
if (STMR > 20)
Idte = Math.sqrt(Math.pow(Idte,2)+Math.pow(Ist,2));
double Id = Idte + Idle + Idd;

double Ieef = Ie+(95-Ie)*(Ppl/((Ppl/BurstR)+BPL));

double R = Ro - Is - Id - Ieef +A;
return R;
}
/**
 * Using default values recommended by ITU-T in G.107
 * @param T  mean one-way delay
 * @param Ppl packet-loss probability
 * @return
 */
public static double calRValue(double T, double Ppl)
{
double SLR = 8,
RLR = 2,
Ds = 3,
STMR = 15,
Dr = 3,

TELR = 65,
WEPL = 110,

Ie = 0,
BPL = 1,
BurstR = 1,
A = 0,
Nc = -70,
Ps = 35,
Pr = 35,
qdu = 1,
Nfor = -64;

return calRValue(T, Ppl, SLR, RLR, Ds, STMR, Dr, TELR, WEPL, Ie, BPL,
BurstR, A, Nc, Ps, Pr, qdu, Nfor);
}

/**
 * This function calculates MOS value from R value
 * @param R
 * @return
 */
public static double calMOSValue(double R)
{
```

```
double mos = 0;
if (R<0)
mos  =1;
else if (R>100)
mos = 4.5;
else
mos = 1 + 0.035*R +R*(R-60)*(100-R)*7*Math.pow(10, -6);
return mos;
}

public static void main(String... Args)
{

new emodel().readdata();

}

/*****************************************************************************/
public void readdata() {
  try {
    // we assume 144 doubles (max)to be read
  //Initialise
  double T=0;
  double Tr=0;
  double Ppl=0;
  double R = 0;
  double MOSValue = 0;

  System.out.println( "Round_trip_time(ms)  Packet_Loss_Prob(%) R-Factor MOS");
  double results [] = readFile("floatwithdelimitercolon.dat", ",");
    for(int i = 0; i < results.length; i=i+2 ) {
      //System.out.println(results[i]);
      Tr=results[i];
   T=Tr/2;
      Ppl=results[i+1];
      R = calRValue(T,Ppl);
      MOSValue = calMOSValue(R);
      System.out.print(Tr    + " ");
      System.out.print(Ppl + " ");
      System.out.print(roundTwoDecimals(R)    + " ");
      System.out.println(roundTwoDecimals(MOSValue));
       }
    System.out.println("One_way_delay(ms)  Packet_Loss_Prob(%) R-Factor MOS");
    results = readFile("floatwithdelimiterspace.dat", " ");
    for(int i = 0; i < results.length; i=i+2 ) {
      //System.out.println(results[i]);
      T=results[i];
      Ppl=results[i+1];
      R = calRValue(T,Ppl);
      MOSValue = calMOSValue(R);
      System.out.print(T    + " ");
      System.out.print(Ppl + " ");
      System.out.print(roundTwoDecimals(R)    + " ");
      System.out.println(roundTwoDecimals(MOSValue));
      }
}
```

```
  catch (Exception e) {
    e.printStackTrace();
    }
}
/*****************************************************************************/
}
```

C.1.5 CVSS

C.1.5.1 Link to CVSS calculator

Note:Two types of CVSS calculators are available from these websites:

i. The first CVSS calculator is from the National Institute of Standards and Technology (NIST), http://nvd.nist.gov/cvss.cfm?version=2&vector=(AV:L/AC:H/Au:N/C:N/I:P/A:C) [49].

ii. The second CVSS calculator was written by Brandon Dixon as openCVSS classes in Python, http://www.dueyesterday.net/system/files/openCVSS.py.txt [42].

Appendix D

D.1 Sample Report

D.1.1 Sample Report of PESQ.exe

```
DEGRADED  PESQMOS  SUBJMOS  COND  SAMPLE_FREQ  CRUDE_DELAY
Skype-IPIP-2011-16-08-15-07-05.wav  2.100  0.000  0  16000 2.5080
 Skype-IPIP-2011-16-08-15-07-17.wav  2.315  0.000  0  16000 2.4720
 Skype-IPIP-2011-16-08-15-07-29.wav  2.180  0.000  0  16000 3.1480
 Skype-IPIP-2011-16-08-15-07-41.wav  2.211  0.000  0  16000 3.5000
 Skype-IPIP-2011-16-08-15-07-53.wav  2.268  0.000  0  16000 3.4280
 Skype-IPIP-2011-16-08-15-08-05.wav  2.187  0.000  0  16000 3.3240
 Skype-IPIP-2011-16-08-15-08-17.wav  2.180  0.000  0  16000 3.3040
 Skype-IPIP-2011-16-08-15-08-28.wav  3.029  0.000  0  16000 1.8320
 Skype-IPIP-2011-16-08-15-08-40.wav  2.242  0.000  0  16000 3.3240
 Skype-IPIP-2011-16-08-15-08-52.wav  3.070  0.000  0  16000 1.4440
  ...

transform to

PARAMETER RUN PROTOCOL SECURITY PESQMOS CRUDE_DELAY PESQ-WB MOS-LQO PESQ-LQ
000ms  1 Skype-IPIP None 2.1 2.508 2.11 1.72 1.42
000ms  2 Skype-IPIP None 2.315 2.472 2.36 1.92 1.70
000ms  3 Skype-IPIP None 2.18 3.148 2.20 1.79 1.52
000ms  4 Skype-IPIP None 2.211 3.5 2.24 1.82 1.56
000ms  5 Skype-IPIP None 2.268 3.428 2.31 1.88 1.64
000ms  6 Skype-IPIP None 2.187 3.324 2.21 1.80 1.53
000ms  7 Skype-IPIP None 2.18 3.304 2.20 1.79 1.52
000ms  8 Skype-IPIP None 3.029 1.832 3.31 2.87 2.79
000ms  9 Skype-IPIP None 2.242 3.324 2.28 1.85 1.61
000ms  10 Skype-IPIP None 3.07 1.444 3.37 2.93 2.85
 ...............................

DEGRADED  PESQMOS  SUBJMOS  COND  SAMPLE_FREQ  CRUDE_DELAY
sip-ipip-dly-0to90-2012-03-05-21-48-46.wav 3.817 0.000 0 16000 8.2
 sip-ipip-dly-0to90-2012-03-05-21-49-19.wav 3.146 0.000 0 16000 5.136
 sip-ipip-dly-0to90-2012-03-05-21-49-41.wav 3.323 0.000 0 16000 5.844
 sip-ipip-dly-0to90-2012-03-05-21-50-14.wav 2.965 0.000 0 16000 6.54
 sip-ipip-dly-0to90-2012-03-05-21-50-47.wav 3.396 0.000 0 16000 1.196
 sip-ipip-dly-0to90-2012-03-05-21-51-20.wav 3.587 0.000 0 16000 4.404
 sip-ipip-dly-0to90-2012-03-05-21-51-42.wav 3.218 0.000 0 16000 7.108
```

```
 sip-ipip-dly-0to90-2012-03-05-21-52-15.wav 3.336 0.000 0 16000 5.348
 sip-ipip-dly-0to90-2012-03-05-21-52-49.wav 3.691 0.000 0 16000 2.416
sip-ipip-dly-0to90-2012-03-05-21-53-17.wav 3.01 0.000 0 16000 2.302

...

transform to
PARAMETER RUN PROTOCOL Security PESQMOS CRUDE_DELAY PESQ-WB MOS-LQO PESQ-LQ
000ms  1 SIP-IPIP IPIP 3.817 8.2 4.20 3.96 3.92
000ms  2 SIP-IPIP IPIP 3.146 5.136 3.47 3.04 2.97
000ms  3 SIP-IPIP IPIP 3.323 5.844 3.69 3.30 3.24
000ms  4 SIP-IPIP IPIP 2.965 6.54 3.23 2.77 2.69
000ms  5 SIP-IPIP IPIP 3.396 1.196 3.78 3.41 3.35
000ms  6 SIP-IPIP IPIP 3.587 4.404 3.99 3.67 3.62
000ms  7 SIP-IPIP IPIP 3.218 7.108 3.56 3.15 3.08
000ms  8 SIP-IPIP IPIP 3.336 5.348 3.70 3.32 3.26
000ms  9 SIP-IPIP IPIP 3.691 2.416 4.09 3.81 3.76
000ms  10 SIP-IPIP IPIP 3.01 2.302 3.29 2.84 2.76

 ...............................

DEGRADED  PESQMOS  SUBJMOS  COND  SAMPLE_FREQ  CRUDE_DELAY
 Gtalk-delay0to90-2011-22-08-16-43-32.wav  2.577  0.000  0  16000 1.0320
 Gtalk-delay0to90-2011-22-08-16-43-46.wav  2.895  0.000  0  16000 0.3320
 Gtalk-delay0to90-2011-22-08-16-44-00.wav  2.796  0.000  0  16000 1.3200
 Gtalk-delay0to90-2011-22-08-16-44-49.wav  2.845  0.000  0  16000 1.4640
 Gtalk-delay0to90-2011-22-08-16-45-03.wav  2.811  0.000  0  16000 1.3800
 Gtalk-delay0to90-2011-22-08-16-45-17.wav  2.815  0.000  0  16000 1.4680
 Gtalk-delay0to90-2011-22-08-16-45-32.wav  2.766  0.000  0  16000 1.3440
 Gtalk-delay0to90-2011-22-08-16-45-46.wav  2.841  0.000  0  16000 1.4400
 Gtalk-delay0to90-2011-22-08-16-46-01.wav  2.802  0.000  0  16000 1.2360
 Gtalk-delay0to90-2011-22-08-16-46-15.wav  2.790  0.000  0  16000 1.5440

  ...

 transform to
PARAMETER RUN PROTOCOL SECURITY PESQMOS CRUDE_DELAY PESQ-WB MOS-LQO PESQ-LQ
000ms  1 Gtalk-IPIP None 2.577 1.032 2.70 2.23 2.09
000ms  2 Gtalk-IPIP None 2.895 0.332 3.13 2.67 2.58
000ms  3 Gtalk-IPIP None 2.796 1.32 3.00 2.53 2.42
000ms  4 Gtalk-IPIP None 2.845 1.464 3.07 2.60 2.50
000ms  5 Gtalk-IPIP None 2.811 1.38 3.02 2.55 2.45
000ms  6 Gtalk-IPIP None 2.815 1.468 3.02 2.55 2.45
000ms  7 Gtalk-IPIP None 2.766 1.344 2.96 2.48 2.38
000ms  8 Gtalk-IPIP None 2.841 1.44 3.06 2.59 2.49
000ms  9 Gtalk-IPIP None 2.802 1.236 3.01 2.53 2.43
000ms  10 Gtalk-IPIP None 2.79 1.544 2.99 2.52 2.41

 ..................................
```

D.1.2 ICMP log

Note: ICMP pings are performed in a control environment without any internetworking or transport layer security on the network.

```
Delay0ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time<1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time<1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time<1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time<1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
Delay20ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=45ms TTL=125
Reply from 158.125.253.14: bytes=100 time=45ms TTL=125
Reply from 158.125.253.14: bytes=100 time=45ms TTL=125
Reply from 158.125.253.14: bytes=100 time=45ms TTL=125
Reply from 158.125.253.14: bytes=100 time=45ms TTL=125
Reply from 158.125.253.14: bytes=100 time=45ms TTL=125
Reply from 158.125.253.14: bytes=100 time=45ms TTL=125
Reply from 158.125.253.14: bytes=100 time=45ms TTL=125
Reply from 158.125.253.14: bytes=100 time=45ms TTL=125
Reply from 158.125.253.14: bytes=100 time=45ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 45ms, Maximum = 45ms, Average = 45ms
Delay40ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=78ms TTL=125
Reply from 158.125.253.14: bytes=100 time=77ms TTL=125
Reply from 158.125.253.14: bytes=100 time=80ms TTL=125
Reply from 158.125.253.14: bytes=100 time=79ms TTL=125
Reply from 158.125.253.14: bytes=100 time=78ms TTL=125
Reply from 158.125.253.14: bytes=100 time=77ms TTL=125
Reply from 158.125.253.14: bytes=100 time=80ms TTL=125
Reply from 158.125.253.14: bytes=100 time=79ms TTL=125
Reply from 158.125.253.14: bytes=100 time=78ms TTL=125
Reply from 158.125.253.14: bytes=100 time=77ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 77ms, Maximum = 80ms, Average = 78ms
Delay60ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=125ms TTL=125
Reply from 158.125.253.14: bytes=100 time=125ms TTL=125
```

```
Reply from 158.125.253.14: bytes=100 time=125ms TTL=125
Reply from 158.125.253.14: bytes=100 time=125ms TTL=125
Reply from 158.125.253.14: bytes=100 time=125ms TTL=125
Reply from 158.125.253.14: bytes=100 time=125ms TTL=125
Reply from 158.125.253.14: bytes=100 time=125ms TTL=125
Reply from 158.125.253.14: bytes=100 time=125ms TTL=125
Reply from 158.125.253.14: bytes=100 time=125ms TTL=125
Reply from 158.125.253.14: bytes=100 time=125ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 125ms, Maximum = 125ms, Average = 125ms
Delay80ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=158ms TTL=125
Reply from 158.125.253.14: bytes=100 time=157ms TTL=125
Reply from 158.125.253.14: bytes=100 time=156ms TTL=125
Reply from 158.125.253.14: bytes=100 time=159ms TTL=125
Reply from 158.125.253.14: bytes=100 time=158ms TTL=125
Reply from 158.125.253.14: bytes=100 time=157ms TTL=125
Reply from 158.125.253.14: bytes=100 time=159ms TTL=125
Reply from 158.125.253.14: bytes=100 time=158ms TTL=125
Reply from 158.125.253.14: bytes=100 time=156ms TTL=125
Reply from 158.125.253.14: bytes=100 time=159ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 156ms, Maximum = 159ms, Average = 157ms
Delay100ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=205ms TTL=125
Reply from 158.125.253.14: bytes=100 time=204ms TTL=125
Reply from 158.125.253.14: bytes=100 time=207ms TTL=125
Reply from 158.125.253.14: bytes=100 time=206ms TTL=125
Reply from 158.125.253.14: bytes=100 time=204ms TTL=125
Reply from 158.125.253.14: bytes=100 time=207ms TTL=125
Reply from 158.125.253.14: bytes=100 time=205ms TTL=125
Reply from 158.125.253.14: bytes=100 time=204ms TTL=125
Reply from 158.125.253.14: bytes=100 time=207ms TTL=125
Reply from 158.125.253.14: bytes=100 time=205ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 204ms, Maximum = 207ms, Average = 205ms
Delay120ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=237ms TTL=125
Reply from 158.125.253.14: bytes=100 time=240ms TTL=125
Reply from 158.125.253.14: bytes=100 time=239ms TTL=125
Reply from 158.125.253.14: bytes=100 time=238ms TTL=125
Reply from 158.125.253.14: bytes=100 time=237ms TTL=125
Reply from 158.125.253.14: bytes=100 time=236ms TTL=125
Reply from 158.125.253.14: bytes=100 time=239ms TTL=125
Reply from 158.125.253.14: bytes=100 time=238ms TTL=125
Reply from 158.125.253.14: bytes=100 time=237ms TTL=125
Reply from 158.125.253.14: bytes=100 time=236ms TTL=125
```

```
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 236ms, Maximum = 240ms, Average = 237ms
Delay140ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=285ms TTL=125
Reply from 158.125.253.14: bytes=100 time=284ms TTL=125
Reply from 158.125.253.14: bytes=100 time=287ms TTL=125
Reply from 158.125.253.14: bytes=100 time=286ms TTL=125
Reply from 158.125.253.14: bytes=100 time=285ms TTL=125
Reply from 158.125.253.14: bytes=100 time=284ms TTL=125
Reply from 158.125.253.14: bytes=100 time=287ms TTL=125
Reply from 158.125.253.14: bytes=100 time=286ms TTL=125
Reply from 158.125.253.14: bytes=100 time=285ms TTL=125
Reply from 158.125.253.14: bytes=100 time=284ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 284ms, Maximum = 287ms, Average = 285ms
Delay160ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=317ms TTL=125
Reply from 158.125.253.14: bytes=100 time=320ms TTL=125
Reply from 158.125.253.14: bytes=100 time=319ms TTL=125
Reply from 158.125.253.14: bytes=100 time=318ms TTL=125
Reply from 158.125.253.14: bytes=100 time=317ms TTL=125
Reply from 158.125.253.14: bytes=100 time=316ms TTL=125
Reply from 158.125.253.14: bytes=100 time=319ms TTL=125
Reply from 158.125.253.14: bytes=100 time=318ms TTL=125
Reply from 158.125.253.14: bytes=100 time=317ms TTL=125
Reply from 158.125.253.14: bytes=100 time=316ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 316ms, Maximum = 320ms, Average = 317ms
Delay180ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=365ms TTL=125
Reply from 158.125.253.14: bytes=100 time=364ms TTL=125
Reply from 158.125.253.14: bytes=100 time=367ms TTL=125
Reply from 158.125.253.14: bytes=100 time=366ms TTL=125
Reply from 158.125.253.14: bytes=100 time=364ms TTL=125
Reply from 158.125.253.14: bytes=100 time=367ms TTL=125
Reply from 158.125.253.14: bytes=100 time=365ms TTL=125
Reply from 158.125.253.14: bytes=100 time=364ms TTL=125
Reply from 158.125.253.14: bytes=100 time=367ms TTL=125
Reply from 158.125.253.14: bytes=100 time=366ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 364ms, Maximum = 367ms, Average = 365ms
Delay200ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=397ms TTL=125
Reply from 158.125.253.14: bytes=100 time=396ms TTL=125
```

```
Reply from 158.125.253.14: bytes=100 time=399ms TTL=125
Reply from 158.125.253.14: bytes=100 time=398ms TTL=125
Reply from 158.125.253.14: bytes=100 time=397ms TTL=125
Reply from 158.125.253.14: bytes=100 time=396ms TTL=125
Reply from 158.125.253.14: bytes=100 time=399ms TTL=125
Reply from 158.125.253.14: bytes=100 time=398ms TTL=125
Reply from 158.125.253.14: bytes=100 time=397ms TTL=125
Reply from 158.125.253.14: bytes=100 time=396ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 396ms, Maximum = 399ms, Average = 397ms
Delay220ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=446ms TTL=125
Reply from 158.125.253.14: bytes=100 time=445ms TTL=125
Reply from 158.125.253.14: bytes=100 time=444ms TTL=125
Reply from 158.125.253.14: bytes=100 time=448ms TTL=124
Reply from 158.125.253.14: bytes=100 time=447ms TTL=124
Reply from 158.125.253.14: bytes=100 time=446ms TTL=124
Reply from 158.125.253.14: bytes=100 time=444ms TTL=124
Reply from 158.125.253.14: bytes=100 time=448ms TTL=124
Reply from 158.125.253.14: bytes=100 time=447ms TTL=124
Reply from 158.125.253.14: bytes=100 time=446ms TTL=124
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 444ms, Maximum = 448ms, Average = 446ms
Delay240ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=477ms TTL=125
Reply from 158.125.253.14: bytes=100 time=476ms TTL=125
Reply from 158.125.253.14: bytes=100 time=480ms TTL=125
Reply from 158.125.253.14: bytes=100 time=479ms TTL=125
Reply from 158.125.253.14: bytes=100 time=478ms TTL=125
Reply from 158.125.253.14: bytes=100 time=477ms TTL=125
Reply from 158.125.253.14: bytes=100 time=476ms TTL=125
Reply from 158.125.253.14: bytes=100 time=480ms TTL=125
Reply from 158.125.253.14: bytes=100 time=479ms TTL=125
Reply from 158.125.253.14: bytes=100 time=479ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 476ms, Maximum = 480ms, Average = 478ms
Delay260ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=528ms TTL=124
Reply from 158.125.253.14: bytes=100 time=527ms TTL=124
Reply from 158.125.253.14: bytes=100 time=525ms TTL=124
Reply from 158.125.253.14: bytes=100 time=525ms TTL=124
Reply from 158.125.253.14: bytes=100 time=525ms TTL=124
Reply from 158.125.253.14: bytes=100 time=527ms TTL=124
Reply from 158.125.253.14: bytes=100 time=526ms TTL=124
Reply from 158.125.253.14: bytes=100 time=524ms TTL=124
Reply from 158.125.253.14: bytes=100 time=528ms TTL=124
Request timed out.
```

```
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 9, Lost = 1 (10\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 524ms, Maximum = 528ms, Average = 526ms
Delay280ms
Pinging 158.125.253.14 with 100 bytes of data:
Request timed out.
Request timed out.
Reply from 158.125.253.14: bytes=100 time=559ms TTL=125
Reply from 158.125.253.14: bytes=100 time=558ms TTL=125
Reply from 158.125.253.14: bytes=100 time=557ms TTL=125
Reply from 158.125.253.14: bytes=100 time=556ms TTL=125
Reply from 158.125.253.14: bytes=100 time=559ms TTL=125
Reply from 158.125.253.14: bytes=100 time=558ms TTL=125
Reply from 158.125.253.14: bytes=100 time=557ms TTL=125
Reply from 158.125.253.14: bytes=100 time=556ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 8, Lost = 2 (20\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 556ms, Maximum = 559ms, Average = 557ms
Delay300ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=606ms TTL=125
Reply from 158.125.253.14: bytes=100 time=605ms TTL=125
Reply from 158.125.253.14: bytes=100 time=604ms TTL=125
Reply from 158.125.253.14: bytes=100 time=607ms TTL=125
Reply from 158.125.253.14: bytes=100 time=606ms TTL=125
Reply from 158.125.253.14: bytes=100 time=605ms TTL=125
Reply from 158.125.253.14: bytes=100 time=604ms TTL=125
Reply from 158.125.253.14: bytes=100 time=607ms TTL=125
Reply from 158.125.253.14: bytes=100 time=606ms TTL=125
Reply from 158.125.253.14: bytes=100 time=605ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 604ms, Maximum = 607ms, Average = 605ms
Delay320ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=637ms TTL=125
Reply from 158.125.253.14: bytes=100 time=636ms TTL=125
Reply from 158.125.253.14: bytes=100 time=639ms TTL=125
Reply from 158.125.253.14: bytes=100 time=638ms TTL=125
Reply from 158.125.253.14: bytes=100 time=637ms TTL=125
Reply from 158.125.253.14: bytes=100 time=636ms TTL=125
Reply from 158.125.253.14: bytes=100 time=639ms TTL=125
Reply from 158.125.253.14: bytes=100 time=638ms TTL=125
Reply from 158.125.253.14: bytes=100 time=638ms TTL=125
Reply from 158.125.253.14: bytes=100 time=637ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 636ms, Maximum = 639ms, Average = 637ms
Delay340ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=686ms TTL=125
Reply from 158.125.253.14: bytes=100 time=686ms TTL=125
```

```
Reply from 158.125.253.14: bytes=100 time=685ms TTL=125
Reply from 158.125.253.14: bytes=100 time=684ms TTL=125
Reply from 158.125.253.14: bytes=100 time=687ms TTL=125
Reply from 158.125.253.14: bytes=100 time=686ms TTL=125
Reply from 158.125.253.14: bytes=100 time=686ms TTL=125
Reply from 158.125.253.14: bytes=100 time=685ms TTL=125
Reply from 158.125.253.14: bytes=100 time=688ms TTL=125
Reply from 158.125.253.14: bytes=100 time=688ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 684ms, Maximum = 688ms, Average = 686ms
Delay360ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=716ms TTL=125
Reply from 158.125.253.14: bytes=100 time=719ms TTL=125
Reply from 158.125.253.14: bytes=100 time=719ms TTL=125
Reply from 158.125.253.14: bytes=100 time=718ms TTL=125
Reply from 158.125.253.14: bytes=100 time=718ms TTL=125
Reply from 158.125.253.14: bytes=100 time=718ms TTL=125
Reply from 158.125.253.14: bytes=100 time=718ms TTL=125
Reply from 158.125.253.14: bytes=100 time=718ms TTL=125
Reply from 158.125.253.14: bytes=100 time=718ms TTL=125
Reply from 158.125.253.14: bytes=100 time=718ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 716ms, Maximum = 719ms, Average = 718ms
Delay380ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=765ms TTL=125
Reply from 158.125.253.14: bytes=100 time=764ms TTL=125
Reply from 158.125.253.14: bytes=100 time=767ms TTL=125
Reply from 158.125.253.14: bytes=100 time=766ms TTL=125
Reply from 158.125.253.14: bytes=100 time=765ms TTL=125
Reply from 158.125.253.14: bytes=100 time=764ms TTL=125
Reply from 158.125.253.14: bytes=100 time=767ms TTL=125
Reply from 158.125.253.14: bytes=100 time=767ms TTL=125
Reply from 158.125.253.14: bytes=100 time=766ms TTL=125
Reply from 158.125.253.14: bytes=100 time=766ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 764ms, Maximum = 767ms, Average = 765ms
Delay400ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=797ms TTL=125
Reply from 158.125.253.14: bytes=100 time=797ms TTL=125
Reply from 158.125.253.14: bytes=100 time=796ms TTL=125
Reply from 158.125.253.14: bytes=100 time=799ms TTL=125
Reply from 158.125.253.14: bytes=100 time=798ms TTL=125
Reply from 158.125.253.14: bytes=100 time=797ms TTL=125
Reply from 158.125.253.14: bytes=100 time=796ms TTL=125
Reply from 158.125.253.14: bytes=100 time=799ms TTL=125
Reply from 158.125.253.14: bytes=100 time=798ms TTL=125
Reply from 158.125.253.14: bytes=100 time=797ms TTL=125
```

```
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 796ms, Maximum = 799ms, Average = 797ms
Delay420ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=844ms TTL=125
Reply from 158.125.253.14: bytes=100 time=848ms TTL=125
Reply from 158.125.253.14: bytes=100 time=847ms TTL=125
Reply from 158.125.253.14: bytes=100 time=846ms TTL=125
Reply from 158.125.253.14: bytes=100 time=845ms TTL=125
Reply from 158.125.253.14: bytes=100 time=848ms TTL=125
Reply from 158.125.253.14: bytes=100 time=847ms TTL=125
Reply from 158.125.253.14: bytes=100 time=846ms TTL=125
Reply from 158.125.253.14: bytes=100 time=845ms TTL=125
Reply from 158.125.253.14: bytes=100 time=848ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 844ms, Maximum = 848ms, Average = 846ms
Delay440ms
Pinging 158.125.253.14 with 100 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Reply from 158.125.253.14: bytes=100 time=877ms TTL=125
Reply from 158.125.253.14: bytes=100 time=877ms TTL=125
Reply from 158.125.253.14: bytes=100 time=877ms TTL=125
Reply from 158.125.253.14: bytes=100 time=877ms TTL=125
Reply from 158.125.253.14: bytes=100 time=877ms TTL=125
Reply from 158.125.253.14: bytes=100 time=877ms TTL=125
Reply from 158.125.253.14: bytes=100 time=877ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 7, Lost = 3 (30\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 877ms, Maximum = 877ms, Average = 877ms
Delay460ms
Pinging 158.125.253.14 with 100 bytes of data:
Request timed out.
Reply from 158.125.253.14: bytes=100 time=927ms TTL=125
Reply from 158.125.253.14: bytes=100 time=926ms TTL=125
Reply from 158.125.253.14: bytes=100 time=925ms TTL=125
Reply from 158.125.253.14: bytes=100 time=924ms TTL=125
Reply from 158.125.253.14: bytes=100 time=927ms TTL=125
Reply from 158.125.253.14: bytes=100 time=926ms TTL=125
Reply from 158.125.253.14: bytes=100 time=925ms TTL=125
Reply from 158.125.253.14: bytes=100 time=924ms TTL=125
Reply from 158.125.253.14: bytes=100 time=927ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 9, Lost = 1 (10\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 924ms, Maximum = 927ms, Average = 925ms
Delay480ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=958ms TTL=125
Reply from 158.125.253.14: bytes=100 time=956ms TTL=125
```

```
Reply from 158.125.253.14: bytes=100 time=958ms TTL=125
Reply from 158.125.253.14: bytes=100 time=957ms TTL=125
Reply from 158.125.253.14: bytes=100 time=959ms TTL=125
Reply from 158.125.253.14: bytes=100 time=959ms TTL=125
Reply from 158.125.253.14: bytes=100 time=958ms TTL=125
Reply from 158.125.253.14: bytes=100 time=956ms TTL=125
Reply from 158.125.253.14: bytes=100 time=959ms TTL=125
Reply from 158.125.253.14: bytes=100 time=958ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 956ms, Maximum = 959ms, Average = 957ms
Delay500ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1006ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1007ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1007ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1007ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1007ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1007ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1007ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1007ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1007ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1007ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1006ms, Maximum = 1007ms, Average = 1006ms
Delay520ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1040ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1039ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1039ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1039ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1039ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1039ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1039ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1039ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1039ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1039ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1039ms, Maximum = 1040ms, Average = 1039ms
Delay540ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1086ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1087ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1087ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1087ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1087ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1087ms TTL=124
Request timed out.
Request timed out.
Request timed out.
Request timed out.
```

```
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 6, Lost = 4 (40\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1086ms, Maximum = 1087ms, Average = 1086ms
Delay560ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1117ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1119ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1119ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1119ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1119ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1119ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1119ms TTL=125
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 7, Lost = 3 (30\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1117ms, Maximum = 1119ms, Average = 1118ms
Delay580ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1165ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1167ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1167ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1167ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1167ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1167ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1167ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1167ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1167ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1167ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1165ms, Maximum = 1167ms, Average = 1166ms
Delay600ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1198ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1199ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1199ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1199ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1199ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1199ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1199ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1199ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1199ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1199ms TTL=124
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1198ms, Maximum = 1199ms, Average = 1198ms
Delay620ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1248ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1247ms TTL=124
```

```
Reply from 158.125.253.14: bytes=100 time=1247ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1247ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1247ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1247ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1247ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1247ms TTL=124
Request timed out.
Request timed out.
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 8, Lost = 2 (20\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1247ms, Maximum = 1248ms, Average = 1247ms
Delay640ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1279ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1279ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1279ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1279ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1279ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1279ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1279ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1279ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1279ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1279ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1279ms, Maximum = 1279ms, Average = 1279ms
Delay660ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1325ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1327ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1327ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1327ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1327ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1327ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1327ms TTL=124
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 7, Lost = 3 (30\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1325ms, Maximum = 1327ms, Average = 1326ms
Delay680ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1359ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1359ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1359ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1359ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1359ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1359ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1359ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1359ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1359ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1359ms TTL=125
```

```
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1359ms, Maximum = 1359ms, Average = 1359ms
Delay700ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1407ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1407ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1407ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1407ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1407ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1407ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1407ms TTL=124
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 7, Lost = 3 (30\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1407ms, Maximum = 1407ms, Average = 1407ms
Delay720ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1437ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1439ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1439ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1439ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1439ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1439ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1439ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1439ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1439ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1439ms TTL=124
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1437ms, Maximum = 1439ms, Average = 1438ms
Delay740ms
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1486ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1487ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1487ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1487ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1487ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1487ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1487ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1487ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1487ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1487ms TTL=124
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1486ms, Maximum = 1487ms, Average = 1486ms
Delay760ms
Pinging 158.125.253.14 with 100 bytes of data:
Request timed out.
Request timed out.
```

```
Request timed out.
Request timed out.
Reply from 158.125.253.14: bytes=100 time=1519ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1519ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1519ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1519ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1519ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1519ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 6, Lost = 4 (40\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1519ms, Maximum = 1519ms, Average = 1519ms
Delay780ms
Pinging 158.125.253.14 with 100 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Reply from 158.125.253.14: bytes=100 time=1565ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1567ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1567ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1567ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1568ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1566ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1567ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 7, Lost = 3 (30\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1565ms, Maximum = 1568ms, Average = 1566ms
Delay800ms
Pinging 158.125.253.14 with 100 bytes of data:
Request timed out.
Reply from 158.125.253.14: bytes=100 time=1600ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1599ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1599ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1599ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1599ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1599ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1599ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1599ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1599ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 9, Lost = 1 (10\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1599ms, Maximum = 1600ms, Average = 1599ms
Plr 0 percent
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time<1ms TTL=125
Reply from 158.125.253.14: bytes=100 time<1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time<1ms TTL=125
Reply from 158.125.253.14: bytes=100 time<1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time<1ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1ms TTL=124
```

```
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
Plr 1 percent
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time<1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time<1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
Plr 2 percent
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1ms TTL=124
Reply from 158.125.253.14: bytes=100 time=1ms TTL=124
Reply from 158.125.253.14: bytes=100 time=2ms TTL=124
Reply from 158.125.253.14: bytes=100 time<1ms TTL=124
Reply from 158.125.253.14: bytes=100 time=2ms TTL=124
Reply from 158.125.253.14: bytes=100 time=2ms TTL=124
Reply from 158.125.253.14: bytes=100 time=2ms TTL=124
Reply from 158.125.253.14: bytes=100 time=2ms TTL=124
Reply from 158.125.253.14: bytes=100 time=2ms TTL=124
Request timed out.
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 9, Lost = 1 (10\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 1ms
Plr 3 percent
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=2ms TTL=124
Reply from 158.125.253.14: bytes=100 time<1ms TTL=124
Reply from 158.125.253.14: bytes=100 time<1ms TTL=124
Reply from 158.125.253.14: bytes=100 time=2ms TTL=124
Reply from 158.125.253.14: bytes=100 time=2ms TTL=124
Reply from 158.125.253.14: bytes=100 time=2ms TTL=124
Reply from 158.125.253.14: bytes=100 time=2ms TTL=124
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 7, Lost = 3 (30\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 1ms
Jitter0
Pinging 158.125.253.14 with 100 bytes of data:
Request timed out.
Request timed out.
```

```
Request timed out.
Request timed out.
Reply from 158.125.253.14: bytes=100 time=699ms TTL=125
Reply from 158.125.253.14: bytes=100 time=498ms TTL=125
Reply from 158.125.253.14: bytes=100 time=897ms TTL=125
Reply from 158.125.253.14: bytes=100 time=600ms TTL=125
Reply from 158.125.253.14: bytes=100 time=800ms TTL=125
Reply from 158.125.253.14: bytes=100 time=400ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 6, Lost = 4 (40\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 400ms, Maximum = 897ms, Average = 649ms
Jitter 1
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=897ms TTL=124
Reply from 158.125.253.14: bytes=100 time=497ms TTL=124
Reply from 158.125.253.14: bytes=100 time=600ms TTL=124
Reply from 158.125.253.14: bytes=100 time=899ms TTL=124
Reply from 158.125.253.14: bytes=100 time=199ms TTL=124
Reply from 158.125.253.14: bytes=100 time=398ms TTL=124
Reply from 158.125.253.14: bytes=100 time=798ms TTL=124
Reply from 158.125.253.14: bytes=100 time=698ms TTL=124
Reply from 158.125.253.14: bytes=100 time=696ms TTL=124
Reply from 158.125.253.14: bytes=100 time=499ms TTL=124
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 199ms, Maximum = 899ms, Average = 618ms
Jitter 2
Pinging 158.125.253.14 with 100 bytes of data:
Request timed out.
Request timed out.
Reply from 158.125.253.14: bytes=100 time<1ms TTL=125
Reply from 158.125.253.14: bytes=100 time=496ms TTL=125
Reply from 158.125.253.14: bytes=100 time=399ms TTL=125
Reply from 158.125.253.14: bytes=100 time=398ms TTL=125
Reply from 158.125.253.14: bytes=100 time=97ms TTL=125
Reply from 158.125.253.14: bytes=100 time=496ms TTL=125
Reply from 158.125.253.14: bytes=100 time=399ms TTL=125
Reply from 158.125.253.14: bytes=100 time=398ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 8, Lost = 2 (20\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 496ms, Average = 335ms
Jitter 3
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=197ms TTL=124
Reply from 158.125.253.14: bytes=100 time=700ms TTL=124
Reply from 158.125.253.14: bytes=100 time=599ms TTL=124
Reply from 158.125.253.14: bytes=100 time=398ms TTL=124
Reply from 158.125.253.14: bytes=100 time=98ms TTL=124
Reply from 158.125.253.14: bytes=100 time=400ms TTL=124
Request timed out.
Request timed out.
Request timed out.
Request timed out.
```

```
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 6, Lost = 4 (40\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 98ms, Maximum = 700ms, Average = 398ms
Jitter 4
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=499ms TTL=125
Reply from 158.125.253.14: bytes=100 time=797ms TTL=125
Reply from 158.125.253.14: bytes=100 time=496ms TTL=125
Reply from 158.125.253.14: bytes=100 time=398ms TTL=125
Reply from 158.125.253.14: bytes=100 time=897ms TTL=125
Reply from 158.125.253.14: bytes=100 time=496ms TTL=125
Reply from 158.125.253.14: bytes=100 time=798ms TTL=125
Reply from 158.125.253.14: bytes=100 time=397ms TTL=125
Reply from 158.125.253.14: bytes=100 time=396ms TTL=125
Reply from 158.125.253.14: bytes=100 time=700ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 396ms, Maximum = 897ms, Average = 587ms
Jitter 5
Pinging 158.125.253.14 with 100 bytes of data:
Request timed out.
Request timed out.
Reply from 158.125.253.14: bytes=100 time=498ms TTL=125
Reply from 158.125.253.14: bytes=100 time=396ms TTL=125
Reply from 158.125.253.14: bytes=100 time=799ms TTL=125
Reply from 158.125.253.14: bytes=100 time=598ms TTL=125
Reply from 158.125.253.14: bytes=100 time=397ms TTL=125
Reply from 158.125.253.14: bytes=100 time=97ms TTL=125
Reply from 158.125.253.14: bytes=100 time=496ms TTL=125
Reply from 158.125.253.14: bytes=100 time=198ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 8, Lost = 2 (20\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 97ms, Maximum = 799ms, Average = 434ms
Jitter 6
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=797ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1097ms TTL=125
Reply from 158.125.253.14: bytes=100 time=599ms TTL=125
Reply from 158.125.253.14: bytes=100 time=698ms TTL=125
Reply from 158.125.253.14: bytes=100 time=397ms TTL=125
Reply from 158.125.253.14: bytes=100 time=396ms TTL=125
Reply from 158.125.253.14: bytes=100 time=599ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1098ms TTL=124
Reply from 158.125.253.14: bytes=100 time=799ms TTL=124
Reply from 158.125.253.14: bytes=100 time=899ms TTL=124
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 396ms, Maximum = 1098ms, Average = 737ms
Jitter 7
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=1097ms TTL=125
Reply from 158.125.253.14: bytes=100 time=899ms TTL=125
```

```
Reply from 158.125.253.14: bytes=100 time=898ms TTL=125
Reply from 158.125.253.14: bytes=100 time=997ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1200ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1199ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1199ms TTL=125
Reply from 158.125.253.14: bytes=100 time=899ms TTL=125
Reply from 158.125.253.14: bytes=100 time=1098ms TTL=125
Reply from 158.125.253.14: bytes=100 time=899ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 898ms, Maximum = 1200ms, Average = 1038ms
Jitter 8
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=799ms TTL=125
Reply from 158.125.253.14: bytes=100 time=799ms TTL=125
Reply from 158.125.253.14: bytes=100 time=798ms TTL=125
Reply from 158.125.253.14: bytes=100 time=597ms TTL=125
Reply from 158.125.253.14: bytes=100 time=796ms TTL=125
Reply from 158.125.253.14: bytes=100 time=800ms TTL=125
Reply from 158.125.253.14: bytes=100 time=800ms TTL=125
Reply from 158.125.253.14: bytes=100 time=700ms TTL=125
Reply from 158.125.253.14: bytes=100 time=799ms TTL=125
Reply from 158.125.253.14: bytes=100 time=798ms TTL=125
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 597ms, Maximum = 800ms, Average = 768ms
Jitter 9
Pinging 158.125.253.14 with 100 bytes of data:
Reply from 158.125.253.14: bytes=100 time=398ms TTL=125
Reply from 158.125.253.14: bytes=100 time=398ms TTL=124
Reply from 158.125.253.14: bytes=100 time=298ms TTL=124
Reply from 158.125.253.14: bytes=100 time=398ms TTL=124
Reply from 158.125.253.14: bytes=100 time=298ms TTL=124
Reply from 158.125.253.14: bytes=100 time=398ms TTL=124
Reply from 158.125.253.14: bytes=100 time=297ms TTL=124
Reply from 158.125.253.14: bytes=100 time=297ms TTL=124
Reply from 158.125.253.14: bytes=100 time=397ms TTL=124
Reply from 158.125.253.14: bytes=100 time=397ms TTL=124
Ping statistics for 158.125.253.14:
    Packets: Sent = 10, Received = 10, Lost = 0 (0\% loss),
Approximate round trip times in milli-seconds:
    Minimum = 297ms, Maximum = 398ms, Average = 357ms
```

D.1.3 Sample Report of emodel.java

Term:
Send Loudness Rating(SLR)
Receive Loudness Rating(RLR)
D-factor(Send) Ds
Sidetone Masking Rating(STMR)
D-factor (Receive)Dr
Talker Echo Loudness Rating(TELR)
Weighted Echo Path Loss(WEPL)

Equipment Impairment Factor(Ie)
Packet-loss Robustness Factor(BPL)
Burst Ratio(BurstR)
Advantage Factor(A)
Electric Circuit Noise(Nc)
Room Noise (Send)Ps
Room Noise (Receive)Pr
Quantizing Distortion Units(qdu)
and Noise Floor(Nfor)

The default setting:
$SLR = 8dB, RLR = 2dB, Ds = 3, STMR = 15dB, Dr = 3, TELR = 65dB, WEPL = 110dB, Ie = 0, BPL = 1$
$BurstR = 1, A = 0, Nc = -70dBm0p, Ps = 35dB, Pr = 35dB, qdu = 1, Nfor = -64dBmp$

```
One_way_delay(ms) Packet_Loss_Prob(%)  R-Factor MOS
0 0 93.21 4.41
10 0 92.85 4.4
20 0 92.53 4.4
30 0 92.25 4.39
40 0 92 4.38
50 0 91.76 4.38
60 0 91.52 4.37
70 0 91.3 4.37
80 0 91.08 4.36
90 0 90.87 4.36
100 0 90.66 4.35
110 0 90.46 4.35
120 0 90.26 4.35
130 0 90.06 4.34
140 0 89.83 4.33
150 0 89.54 4.33
160 0 89.14 4.32
170 0 88.58 4.3
180 0 87.83 4.28
190 0 86.9 4.26
200 0 85.8 4.22
210 0 84.58 4.18
220 0 83.28 4.14
230 0 81.92 4.09
240 0 80.55 4.04
250 0 79.18 3.99
260 0 77.82 3.94
270 0 76.48 3.88
280 0 75.18 3.83
290 0 73.9 3.77
300 0 72.67 3.72
310 0 71.47 3.66
320 0 70.3 3.61
330 0 69.17 3.56
340 0 68.08 3.51
350 0 67.03 3.45
360 0 66.01 3.4
370 0 65.02 3.36
380 0 64.06 3.31
390 0 63.14 3.26
```

```
400 0 62.25 3.22
0 0 93.21 4.41
0 0.1 84.57 4.18
0 0.2 77.37 3.92
0 0.3 71.28 3.66
0 0.4 66.06 3.41
0 0.5 61.54 3.18
0 0.6 57.58 2.97
0 0.7 54.09 2.79
0 0.8 50.98 2.63
0 0.9 48.21 2.48
0 1 45.71 2.35
0 1.1 43.44 2.24
0 1.2 41.39 2.13
0 1.3 39.51 2.04
0 1.4 37.79 1.96
0 1.5 36.21 1.88
0 1.6 34.74 1.82
0 1.7 33.39 1.75
0 1.8 32.13 1.7
0 1.9 30.96 1.65
0 2 29.87 1.6
0 2.1 28.85 1.56
0 2.2 27.89 1.52
0 2.3 26.99 1.49
0 2.4 26.15 1.46
0 2.5 25.35 1.43
0 2.6 24.6 1.4
0 2.7 23.88 1.38
0 2.8 23.21 1.35
0 2.9 22.57 1.33
0 3 21.96 1.31
```

D.1.4 Router log

D.1.4.1 TR

```
vyatta@TR:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

S>* 0.0.0.0/0 [1/0] via 158.125.253.17, eth0
C>* 10.0.1.0/24 is directly connected, lo
C>* 10.0.2.0/24 is directly connected, lo
O   10.0.3.0/24 [110/10] is directly connected, tun0, 01w5d00h
C>* 10.0.3.0/24 is directly connected, tun0
O   10.0.4.0/24 [110/10] is directly connected, tun1, 01w5d00h
C>* 10.0.4.0/24 is directly connected, tun1
C>* 127.0.0.0/8 is directly connected, lo
C>* 158.125.253.16/30 is directly connected, eth0
S>* 158.125.253.24/29 [1/0] via 158.125.253.17, eth0
vyatta@TR:~$ show interfaces
Interface    IP Address         State       Link   Description
eth0         158.125.253.18/30  up          up     TR-R3
```

```
lo           127.0.0.1/8        up          up     net1-R2-net2-R1
lo           10.0.1.18/24       up          up     net1-R2-net2-R1
lo           10.0.2.18/24       up          up     net1-R2-net2-R1
lo           ::1/128            up          up     net1-R2-net2-R1
tun0         10.0.3.18/24       up          up     IPIP-tunnel-Router2
tun1         10.0.4.18/24       up          up     IPIP-tunnel-Router1
tunl0        -                  admin down  down
vyatta@TR:~$
```

D.1.4.2 Router1

Log on IPIP connectivity:

```
vyatta@Router1:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

S>* 0.0.0.0/0 [120/0] via 158.125.253.1, eth1
O>* 10.0.3.0/24 [110/30] via 158.125.253.1, eth1, 3d23h33m
O   10.0.4.0/24 [110/10] is directly connected, tun1, 03w6d02h
C>* 10.0.4.0/24 is directly connected, tun1
C>* 10.0.6.0/24 is directly connected, lo
C>* 10.0.12.0/24 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
S>* 131.231.127.0/24 [115/0] via 158.125.253.1, eth1
O   158.125.253.0/30 [110/10] is directly connected, eth1, 03w6d02h
C>* 158.125.253.0/30 is directly connected, eth1
C>* 158.125.253.4/30 is directly connected, eth0
O   158.125.253.4/30 [110/10] is directly connected, eth0, 03w6d02h
O>* 158.125.253.8/30 [110/20] via 158.125.253.1, eth1, 01w5d21h
O>* 158.125.253.12/30 [110/30] via 158.125.253.1, eth1, 3d23h33m
O>* 158.125.253.16/30 [110/20] via 158.125.253.1, eth1, 6d01h29m
O   158.125.253.24/29 [110/10] is directly connected, eth2, 6d00h07m
C>* 158.125.253.24/29 is directly connected, eth2
O>* 192.168.1.0/24 [110/30] via 158.125.253.1, eth1, 3d23h33m

vyatta@Router1:~$ show interfaces
Interface    IP Address         State       Link   Description
eth0         158.125.253.5/30   up          up     Router1 to segment 158.125.253.4/30-PC1
eth1         158.125.253.2/30   up          up     Router1 to Router3
eth2         158.125.253.25/29  up          up     VLAN4
lo           127.0.0.1/8        up          up     net6-TR-net12-TR-TJ
lo           10.0.6.2/24        up          up     net6-TR-net12-TR-TJ
lo           10.0.12.2/24       up          up     net6-TR-net12-TR-TJ
lo           ::1/128            up          up     net6-TR-net12-TR-TJ
tun0         10.0.8.2/24        admin down  down   IPIP-TR-FJ
tun1         10.0.4.2/24        up          up     IPIP-TR
tunl0        -                  admin down  down
```

Log on IPSec connectivity:

```
vyatta@Router1:~$ show vpn ipsec sa
```

```
Peer            Tunnel# Dir SPI      Encrypt    Hash       NAT-T A-Time L-Time
-------         ------- --- ---      -------    ----       ----- ------ ------
158.125.253.18  1       in  e5eed198 aes128     sha1       No    1895   3600
158.125.253.18  1       out 1e5bcfbe aes128     sha1       No    1895   3600

vyatta@Router1:~$ show vpn ipsec sa statistics
Peer            Dir SRC Network        DST Network        Bytes
-------         --- -----------        -----------        -----
158.125.253.18  in  10.0.6.2/32        10.0.2.18/32       97156
158.125.253.18  out 10.0.2.18/32       10.0.6.2/32        30060

vyatta@Router1:~$ show vpn ipsec status
IPSec Process Running PID: 7673

1 Active IPsec Tunnels

IPsec Interfaces :
eth1 (158.125.253.2)
IKE Process Running

PID: 7673
vyatta@Router1:~$ show vpn ike sa
Local           Peer            State     Encrypt   Hash     NAT-T A-Time L-Time
--------        -------         -----     -------   ----     ----- ------ ------
158.125.253.2   158.125.253.18  up        aes128    sha1     No    7072   28800

vyatta@Router1:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

S>* 0.0.0.0/0 [120/0] via 158.125.253.1, eth1
K>* 10.0.2.18/32 is directly connected, eth1
O>* 10.0.3.0/24 [110/30] via 158.125.253.1, eth1, 00:00:20
O   10.0.4.0/24 [110/10] is directly connected, tun1, 6d00h32m
C>* 10.0.4.0/24 is directly connected, tun1
C>* 10.0.6.0/24 is directly connected, lo
C>* 10.0.12.0/24 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
S>* 131.231.127.0/24 [115/0] via 158.125.253.1, eth1
O   158.125.253.0/30 [110/10] is directly connected, eth1, 6d00h10m
C>* 158.125.253.0/30 is directly connected, eth1
O   158.125.253.4/30 [110/10] is directly connected, eth0, 5d23h20m
C>* 158.125.253.4/30 is directly connected, eth0
O>* 158.125.253.8/30 [110/20] via 158.125.253.1, eth1, 5d22h37m
O>* 158.125.253.12/30 [110/30] via 158.125.253.1, eth1, 00:00:20
O>* 158.125.253.16/30 [110/20] via 158.125.253.1, eth1, 6d00h09m
O   158.125.253.24/29 [110/10] is directly connected, eth2, 6d00h08m
C>* 158.125.253.24/29 is directly connected, eth2
O>* 192.168.1.0/24 [110/30] via 158.125.253.1, eth1, 00:00:20

vyatta@Router1:~$ show interfaces
Interface    IP Address         State       Link   Description
eth0         158.125.253.5/30   up          up     Router1 to segment 158.125.253.4/30-PC1
eth1         158.125.253.2/30   up          up     Router1 to Router3
eth2         158.125.253.25/29  up          up     VLAN4
lo           127.0.0.1/8        up          up     net6-TR-net12-TR-TJ
```

```
lo           10.0.6.2/24        up          up     net6-TR-net12-TR-TJ
lo           10.0.12.2/24       up          up     net6-TR-net12-TR-TJ
lo           ::1/128            up          up     net6-TR-net12-TR-TJ
tun0         10.0.8.2/24        admin down  down   IPIP-TR-FJ
tun1         10.0.4.2/24        up          up     IPIP-TR
tunl0        -                  admin down  down
```

Log on OpenVPN TLS connectivity:

```
vyatta@Router1:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

S>* 0.0.0.0/0 [120/0] via 158.125.253.1, eth1
O>* 10.0.3.0/24 [110/30] via 158.125.253.1, eth1, 5d23h10m
O   10.0.4.0/24 [110/10] is directly connected, tun1, 04w1d01h
C>* 10.0.4.0/24 is directly connected, tun1
C>* 10.0.6.0/24 is directly connected, lo
C>* 10.0.12.0/24 is directly connected, lo
C>* 10.18.1.2/32 is directly connected, vtun0
C>* 127.0.0.0/8 is directly connected, lo
S>* 131.231.127.0/24 [115/0] via 158.125.253.1, eth1
O   158.125.253.0/30 [110/10] is directly connected, eth1, 04w1d01h
C>* 158.125.253.0/30 is directly connected, eth1
S   158.125.253.4/30 [1/0] is directly connected, vtun0
C>* 158.125.253.4/30 is directly connected, eth0
O   158.125.253.4/30 [110/10] is directly connected, eth0, 04w1d01h
O>* 158.125.253.8/30 [110/20] via 158.125.253.1, eth1, 02w0d20h
O>* 158.125.253.12/30 [110/30] via 158.125.253.1, eth1, 5d23h10m
O>* 158.125.253.16/30 [110/20] via 158.125.253.1, eth1, 01w1d01h
O   158.125.253.24/29 [110/10] is directly connected, eth2, 01w0d23h
C>* 158.125.253.24/29 is directly connected, eth2
O>* 192.168.1.0/24 [110/30] via 158.125.253.1, eth1, 5d23h10m
vyatta@Router1:~$ show interfaces
Interface    IP Address         State       Link   Description
eth0         158.125.253.5/30   up          up     Router1 to segment 158.125.253.4/30-PC1
eth1         158.125.253.2/30   up          up     Router1 to Router3
eth2         158.125.253.25/29  up          up     VLAN4
lo           127.0.0.1/8        up          up     net6-TR-net12-TR-TJ
lo           10.0.6.2/24        up          up     net6-TR-net12-TR-TJ
lo           10.0.12.2/24       up          up     net6-TR-net12-TR-TJ
lo           ::1/128            up          up     net6-TR-net12-TR-TJ
tun0         10.0.8.2/24        admin down  down   IPIP-TR-FJ
tun1         10.0.4.2/24        up          up     IPIP-TR
tunl0        -                  admin down  down
vtun0        10.18.1.1          up          up
vyatta@Router1:~$  show interfaces openvpn vtun0 brief
Interface    IP Address         State       Link   Description
vtun0        10.18.1.1          up          up
```

D.1.4.3 Router2

Log on IPIP connectivity:

```
vyatta@Router2:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

S>* 0.0.0.0/0 [120/0] via 158.125.253.9, eth1
O   10.0.3.0/24 [110/10] is directly connected, tun0, 03w6d02h
C>* 10.0.3.0/24 is directly connected, tun0
O>* 10.0.4.0/24 [110/30] via 158.125.253.9, eth1, 3d23h32m
C>* 10.0.5.0/24 is directly connected, lo
C>* 10.0.11.0/24 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
S>* 131.231.127.0/24 [115/0] via 158.125.253.9, eth1
O>* 158.125.253.0/30 [110/20] via 158.125.253.9, eth1, 03w6d02h
O>* 158.125.253.4/30 [110/30] via 158.125.253.9, eth1, 3d23h32m
O   158.125.253.8/30 [110/10] is directly connected, eth1, 03w6d02h
C>* 158.125.253.8/30 is directly connected, eth1
O   158.125.253.12/30 [110/10] is directly connected, eth0, 6d00h05m
C>* 158.125.253.12/30 is directly connected, eth0
O>* 158.125.253.16/30 [110/20] via 158.125.253.9, eth1, 6d01h28m
O>* 158.125.253.24/29 [110/30] via 158.125.253.9, eth1, 3d23h32m
O   192.168.1.0/24 [110/10] is directly connected, eth2, 03w6d02h
C>* 192.168.1.0/24 is directly connected, eth2
vyatta@Router2:~$ show interfaces
Interface    IP Address         State       Link   Description
eth0         158.125.253.13/30  up          up     Router2 to sgment 158.125.253.12/30-PC2
eth1         158.125.253.10/30  up          up     Router2 to Router3
eth2         192.168.1.1/24     up          up     To VLAN management
lo           127.0.0.1/8        up          up     net5-TR-net11-TR-TJ
lo           10.0.5.10/24       up          up     net5-TR-net11-TR-TJ
lo           10.0.11.10/24      up          up     net5-TR-net11-TR-TJ
lo           ::1/128            up          up     net5-TR-net11-TR-TJ
tun0         10.0.3.10/24       up          up     IPIP-TR
tun1         10.0.7.10/24       admin down  down   IPIP-TR-FJ
tun10        -                  admin down  down
```

Log on IPSec connectivity:

```
vyatta@Router2:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

S>* 0.0.0.0/0 [120/0] via 158.125.253.9, eth1
K>* 10.0.1.18/32 is directly connected, eth1
O   10.0.3.0/24 [110/10] is directly connected, tun0, 6d01h53m
C>* 10.0.3.0/24 is directly connected, tun0
O>* 10.0.4.0/24 [110/20] via 10.0.3.18, tun0, 00:00:45
C>* 10.0.5.0/24 is directly connected, lo
C>* 10.0.11.0/24 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
S>* 131.231.127.0/24 [115/0] via 158.125.253.9, eth1
O>* 158.125.253.0/30 [110/20] via 158.125.253.9, eth1, 5d22h31m
O>* 158.125.253.4/30 [110/30] via 10.0.3.18, tun0, 00:00:45
  *                            via 158.125.253.9, eth1, 00:00:45
O   158.125.253.8/30 [110/10] is directly connected, eth1, 5d22h32m
```

```
C>* 158.125.253.8/30 is directly connected, eth1
O   158.125.253.12/30 [110/10] is directly connected, eth0, 5d22h32m
C>* 158.125.253.12/30 is directly connected, eth0
O>* 158.125.253.16/30 [110/20] via 158.125.253.9, eth1, 5d22h31m
O>* 158.125.253.24/29 [110/30] via 10.0.3.18, tun0, 00:00:45
  *                            via 158.125.253.9, eth1, 00:00:45
O   192.168.1.0/24 [110/10] is directly connected, eth2, 5d22h33m
C>* 192.168.1.0/24 is directly connected, eth2
vyatta@Router2:~$ show vpn ipsec status
IPSec Process Running PID: 8832

1 Active IPsec Tunnels

IPsec Interfaces :
eth1 (158.125.253.10)
vyatta@Router2:~$ show vpn ipsec sa
Peer            Tunnel# Dir SPI      Encrypt    Hash       NAT-T A-Time L-Time
-------         ------- --- ---      -------    ----       ----- ------ ------
158.125.253.18  1       in  fee591cb aes128     sha1       No    1061   3600
158.125.253.18  1       out 71f6d70b aes128     sha1       No    1061   3600

vyatta@Router2:~$ show vpn ipsec sa statistics
Peer            Dir SRC Network        DST Network        Bytes
-------         --- -----------        -----------        -----
158.125.253.18  in  10.0.5.10/32       10.0.1.18/32       32389
158.125.253.18  out 10.0.1.18/32       10.0.5.10/32       18692

vyatta@Router2:~$ show vpn ike sa
Local           Peer            State     Encrypt   Hash     NAT-T A-Time L-Time
--------        -------         -----     -------   ----     ----- ------ ------
158.125.253.10  158.125.253.18  up        aes128    sha1     No    24362  28800

vyatta@Router2:~$ show interfaces
Interface    IP Address         State       Link   Description
eth0         158.125.253.13/30  up          up     Router2 to sgment 158.125.253.12/30-PC2
eth1         158.125.253.10/30  up          up     Router2 to Router3
eth2         192.168.1.1/24     up          up     To VLAN management
lo           127.0.0.1/8        up          up     net5-TR-net11-TR-TJ
lo           10.0.5.10/24       up          up     net5-TR-net11-TR-TJ
lo           10.0.11.10/24      up          up     net5-TR-net11-TR-TJ
lo           ::1/128            up          up     net5-TR-net11-TR-TJ
tun0         10.0.3.10/24       up          up     IPIP-TR
tun1         10.0.7.10/24       admin down  down   IPIP-TR-FJ
tunl0        -                  admin down  down
```

Log on OpenVPN TLS connectivity:

```
vyatta@Router2:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

S>* 0.0.0.0/0 [120/0] via 158.125.253.9, eth1
O   10.0.3.0/24 [110/10] is directly connected, tun0, 04w1d01h
C>* 10.0.3.0/24 is directly connected, tun0
```

```
O>* 10.0.4.0/24 [110/30] via 158.125.253.9, eth1, 5d23h05m
C>* 10.0.5.0/24 is directly connected, lo
C>* 10.0.11.0/24 is directly connected, lo
C>* 10.18.1.1/32 is directly connected, vtun0
C>* 127.0.0.0/8 is directly connected, lo
S>* 131.231.127.0/24 [115/0] via 158.125.253.9, eth1
O>* 158.125.253.0/30 [110/20] via 158.125.253.9, eth1, 04w1d01h
O>* 158.125.253.4/30 [110/30] via 158.125.253.9, eth1, 5d23h05m
O   158.125.253.8/30 [110/10] is directly connected, eth1, 04w1d01h
C>* 158.125.253.8/30 is directly connected, eth1
S   158.125.253.12/30 [1/0] is directly connected, vtun0
O   158.125.253.12/30 [110/10] is directly connected, eth0, 01w0d23h
C>* 158.125.253.12/30 is directly connected, eth0
O>* 158.125.253.16/30 [110/20] via 158.125.253.9, eth1, 01w1d01h
O>* 158.125.253.24/29 [110/30] via 158.125.253.9, eth1, 5d23h05m
O   192.168.1.0/24 [110/10] is directly connected, eth2, 04w1d01h
C>* 192.168.1.0/24 is directly connected, eth2
vyatta@Router2:~$ show interfaces
Interface    IP Address         State       Link   Description
eth0         158.125.253.13/30  up          up     Router2 to sgment 158.125.253.12/30-PC2
eth1         158.125.253.10/30  up          up     Router2 to Router3
eth2         192.168.1.1/24     up          up     To VLAN management
lo           127.0.0.1/8        up          up     net5-TR-net11-TR-TJ
lo           10.0.5.10/24       up          up     net5-TR-net11-TR-TJ
lo           10.0.11.10/24      up          up     net5-TR-net11-TR-TJ
lo           ::1/128            up          up     net5-TR-net11-TR-TJ
tun0         10.0.3.10/24       up          up     IPIP-TR
tun1         10.0.7.10/24       admin down  down   IPIP-TR-FJ
tunl0        -                  admin down  down
vtun0        10.18.1.2          up          up
vyatta@Router2:~$  show interfaces openvpn vtun0 brief
Interface    IP Address         State       Link   Description
vtun0        10.18.1.2          up          up
```

D.1.5 CACTI report:Snapshots

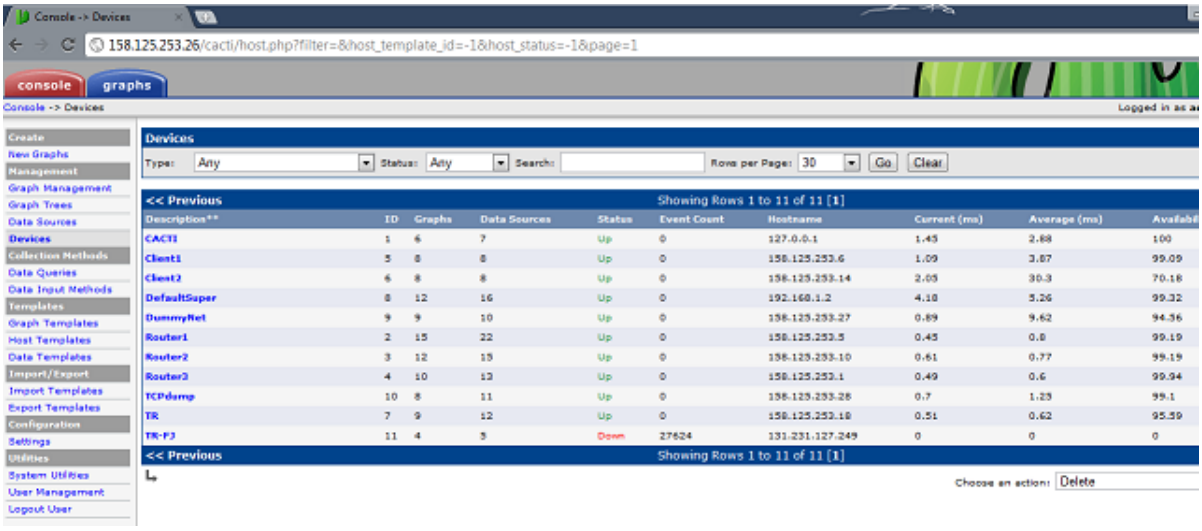


Figure D.1: List of devices on CACTI

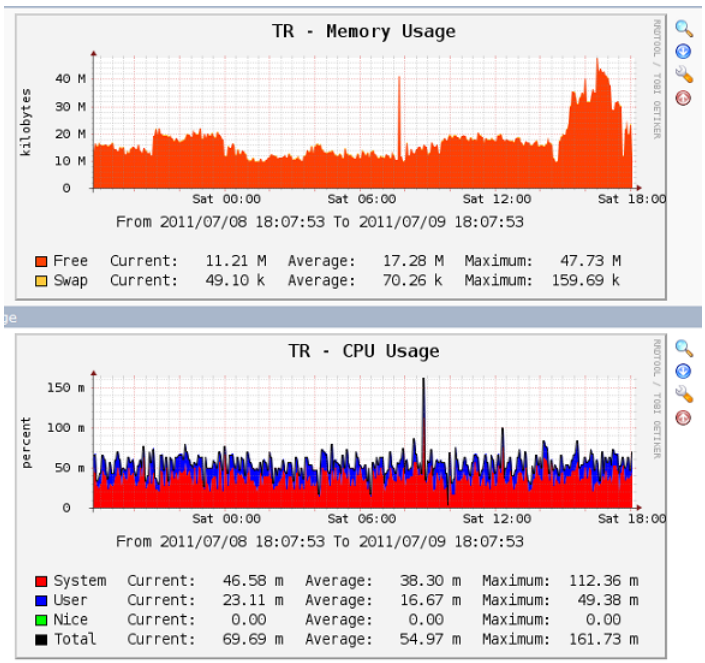


Figure D.2: Snapshot of TR memory usage and CPU usage

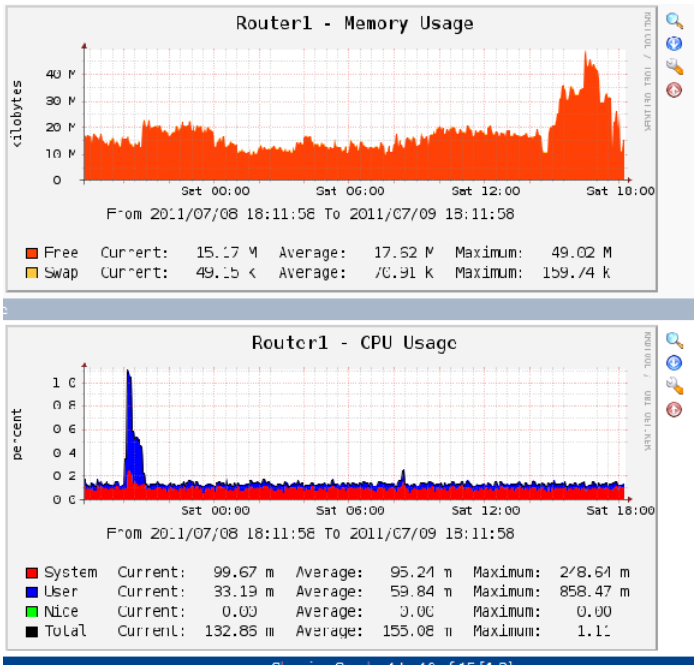


Figure D.3: Snapshot of Router1 memory usage and CPU usage

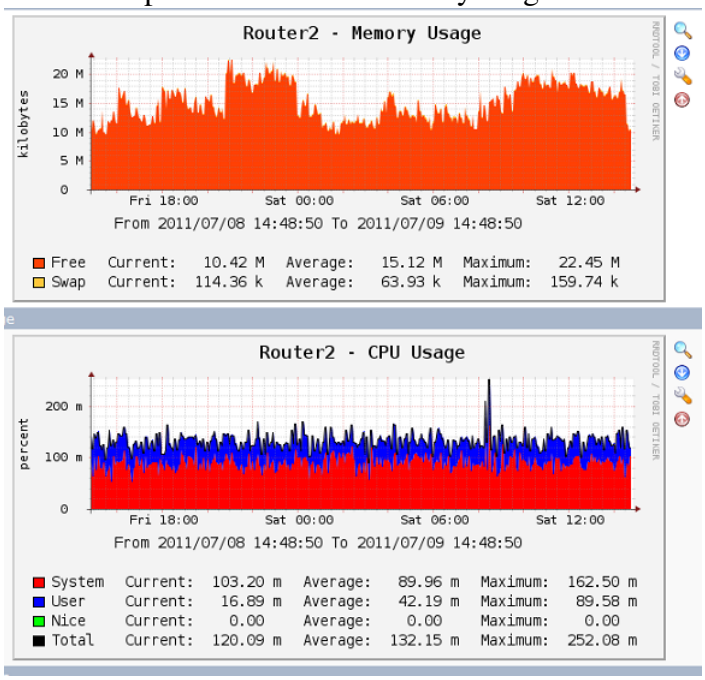


Figure D.4: Snapshot of Router2 memory usage and CPU usage

D.1.6 Validation: Box and Whisker Graphs

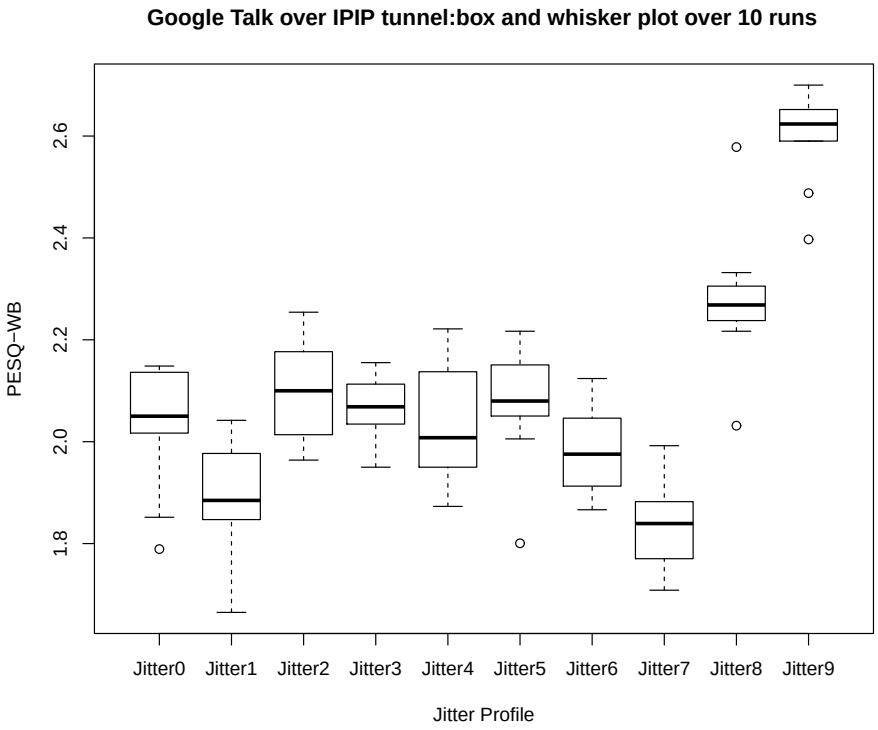


Figure D.5: Graph of Jitter versus PESQ-WB

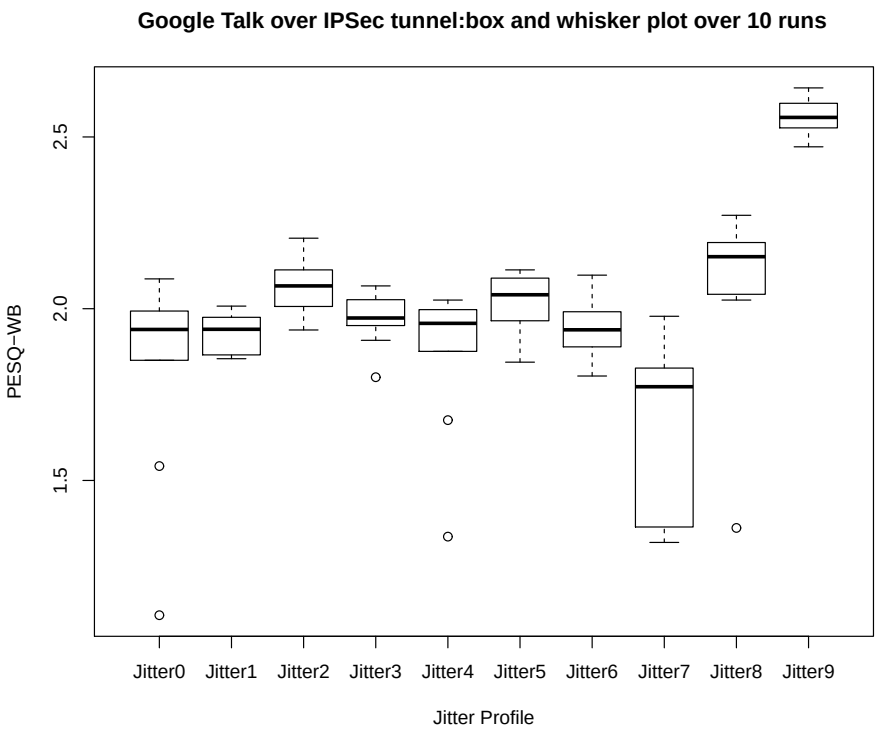


Figure D.6: Graph of Jitter versus PESQ-WB

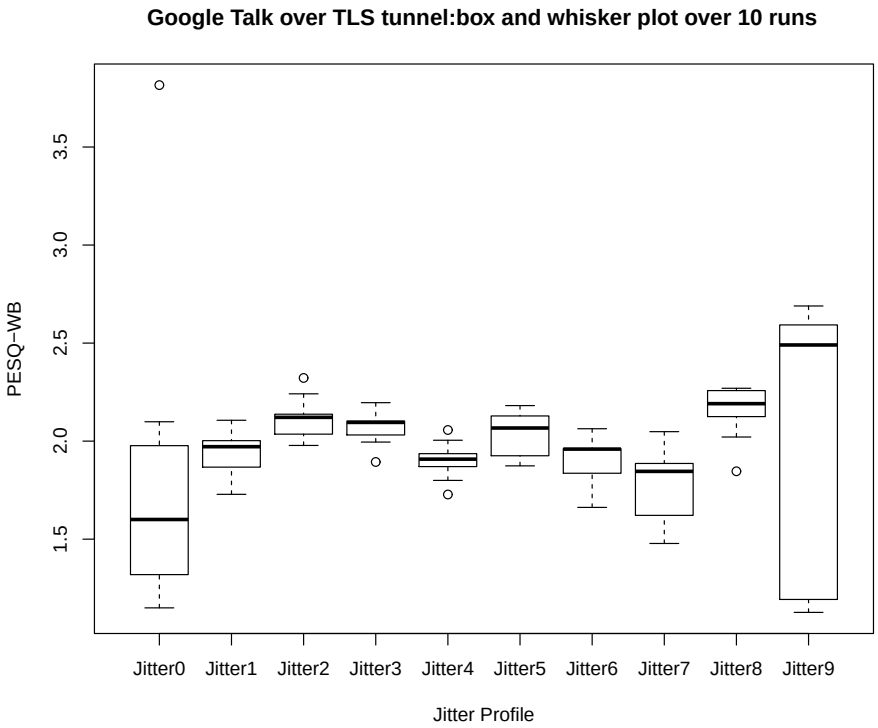


Figure D.7: Graph of Jitter versus PESQ-WB

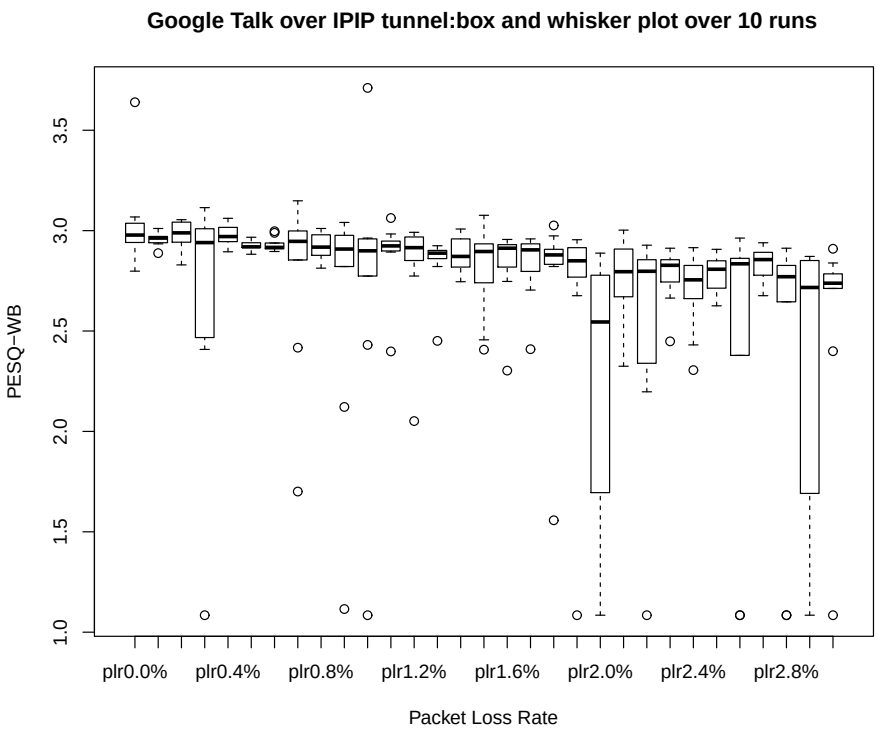


Figure D.8: Graph of Packet Loss Rate versus PESQ-WB

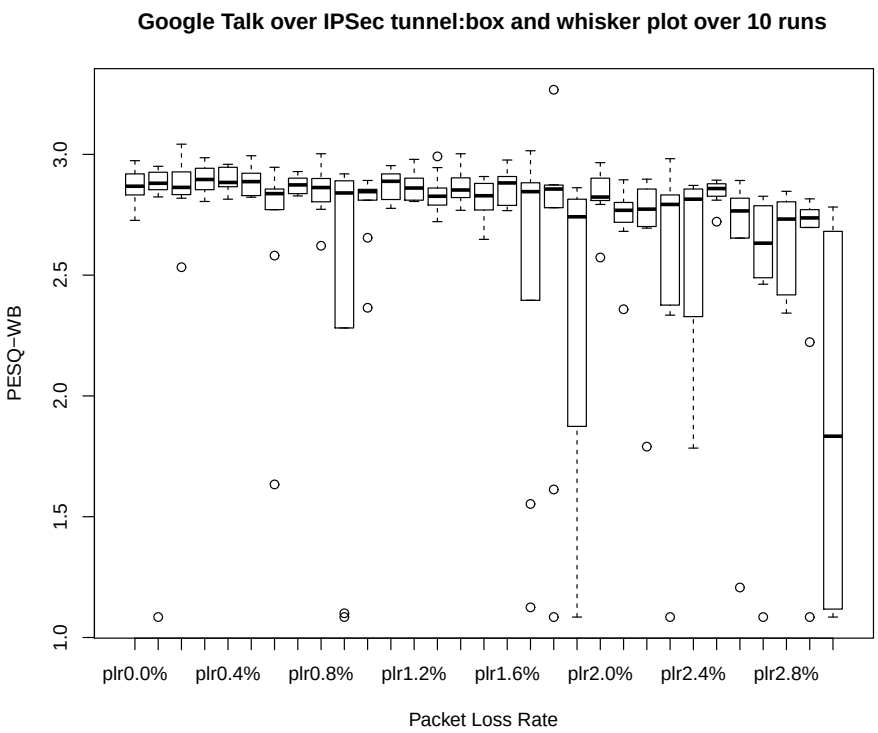


Figure D.9: Graph of Packet Loss Rate versus PESQ-WB

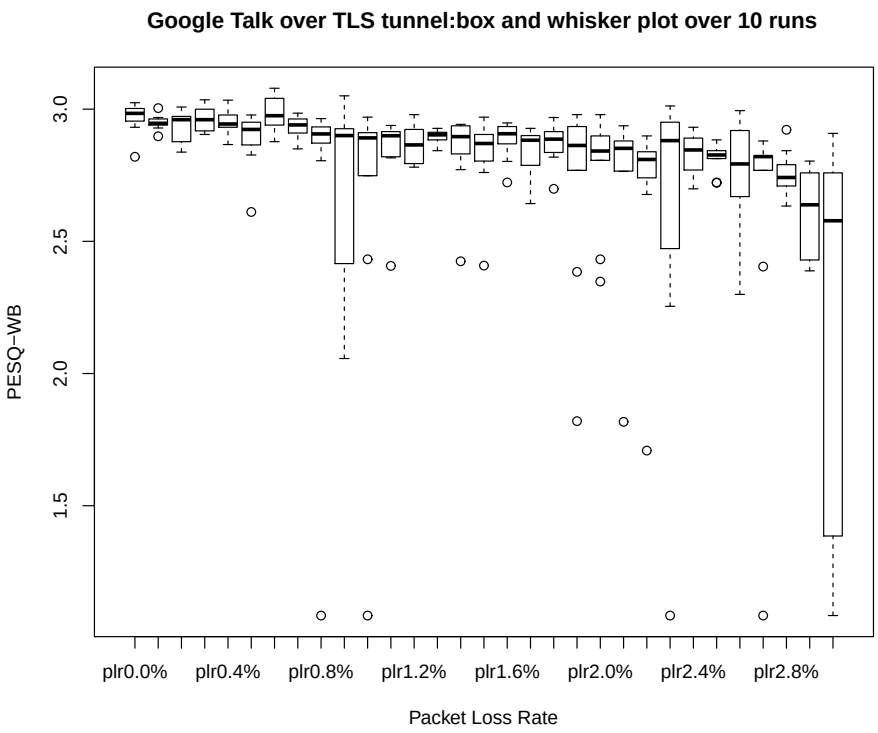


Figure D.10: Graph of Packet Loss Rate versus PESQ-WB

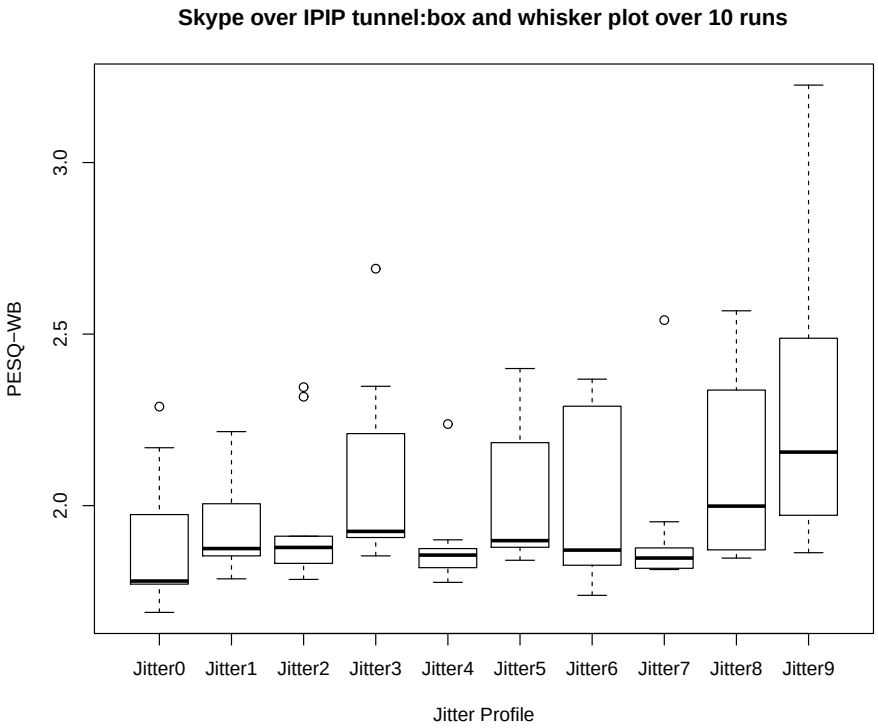


Figure D.11: Graph of Jitter versus PESQ-WB

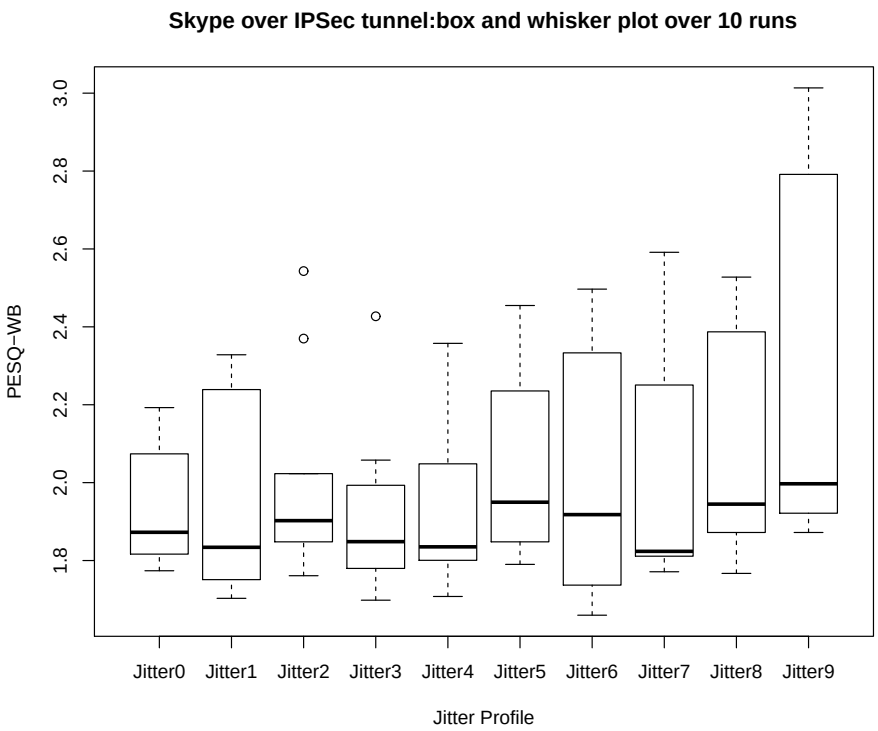


Figure D.12: Graph of Jitter versus PESQ-WB

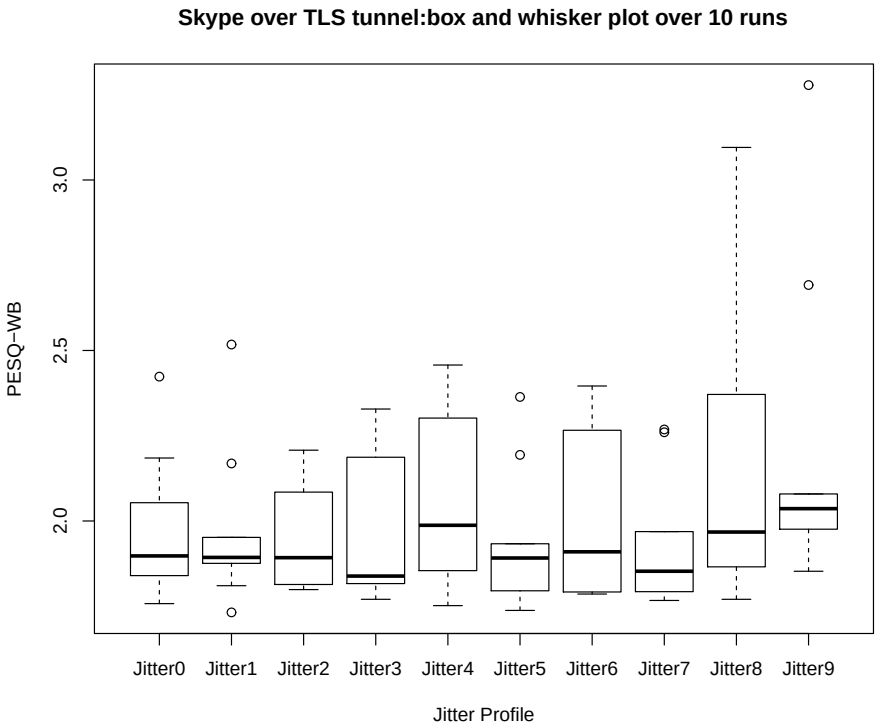


Figure D.13: Graph of Jitter versus PESQ-WB

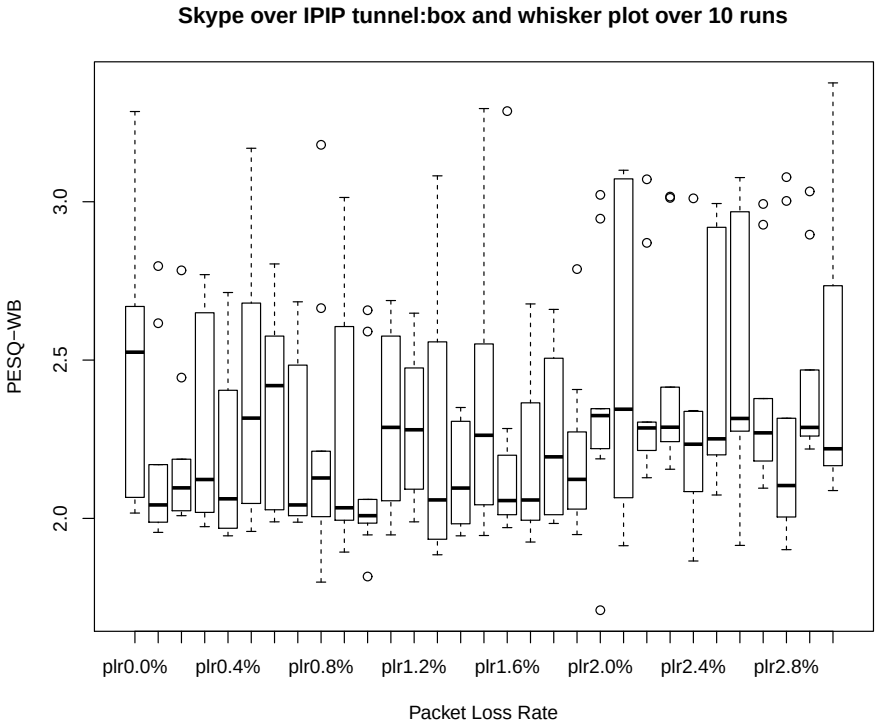


Figure D.14: Graph of Packet Loss Rate versus PESQ-WB

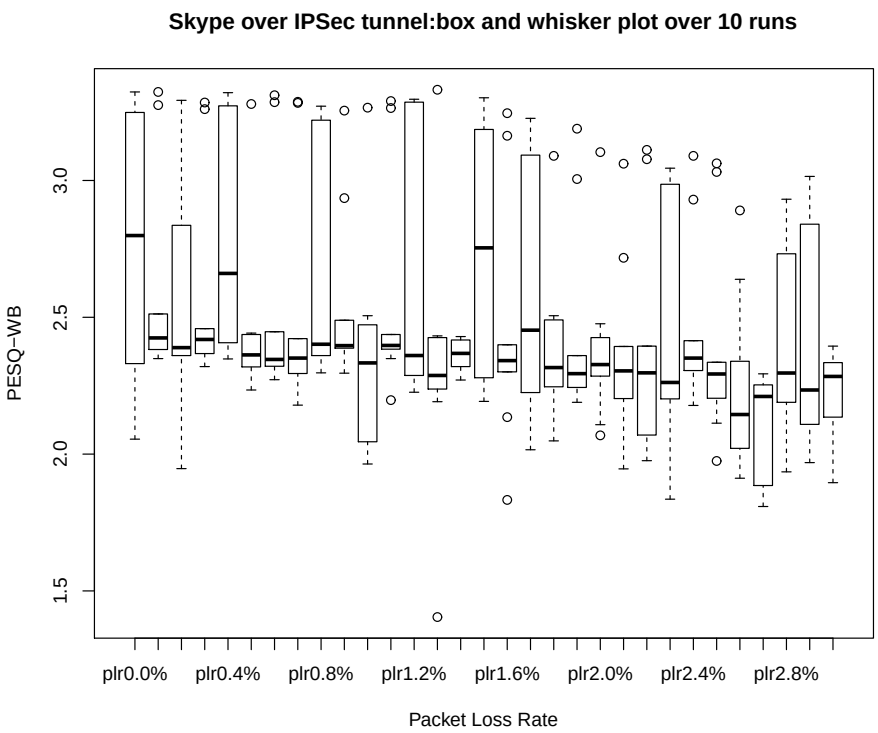


Figure D.15: Graph of Packet Loss Rate versus PESQ-WB

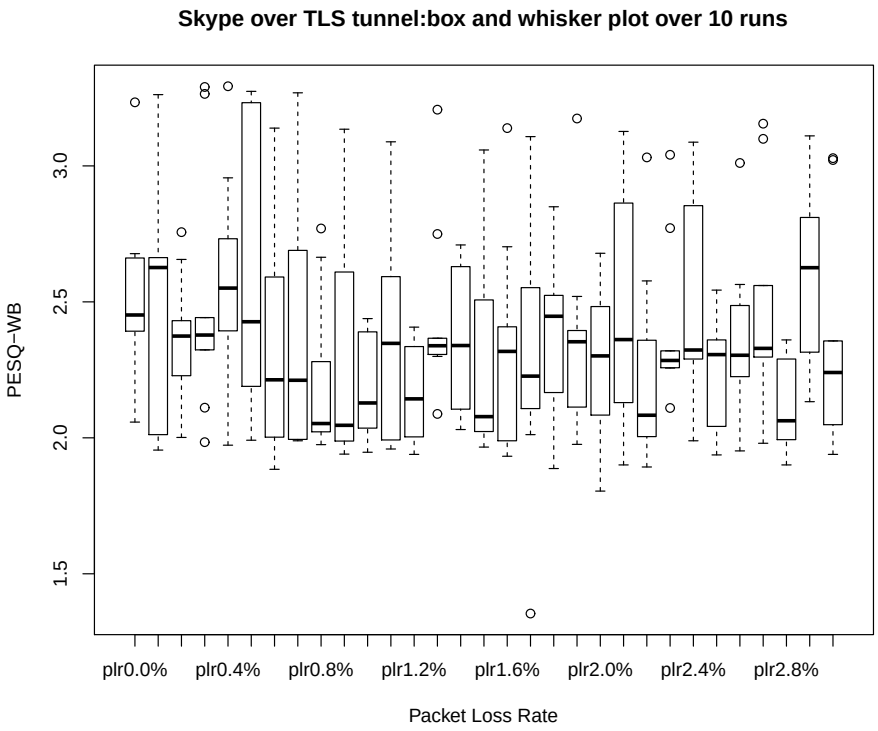


Figure D.16: Graph of Packet Loss Rate versus PESQ-WB

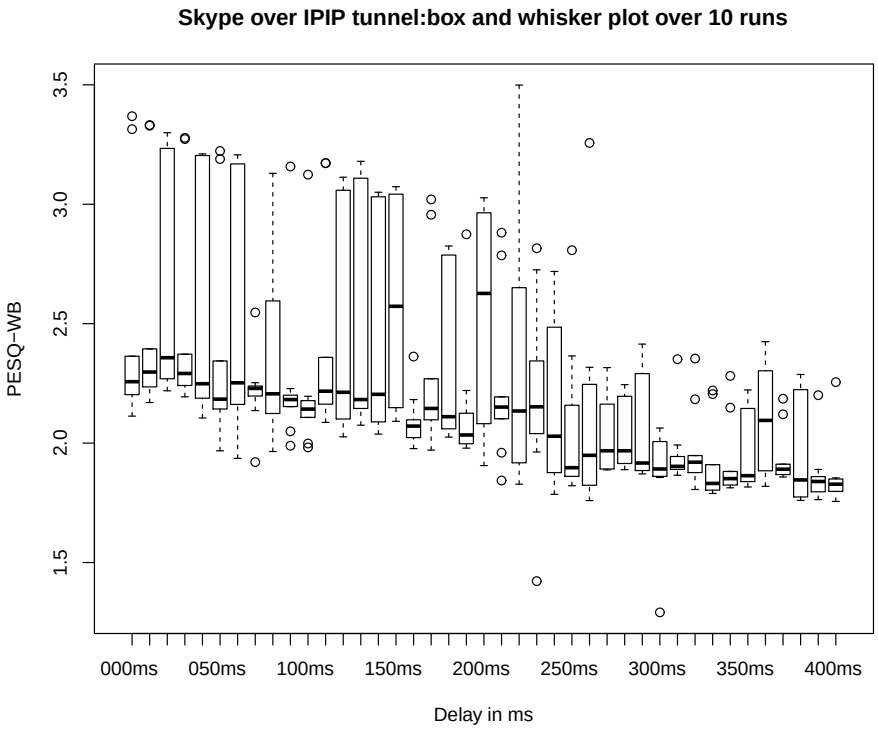


Figure D.17: Graph of Delay versus PESQ-WB

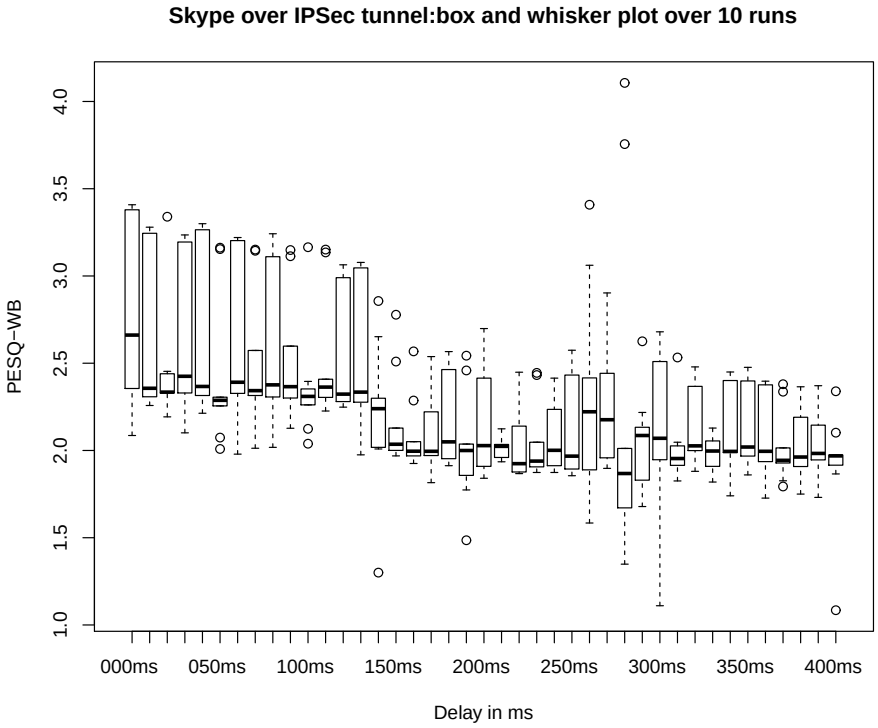


Figure D.18: Graph of Delay versus PESQ-WB

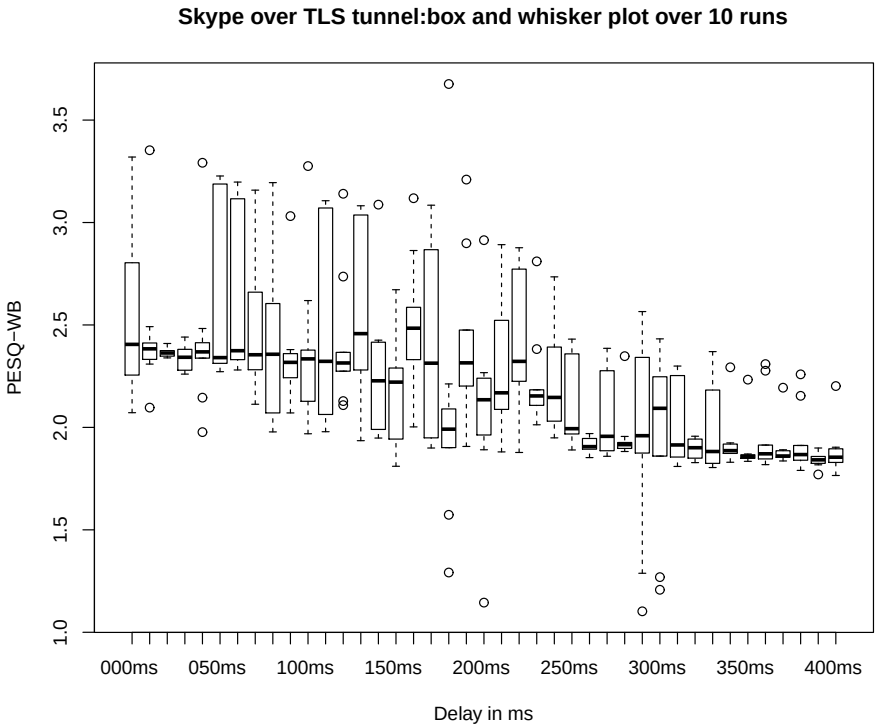


Figure D.19: Graph of Delay versus PESQ-WB

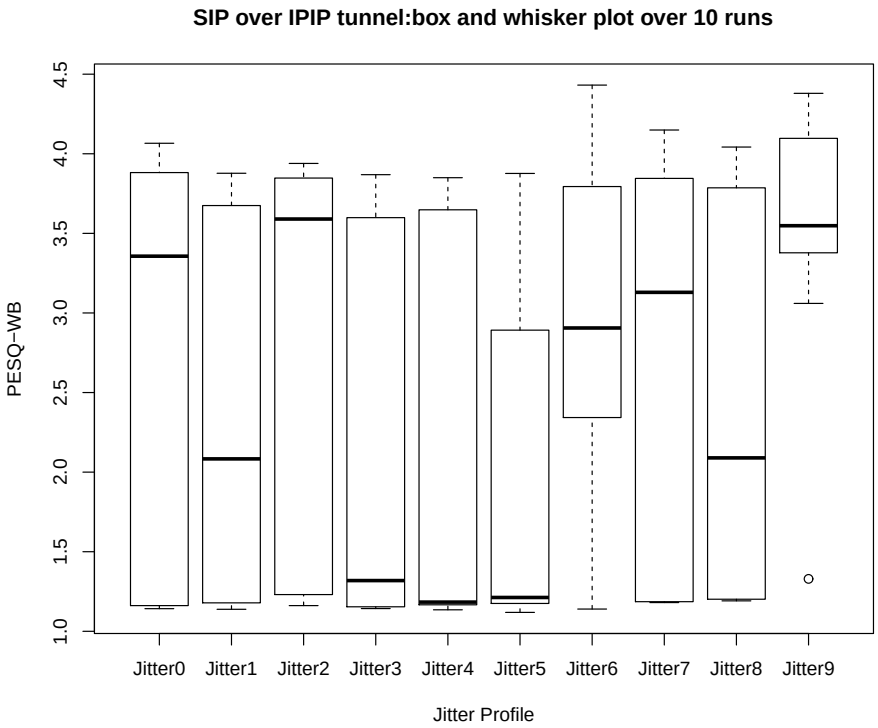


Figure D.20: Graph of Jitter versus PESQ-WB

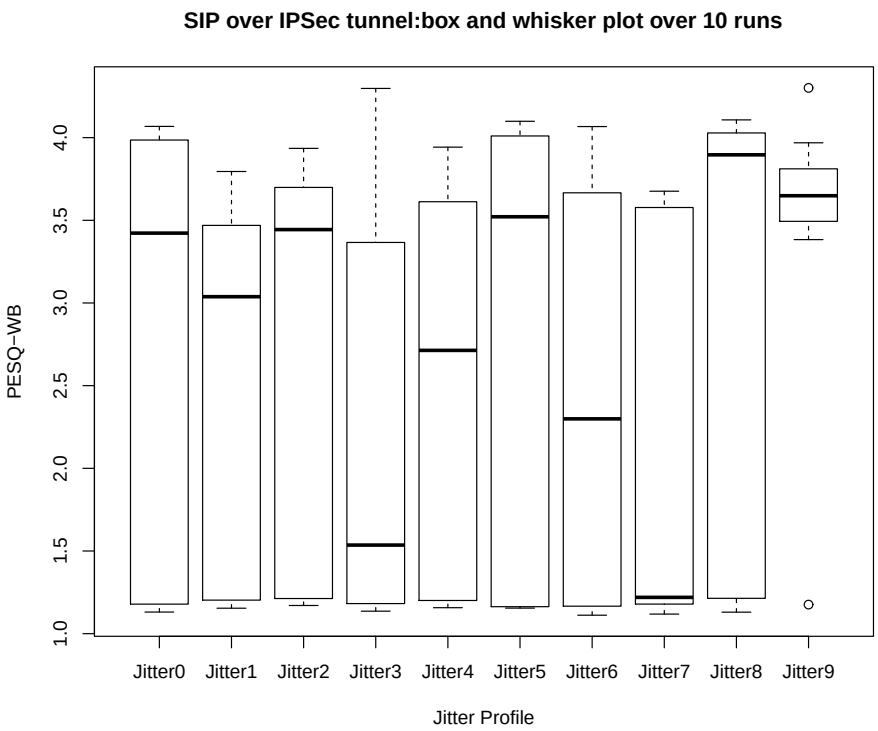


Figure D.21: Graph of Jitter versus PESQ-WB

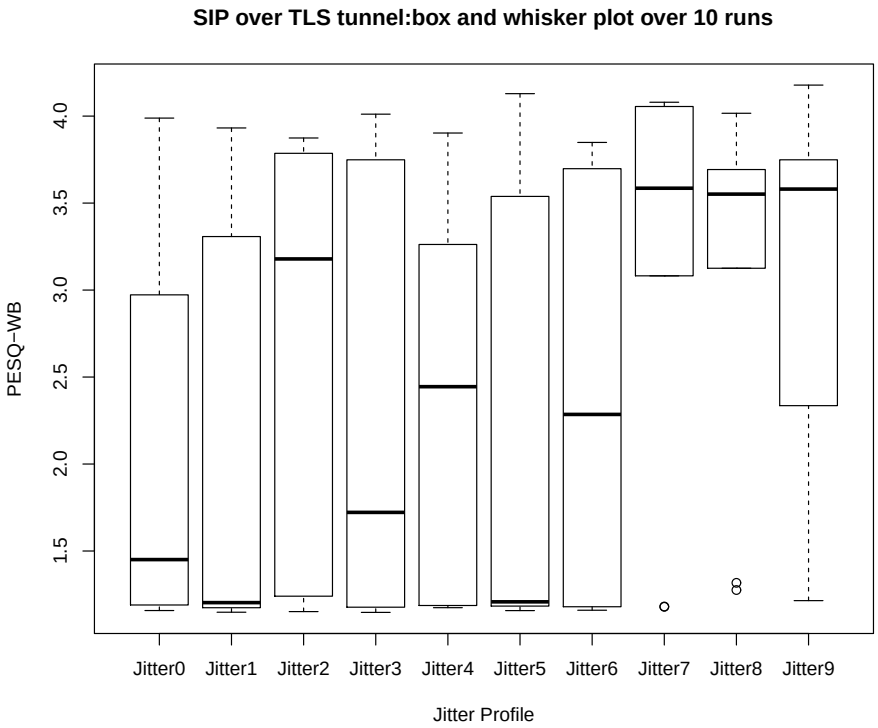


Figure D.22: Graph of Jitter versus PESQ-WB

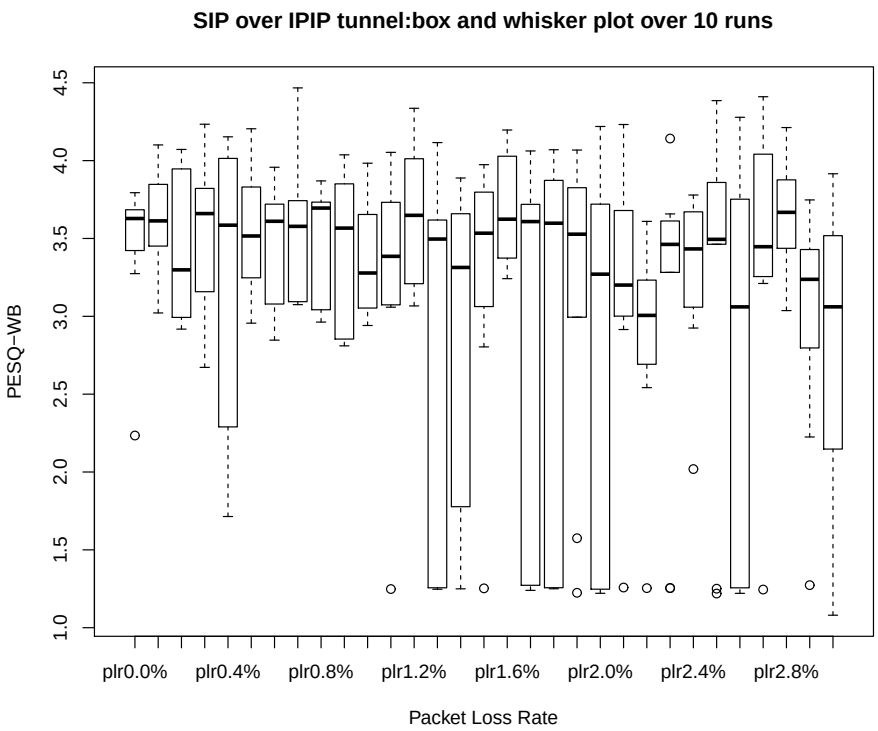


Figure D.23: Graph of Packet Loss Rate versus PESQ-WB

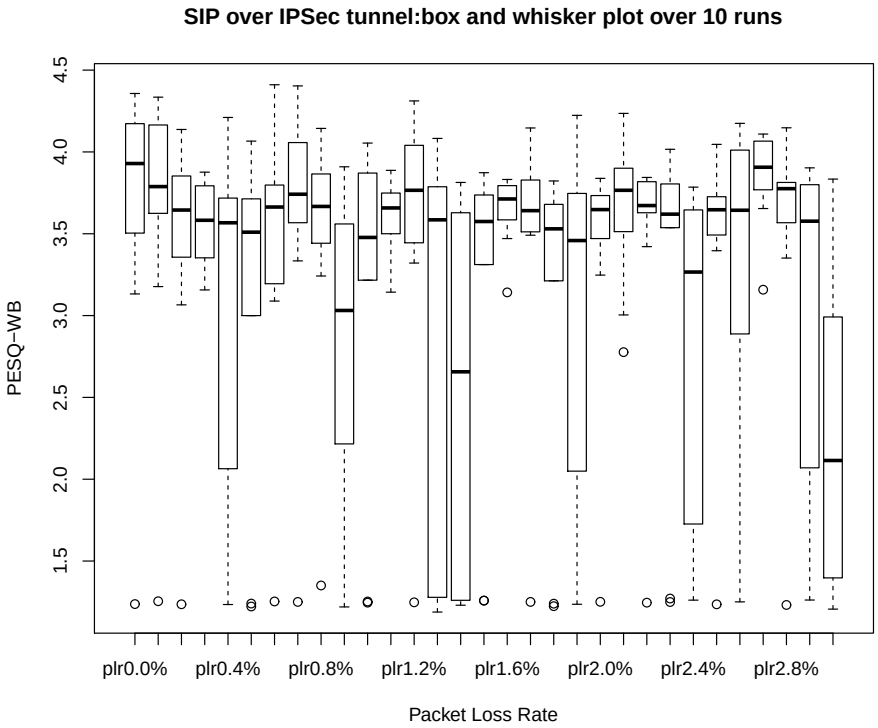


Figure D.24: Graph of Packet Loss Rate versus PESQ-WB

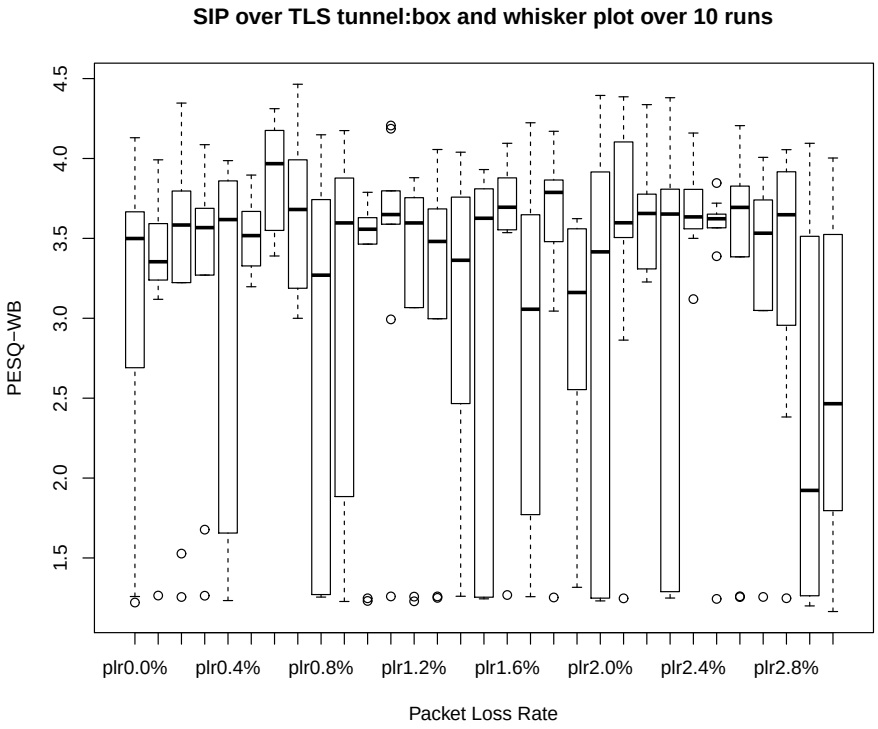


Figure D.25: Graph of Packet Loss Rate versus PESQ-WB

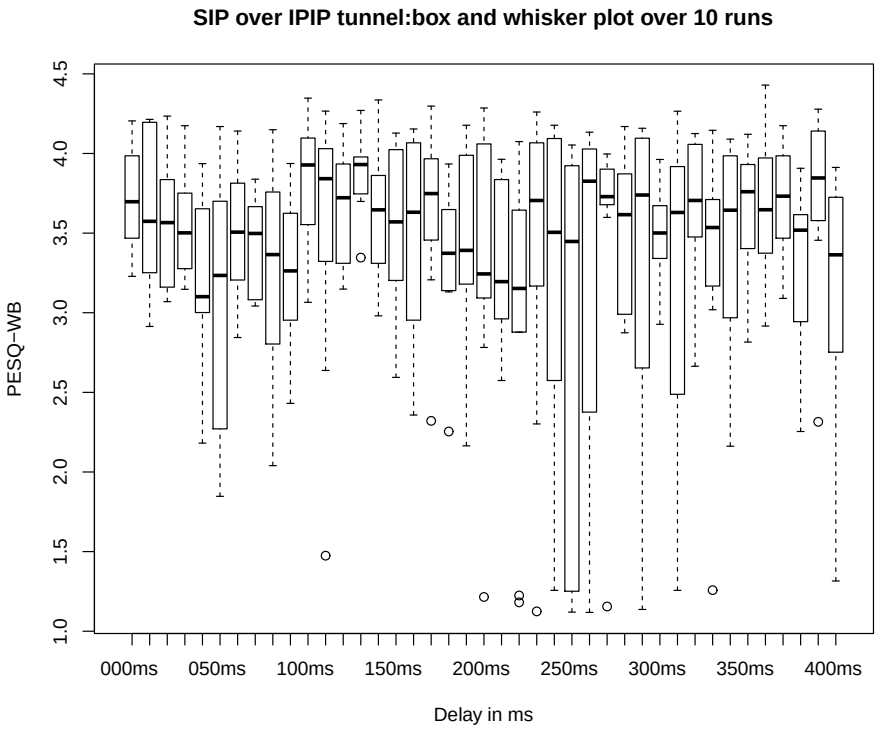


Figure D.26: Graph of Delay versus PESQ-WB

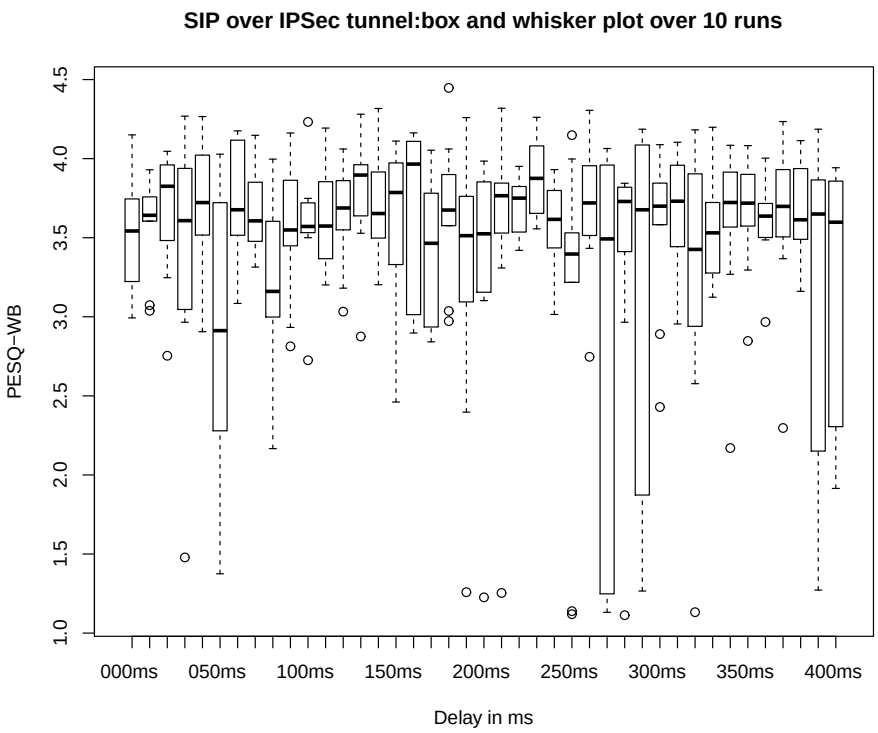


Figure D.27: Graph of Delay versus PESQ-WB

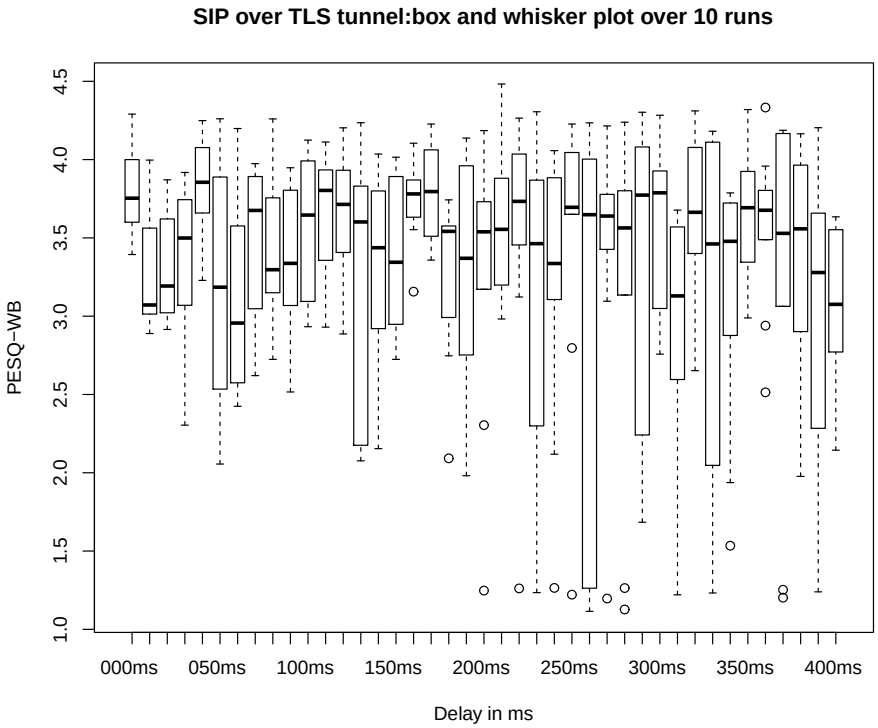


Figure D.28: Graph of Delay versus PESQ-WB

Appendix E

E.1 Framework

The aim of this framework is to provide a tool that allow us to monitor and measure VoIP performance as follows:

i) The framework consists of a live testbed using a wired network with instrumentation data being polled in real time.

ii) The network is designed in such a way that the system and design are in full control and yet still transmit over the Internet.

iii) A packet interceptor is used to simulate various testing conditions.

iv) The research is conducted on different types of VoIP services.

v) The processes involve are listed in a systematic way.

vi) The overall performance score is based on the scores on MOS, R-factor and CVSS Based Metrics.

The testbed designed is shown in figure E.1 on page lxxxv. R1, R2 and R3 are the gateway routers to the Loughborough University network. R1-Secure and R2-Secure are the routers that are placed in the laboratory. R1-Secure is connected to R1 and R2-Secure is connected to R2, respectively. A secure tunnel (i.e. IPSec) is created from R1-Secure to R2-Secure. The sender and monitoring devices are connected to a switch that connect to R1-Secure. Similarly, the receiver and monitoring devices are connected to a switch that connect to R2-Secure. The hybrid testbed can be used for other types of VoIP services. The testbed is not limited by time and place. It is possible to place Router R3 at any designated place. However, to simplify the testbed and to have more control on the network and parameters, a packet interceptor to simulate various conditions and distances is used. This architecture gives more control of the testbed and create a more realistic environment for the experiment. For example to simulate

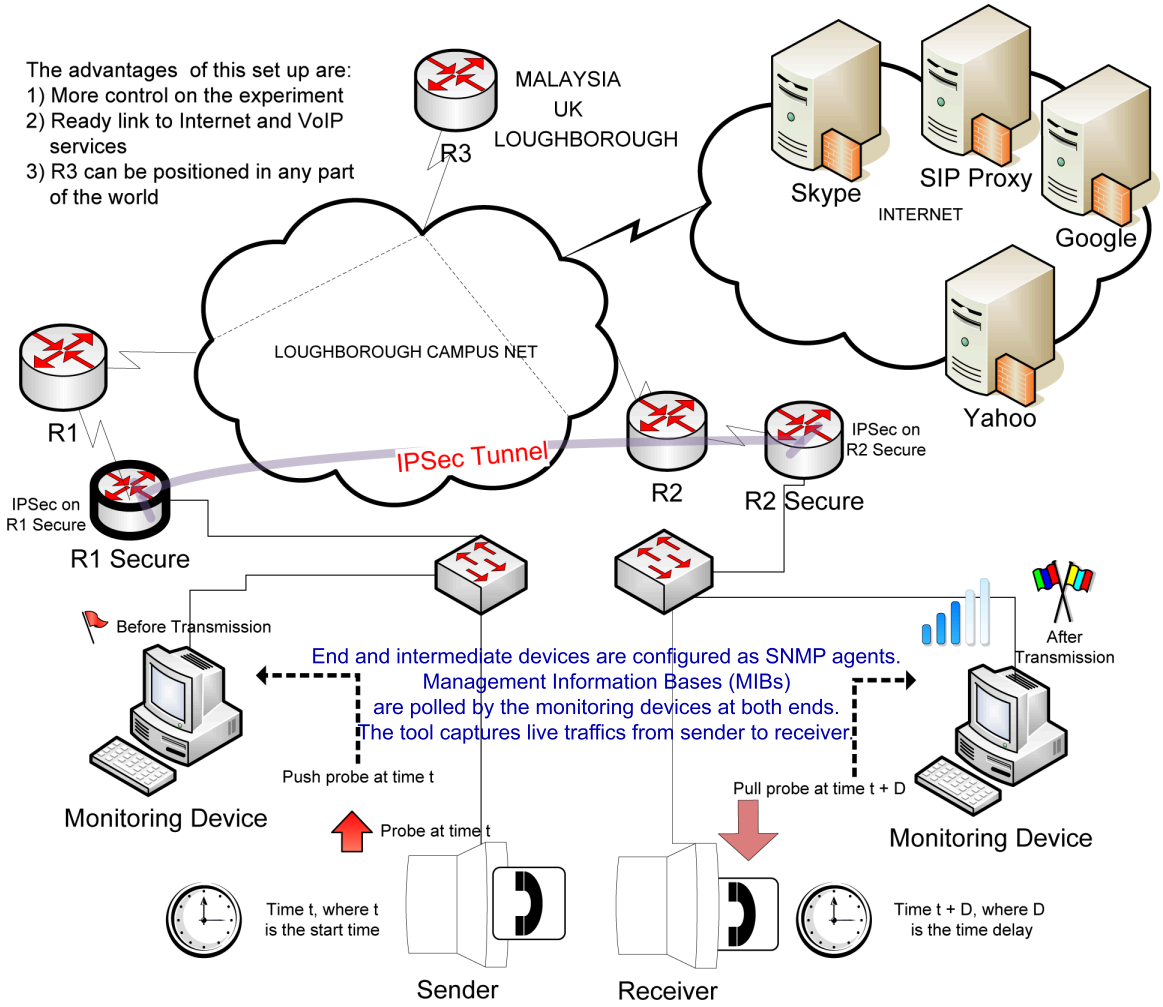


Figure E.1: Testbed Architecture

the distance between the sender and receiver from United Kingdom to Malaysia one can get the RTT value through ICMP then set the extra delay value into a packet interceptor.

In the design we include a link to the Internet as SIP Proxy, Skype server, YMSG server and Google Talk Server are in the external network. The aim is to be able to instrument the VoIP call performance in use and yet still make use of Internet-wide directory services. As each VoIP service has a different mechanism for NAT- and firewall-traversal unfirewalled public IP addresses are used for these experiments. Next the packet interceptor is placed between the sender PC and switch-to-R1-secure. Dummynet is used to simulate bandwidth contention with data packets[141]. Quality of service (QoS) scenario is simulated using this tool as well.

Wireshark is installed into the monitoring device at sender/receiver end. Wireshark is a packet analyser that can capture and interpret live data as it flows across a network[148]. Wireshark is used to analyse the existence of VoIP traffic within the network. The performance monitoring tool CACTI is installed at the receiver/sender end. Section 4.2 on page 92 describes the installation process in detail. Wireshark is used to prove that VoIP call is made and CACTI is used to capture the transmission from sender to receiver. CACTI is used to capture live traffic from sender to receiver and plot graphs. Data collected are presented as graphs of one-way delay or RTT delay, jitter, packet loss, available CPU, memory, buffer size of the end devices and bandwidth utilisation at gateway (i.e. R1 and R2). From these data the R factor of each VoIP service can be calculated. The network trend of each VoIP service can be deduced from these data. The results are verified against other related work from other researchers[134, 25].