Performance of Voice & Video over IP using various IP Transition Mechanisms

Mohib A. SHAH

A thesis submitted in partial fulfilment of the requirements for the degree in Master of Computing

UNITEC Institute of Technology, New Zealand
Abstract

The purpose of this study was to evaluate the performance of voice over IP and video over IP on IP transition mechanisms and clarify the impact of each IP transition mechanism on voice and video transmission under different environments. Studies conducted predicted that IPv4 will soon run out of IP addresses and IPv6 will be the future communication protocol of choice. However, IPv6 still has a challenge ahead, since it does not communicate with IPv4 directly. This issue needs to be resolved prior to establishing IPv6 networks. The migration from IPv4 to IPv6 will take several years, as it is highly complex and expensive for all users around the globe to make a switch. Researchers have developed several methods called IP Translation, Tunnelling and Dual-Stack mechanisms, which allow IPv4 and IPv6 to communicate with each other.

In this study, performance of voice and video was measured and analysed on different IP transition mechanisms on a network environment created in a laboratory. This study was carried out in three parts. First part includes; VoIP performance on three IP transition mechanisms using five different platforms. Second part relates to performance comparison of VoIP over pure IP version 4 and IP version 6 with IP transition mechanisms using five voice CODECS. Third part includes; impact of IP transition mechanisms on video protocols. Main focus of this research was to identify the impact caused by IP transition mechanisms on VoIP and video over IP.

The results obtained for VoIP showed that performance of VoIP on Windows 7 OS using the three IP transition mechanisms performed much better as compared to the other four operating systems. Observation of packet-loss indicated that Windows based OSs had higher packet loss while Linux based OSs had lesser packet loss over all five CODECS for VoIP trials. Results compiled for delay indicated that IPv6-to-4 and IPv6-in-4 marginally performed better than Dual-Stack mechanism. Video over IP transition mechanisms confirmed that video protocols were highly impacted by encapsulation and de-capsulation process except where FLV protocol was used. FLV was the least impacted by IP transition mechanisms. It also indicated that using IPv6-to-4 and IPv6-in-4 tunnelling mechanisms caused more bandwidth wastage than Dual-Stack mechanism.
Acknowledgements

First of all I would like to acknowledge appreciation of the postgraduate programme leaders Donald Joyce and Guan Hong. While the former helped in the initiation of this study the later has helped in providing the needed support. I would also like to thank and place on record the guidance and support provided throughout this study by my Principal Supervisor, Hira Sathu and Associate Supervisor Kathiravelu, Ganeshen. People mentioned above have given me their valuable time and advice that has kept me focused and guided me in the right direction leading to the completion of my thesis and in turn the master’s degree. Without their guidance I would not have achieved the desired result.

Secondly, I would like to thank, all the lecturers in Department of Computing at Unitec who have provided me the background knowledge during course work. Their experience and knowledge have helped and empowered me to the level where I was able to undertake this study to a further level.

I would also like to thank all the class mates (seniors and juniors) in Department of Computing at Unitec, who have supported me directly or indirectly throughout my studies at Unitec.

Finally, I would like to appreciate and thank my family members who have facilitated and provided me this opportunity and support during this thesis as well as other requirements to fulfil my master’s degree.
# Table of Contents

Abstract ........................................................................................................................... i  
Acknowledgements ......................................................................................................... ii  
List of Figures ................................................................................................................. viii  
List of Tables ................................................................................................................ x  
Glossary List ................................................................................................................ xi  
1. Chapter 1: Introduction ......................................................................................... 1  
   1.1. Organisation of This Thesis ........................................................................... 3  
   1.2. Motivation for This Study ............................................................................. 4  
   1.2.1. Published work and material from this thesis .............................................. 5  
2. Chapter 2: Literature Review ............................................................................... 6  
   2.1. VoIP (Voice over Internet Protocol) ............................................................... 6  
   2.2. VVoIP (Video & Voice over Internet Protocol) ................................................ 8  
   2.3. IPv4 (Internet Protocol version 4) ................................................................. 9  
   2.3.1. IPv4 limitations .......................................................................................... 10  
   2.4. IPv6 (Internet Protocol version 6) ................................................................. 11  
   2.4.1. Comparison between IPv4 and IPv6 .......................................................... 12  
   2.4.2. IPv6 limitations .......................................................................................... 13  
   2.5. IP Transition Mechanisms .......................................................................... 14  
   2.5.1. IPv6-to-4 .................................................................................................... 14  
   2.5.2. IPv6-in-4 .................................................................................................... 15  
   2.5.3. DSTM ........................................................................................................ 16  
   2.5.4. Teredo ........................................................................................................ 18  
   2.5.5. NAT-PT ..................................................................................................... 19  
   2.6. Voice over IP Protocols .............................................................................. 21  
   2.6.1. SIP ............................................................................................................. 21  
   2.6.2. RTP ............................................................................................................ 21  
   2.6.3. H.323 ......................................................................................................... 22  
   2.7. Voice over IP CODECS ............................................................................... 22  
   2.7.1. G.711 .......................................................................................................... 22  
   2.7.2. G.723 .......................................................................................................... 23  
   2.7.3. G.729 .......................................................................................................... 23  
   2.8. VVoIP (Video & Voice over Internet Protocol) Protocols .............................. 25
5.4.3. Packet loss for G.723.1 codec over IP transition mechanisms .......... 94
5.4.4. Packet loss for G.729.2 codec over IP transition mechanisms .......... 95
5.4.5. Packet loss for G.729.3 codec over IP transition mechanisms .......... 96
5.5. Results for Throughput............................................................................. 98
  5.5.1. Throughput for all 5 CODECS over IP transition mechanisms .......... 98
5.6. Comparison of VoIP on IP versions and IP Transition Mechanisms ......... 100
5.7. Results for Delay..................................................................................... 101
  5.7.1. Delay for G.711.1 codec using IP versions & IPTMs ...................... 101
  5.7.2. Delay for G.711.2 codec using IP versions & IPTMs ...................... 102
  5.7.3. Delay for G.723.1 codec using IP versions & IPTMs ...................... 103
  5.7.4. Delay for G.729.2 codec using IP versions & IPTMs ...................... 105
  5.7.5. Delay for G.729.3 codec using IP versions & IPTMs ...................... 106
5.8. Results for Jitter..................................................................................... 108
  5.8.1. Jitter for G.711.1 codec using IP versions & IPTMs ...................... 108
  5.8.2. Jitter for G.711.2 codec using IP versions & IPTMs ...................... 109
  5.8.3. Jitter for G.723.1 codec using IP versions & IPTMs ...................... 111
  5.8.4. Jitter for G.729.2 codec using IP versions & IPTMs ...................... 112
  5.8.5. Jitter for G.729.3 codec using IP versions & IPTMs ...................... 114
5.9. Results for Packet loss........................................................................... 115
  5.9.1. Packet loss for G.711.1 codec over IP versions & IPTMs .............. 116
  5.9.2. Packet loss for G.711.2 codec over IP versions & IPTMs .............. 117
  5.9.3. Packet loss for G.723.1 codec over IP versions & IPTMs .............. 118
  5.9.4. Packet loss for G.729.2 codec over IP versions & IPTMs .............. 120
  5.9.5. Packet loss for G.729.3 codec over IP versions & IPTMs .............. 121
5.10. Results for Throughput......................................................................... 123
  5.10.1. Throughput of 5 CODECS on IP versions & IP transition mechanisms 123
5.11. Performance of VVoIP & Impact of IPTMs on Video Protocols ............ 125
  5.12.1. MPEG-1 protocol......................................................................... 126
  5.12.2. MPEG-2 protocol......................................................................... 127
  5.12.3. MPEG-4 protocol......................................................................... 128
  5.12.4. MKV protocol............................................................................. 130
  5.12.5. FLV protocol............................................................................. 131
5.13. Actual-throughput versus Impacted-throughput.................................. 132
## List of Figures

- Figure 2-1: End-to-end components of VoIP [17] ............................................................ 7
- Figure 2-2: The population of IPv4 internet hosts around the world [29] ....................... 11
- Figure 2-3: Architecture of IPv6to4 tunnel [35] .............................................................. 14
- Figure 2-4: IPv6 hosts connected to other IPv6 hosts via 6to4 tunnel ........................... 15
- Figure 2-5: Network based on IPv6-in-4 tunnel between IPv6-capable hosts ............... 16
- Figure 2-6: Network based on Dual-Stack transition mechanism .............................. 17
- Figure 2-7: An example of a Teredo based network [43] ............................................. 18
- Figure 2-8: Network based on NAT-PT ......................................................................... 19
- Figure 2-9: NAT-PT translation processing system [48] ............................................. 20
- Figure 2-10: Architecture of D-ITG and its suite [77] .................................................. 31
- Figure 2-11: Network test-bed for SIP based VoIP [81] ............................................. 36
- Figure 2-12: Flowchart of ASDM-TS method [82] ..................................................... 39
- Figure 2-13: The results of LAB, MAB and HAB [82] ................................................ 40
- Figure 2-14: Network diagram based on four scenarios [83] ...................................... 42
- Figure 2-15: Network setup [84] .................................................................................. 45
- Figure 2-16: Network design for a two-way video conference [85] ......................... 48
- Figure 2-17: Available routes of Dual-Stack mechanism over the internet [85] .......... 48
- Figure 2-18: Network architecture from Korea to China using IPv6 [85] ............... 50
- Figure 3-1: Water-fall methodology [92] ...................................................................... 59
- Figure 3-2: An example of D-ITG output .................................................................... 62
- Figure 3-3: An example of bar chart .......................................................................... 63
- Figure 3-4: Network performance print-screen ......................................................... 64
- Figure 3-5: An example of video data analysis ............................................................ 65
- Figure 3-6: An example of area chart used for results ............................................... 65
- Figure 4-1: Network design ....................................................................................... 67
- Figure 5-1: Delay on five platforms using G.711.1 codec ........................................ 79
- Figure 5-2: Delay on five platforms using G.711.2 codec ........................................ 80
- Figure 5-3: Delay on five platforms using G.723.1 codec ........................................ 82
- Figure 5-4: Delay on five platforms using G.729.2 codec ........................................ 83
- Figure 5-5: Delay on five platforms using G.729.3 codec ........................................ 84
- Figure 5-6: Jitter on five platforms using G.711.1 codec .......................................... 86
- Figure 5-7: Jitter on five platforms using G.711.2 codec .......................................... 87
- Figure 5-8: Jitter on five platforms using G.723.1 codec .......................................... 88
- Figure 5-9: Jitter on five platforms using G.729.2 codec .......................................... 89
- Figure 5-10: Jitter on five platforms using G.729.3 codec ......................................... 90
- Figure 5-11: Packet loss on five platforms using G.711.1 codec ............................ 92
- Figure 5-12: Packet loss on five platforms using G.711.2 codec ............................ 93
List of Tables

Table 2-1: Video services over the internet and bandwidth required [23] ......................... 8
Table 2-2: Category of IPv4 addresses [26] ..................................................................... 10
Table 2-3: IP version 4 architecture [27] ........................................................................ 10
Table 2-4: IPv6 architecture [30] .................................................................................. 12
Table 2-5: IPv4 versus IPv6 [34] .................................................................................. 13
Table 2-6: Teredo tunnel’s address structure [43] ............................................................ 18
Table 2-7: VoIP CODECS specification [62] .................................................................. 24
Table 2-8: VoIP CODECS supported by D-ITG & their specification [76] ....................... 30
Table 2-9: All scenarios tested and their results [81] ....................................................... 37
Table 2-10: Simulation cases [84] .................................................................................. 46
Table 2-11: The detail of standard definition TV and high definition TV [85] ............... 49
Table 2-12: Summary of related studies ......................................................................... 52
Table 4-1: Hardware details ......................................................................................... 69
Table 4-2: Software details ......................................................................................... 70
Table 5-1: Throughput results for all 5 CODECS on five operating systems ............... 98
Table 5-2: Throughput results for five VoIP CODECS ................................................. 123
Table 5-3: CPU utilisation results for video protocols ................................................. 136
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM</td>
<td>Association for Computing Machinery</td>
</tr>
<tr>
<td>ATM</td>
<td>Asynchronous Transfer Mode</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>D-ITG</td>
<td>Distributed Internet Traffic Generator (tool)</td>
</tr>
<tr>
<td>DSTM</td>
<td>Dual Stack Transition Mechanism</td>
</tr>
<tr>
<td>DVD</td>
<td>Digital Versatile Disc</td>
</tr>
<tr>
<td>F4V</td>
<td>Flash MP4 Video</td>
</tr>
<tr>
<td>FBC</td>
<td>Frame Based Classification</td>
</tr>
<tr>
<td>FEC</td>
<td>Forward Error Correction</td>
</tr>
<tr>
<td>FLV</td>
<td>Flash Video</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HAB</td>
<td>High Available Bandwidth</td>
</tr>
<tr>
<td>HDTV</td>
<td>High Definition Television</td>
</tr>
<tr>
<td>HE-AAC</td>
<td>High-Efficiency Advanced Audio Coding</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPPP</td>
<td>IPPP video format</td>
</tr>
<tr>
<td>IPSec</td>
<td>Internet Protocol Security</td>
</tr>
<tr>
<td>IPTM</td>
<td>IP Transition Mechanism</td>
</tr>
<tr>
<td>IPv4</td>
<td>Internet Protocol version 4</td>
</tr>
<tr>
<td>IPv6</td>
<td>Internet Protocol version 6</td>
</tr>
<tr>
<td>IPv6-in-4(6in4)</td>
<td>Internet Protocol version 6 in 4 Transition Mechanism</td>
</tr>
<tr>
<td>IPv6-to-4(6to4)</td>
<td>Internet Protocol version 6 to 4 Transition Mechanism</td>
</tr>
<tr>
<td>ITU</td>
<td>International Telecommunication Union</td>
</tr>
<tr>
<td>ITU-T</td>
<td>International Telecommunication Union-Telecommunication</td>
</tr>
<tr>
<td>Kbps</td>
<td>Kilo bits per second</td>
</tr>
<tr>
<td>KiB/s</td>
<td>Kilo bytes per second</td>
</tr>
<tr>
<td>LAB</td>
<td>Low Available Bandwidth</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>MAB</td>
<td>Medium Available Bandwidth</td>
</tr>
<tr>
<td>Mbps</td>
<td>Mega bits per second</td>
</tr>
<tr>
<td>MCUs</td>
<td>Multipoint Control Units</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Megaco</td>
<td>Media Gateway Control Protocol</td>
</tr>
<tr>
<td>MGEN</td>
<td>Multi-Generator (tool)</td>
</tr>
<tr>
<td>MKV</td>
<td>Matroska Multimedia Container</td>
</tr>
<tr>
<td>MM</td>
<td>Multimedia</td>
</tr>
<tr>
<td>MPEG-1, -2, -4</td>
<td>Moving Picture Experts Group 1, 2 &amp; 4</td>
</tr>
<tr>
<td>MOS</td>
<td>Mean Opinion Score</td>
</tr>
<tr>
<td>ms</td>
<td>Milliseconds</td>
</tr>
<tr>
<td>MSP</td>
<td>Mini SIP Proxy</td>
</tr>
<tr>
<td>NAT</td>
<td>Network Address Translation</td>
</tr>
<tr>
<td>NAT-PT</td>
<td>Network Address Translation - Protocol Translation</td>
</tr>
<tr>
<td>NGN</td>
<td>Next Generation Network</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>NS-2</td>
<td>Network Simulator version 2 (tool)</td>
</tr>
<tr>
<td>NTP</td>
<td>Network Time Protocol</td>
</tr>
<tr>
<td>OS or OSs</td>
<td>Operating System or Operating Systems</td>
</tr>
<tr>
<td>PSTN</td>
<td>Public Switched Telephone Network</td>
</tr>
<tr>
<td>Pure IPv4</td>
<td>(It means networks fully based on IPv4 infrastructure)</td>
</tr>
<tr>
<td>Pure IPv6</td>
<td>(It means networks fully based on IPv6 infrastructure)</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Services</td>
</tr>
<tr>
<td>RTP</td>
<td>Real-time Transport Protocol</td>
</tr>
<tr>
<td>RTT</td>
<td>Round Trip Time</td>
</tr>
<tr>
<td>SER</td>
<td>SIP Express Router</td>
</tr>
<tr>
<td>SIP</td>
<td>Session Initiation Protocol</td>
</tr>
<tr>
<td>SIPp</td>
<td>SIP Protocol (tool)</td>
</tr>
<tr>
<td>SNT</td>
<td>Simple Network Tester (tool)</td>
</tr>
<tr>
<td>SWF</td>
<td>Shockwave Flash</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>UAs</td>
<td>User Agents</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>VLC</td>
<td>Video LAN Client (tool)</td>
</tr>
<tr>
<td>VLSM</td>
<td>Variable Length Subnet Mask</td>
</tr>
<tr>
<td>VoIP</td>
<td>Voice over Internet Protocol</td>
</tr>
<tr>
<td>VVoIP</td>
<td>Video and Voice over Internet Protocol</td>
</tr>
<tr>
<td>WAN</td>
<td>Wide Area Network</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
<tr>
<td>xDSL</td>
<td>Anything: Digital Subscriber Line</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>.MK3D</td>
<td>Matroska 3D</td>
</tr>
<tr>
<td>.MP4</td>
<td>MPEG-4</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

In the past few years VoIP (Voice over Internet Protocol) has become one of the fastest growing concepts in the communication world. VoIP has changed the traditional way of voice communication and has established a contemporary infrastructure for voice transmission. PSTN (Public Switched Telephone Network) system was established many years ago to carry analogue voice over circuit switched conventional telephone cable [01]. However, VoIP has changed this concept as it converts analogue signals to digital packets and transmits them over the internet and converts them back to analogue at the other end. Fundamental concept of VoIP is not so new; however due to the limitations of fast internet connections availability, VoIP technology was restricted until broadband internet connections were established [02]. These days’ broadband connections have enabled users to have VoIP communication on IPv4 infrastructures but a new challenge has risen for VoIP as IPv4 is running out of IP addresses.

Video over IP is another significant and innovative technology, which is rapidly growing everyday since broadband connections were available publicly. Trend of using video over IP conferences is rapidly increasing these days and researchers have developed new video protocols to enhance the ability of transmission of video packets over internet. However, as numbers of users increase, this technology also relies on IPv4 based addresses and as cautioned by related research [3, 4 & 5] IPv4 is expected to be out of IP addresses in the next few years.

To resolve IPv4 shortage, IETF (Internet Engineering Task Force) has developed new version of Internet Protocol known as IPv6. IPv6 has the ability to host a lot more IP addresses than IPv4 ($2^{128}$) and it also has new features to improve the performance of the internet [6]. IPv6 and the new features it offers are still not compatible across all internet users, as it does not communicate with IPv4 directly, which means it will create a cloud of its own apart from IPv4, forcing internet users to replace their existing infrastructure to IPv6. According to researchers, migrating from IPv4 to IPv6 is time consuming and this process will take years to complete because it is highly complex
and expensive as well. “The transition between IPv4 internet and IPv6 will be a long process as they are two completely separate protocols and it is impossible to switch the entire internet over to IPv6 over night” [7].

To resolve the issue different types of IP transition mechanisms will be used during the migration period. Broadly there are three types of IP transition mechanisms that have been developed such as Tunnelling, Translation and Dual-Stack mechanisms. These mechanisms have the capability to enable IPv4 and IPv6 based networks to communicate with each other. However, these IP transition mechanisms raise different quality concerns for VoIP and video communication when both voice & video are transmitted over networks using these IP transition mechanisms.

To clarify the different related issues on this subject area, the author initiated this research to identify the performance of VoIP (Voice over IP) and video over IP on IP transition mechanisms using multiple platforms. Metrics covered in the experiment are jitter, delay, throughput, actual-throughput, impacted-throughput, packet loss and CPU utilisation using three well known IP transition mechanisms such as IPv6-to-4, IPv6-in-4 and Dual-Stack. More specification and depth will be covered in this thesis. Next section presents organisation of this thesis.
1.1. Organisation of This Thesis

This thesis includes altogether seven chapters and describes step by step progression of the study through each chapter. First chapter covers introduction and acknowledgement. It briefly mentions voice and video over IPv4 and the ensuing dilemma. Currently transmission of voice and video over IP is fully based on IPv4 networks; however, in next few years it will have to integrate with IPv6 based networks due to the shortage in IPv4 addresses. According to researchers, it will take several years to migrate from IPv4 to IPv6 and during the migration IP transition mechanisms will emerge to enable IPv4 and IPv6 to communicate to each other. Second chapter includes literature review and describes both versions of IP and different IP transition mechanisms to provide the required background. It also covers related research work and metrics measured in this research. Third chapter describes methodology and techniques used for this research. It also illustrates the procedures of data collection and steps involved in the process of literature review for this research. Fourth chapter covers the network test-bed and illustrates the network configuration. It also mentions software and hardware specification. Fifth chapter covers data analysis and results obtained from this research. It has three different parts; first part includes performance of VoIP on IP transition mechanisms using five different platforms while second part covers performance comparison of VoIP over pure IP version 4 and IP version 6 with IP transition mechanisms using five voice CODECS. Third part includes; impact of IP transition mechanisms on video protocols. Sixth chapter includes the ensuing discussion of the findings, followed by seventh chapter, which covers the conclusion. Next section presents motivation of this study.
1.2. Motivation for This Study

Considering the growth of IPv4 rapidly increasing over the next few years, all available IP addresses in IP version 4 will be exhausted. Even the use of private IP address with NAT and Proxy Servers offer only a limited interim solution. The Internet Engineering Task Force (IETF) has designed IPv6 to resolve the issue, which supports large number of IP addresses. In past few years, experts have carried out the research of voice and video over IP [8, 9, 10, 11 & 12] over IPv4 and IPv6. However, IPv6 lead to another issue, as it cannot communicate with IPv4 directly. To resolve these issues IP transition mechanisms were designed to enable IPv6 and IPv4 users to communicate with each other.

This study investigates the impact of IP transition mechanisms on voice and video traffic. These experiments were deployed based on real machines in a lab environment.

The main objectives covered in this study are:

- The performance of voice over IP (VoIP) was tested on real machines in order to clarify the quality of VoIP on IPv4 and IPv6.

- The IP transition mechanisms were deployed on real machines and the VoIP quality was evaluated and the results were compared with pure IPv4 and pure IPv6. Impact of IPv6to4, IPv6in4 and Dual-Stack mechanisms on voice quality was tested.

- Parameters considered were jitter, throughput, packet loss and delay (RTT) to evaluate the performance of VoIP from different aspects. Five different operating systems were included which acted as soft-routers to identify the behaviour of each router from VoIP perspective.

- Impact of IPv6in4, IPv6to4 and Dual-Stack mechanisms on video quality was tested and results were compared with pure IPv6 based network.
Five different video protocols were tested and the impact caused by IP transition mechanisms on throughput was measured and compared.

1.2.1. Published work and material from this thesis

This research has provided interesting and credible results of voice and video over IP transition mechanisms, which led this research into credible publications in IEEE (Institute of Electrical and Electronics Engineers) and ACM (Association for Computing Machinery). Following is the list of the publications which were accomplished throughout this research thesis.

Published Conference Papers:


Accepted Conference Papers:

- Evaluating Performance Impact of 6to4 & 6in4 Tunnelling Mechanism on VoIP. See reference [15].

Papers under Review:

- Comparative Performance of VoIP on three Linux Servers using IPv4 and IPv6
- Impact of Dual-Stack Transition Mechanism on VoIP quality
Chapter 2: Literature Review

This chapter covers literature reviewed and describes the steps structured to drive this research. Firstly, VoIP (Voice over Internet Protocol) and video over IP are defined and then it leads to the reason for this research. The IP address space issues in IPv4 and its ensuing issues are discussed next. The replacement of existing IPv4 is being made by using another protocol which was designed with enhanced features (IPv6). Even though IPv6 has advanced features, the problem of IPv4 and IPv6 compatibility still remains. Therefore, IP transition mechanisms were designed to aid this. Moreover, having all these changes has led to quality concerns in voice and video transmission. Furthermore this chapter describes a selection of voice protocols, CODECS and video protocols covered in this research. It also mentions the selection of traffic generating and measuring tools to calculate the quality of voice and video over IP transition mechanisms. Finally it discusses the related studies to this study. Next section defines the concept and procedure of VoIP (Voice over Internet Protocol).

2.1. VoIP (Voice over Internet Protocol)

VoIP stands for voice over internet protocol and is also known as IP telephony. VoIP is another way of voice communication apart from PSTN (Public Switched Telephone Network) and cellular system. The concept of VoIP is not so new but due to slow internet connections available publicly, this technology remained in the background. Recently fast internet connections also known as broadband have been available publicly which, enabled users to use voice communication via internet [16].

VoIP is used for the same purpose as PSTN which is calling from one place to another but using a different feature of the network infrastructure. VoIP allows a user to have voice conversation via LANs, WLANs, and WANs with reasonable broadband connection. This would enable voice and data communication using an IP phone set or a soft-phone that converts analogue voice into digital packets. Then those digital
packets flow through internet and reach their destination. On arrival these digital packets get converted into analogue voice at the other end. There are various voice protocols and CODECS, which carry digital packets via internet and deliver them to their destination. (More information on voice CODECS and voice protocols will be covered in protocols section at the end of this chapter). Figure 2-1 below illustrates the procedure of voice transmission via internet.

According to Ding & Radwan et al, 2007 [18] “VoIP has emerged as an important application and it is expected to replace the current Public Switched Telephone Network in next few years”. The main reason VoIP has become so popular over the last few years is because of the reduced cost associated with using VoIP compared to the conventional circuit switched PSTN. Moreover, VoIP communication system is mostly being used over IPv4 infrastructure (Internet); however futuristic research studies of VoIP state that VoIP has greater challenges ahead, when it will be used over IPv6 infrastructure and has to integrate with both IPv4 and IPv6 networks [19]. Next section covers video over IP followed by the dilemma that arises when voice and video traffic is transmitted over IPv4 and IPv6.
2.2. VVoIP (Video & Voice over Internet Protocol)

VVoIP stands for video and voice over Internet Protocol and also known as multimedia over IP. VVoIP is another technology, which is rapidly growing. Latest studies [20, 21 & 22] bring forth that video & voice over IP is a vital technology, which has an essential role in computing world and this technology will grow even faster in the next few years.

VVoIP works in similar way as voice over IP. VVoIP allows users to watch video and listen to audio over the internet. VVoIP would require a broadband connection with more bandwidth as VVoIP transmits larger packets over the internet while VoIP has smaller packet size as compared to VVoIP. For VVoIP a video camera is used to capture the video which is then converted into digital packets. These packets flow via internet and at the other end, digital packets are received that allow a user to listen and watch the captured video on the monitor. There are different types of video protocols which carry video packets through internet and deliver them to their destination. Each video protocol has different features which affects the video quality. This is discussed later in the protocols section at the end of this chapter. Table 2-1 below explains different types of video services with the expected bandwidth to transmit them over the internet.

<table>
<thead>
<tr>
<th>Appliance</th>
<th>Service</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Television</td>
<td>High Definition TV (MPEG2)</td>
<td>~19 Mbps</td>
</tr>
<tr>
<td></td>
<td>Pay TV</td>
<td>3-6 Mbps</td>
</tr>
<tr>
<td></td>
<td>Standard Definition TV (MPEG2)</td>
<td>3.5 Mbps</td>
</tr>
<tr>
<td>Personal Computer</td>
<td>Interactive TV on Internet</td>
<td>1-3.5 Mbps</td>
</tr>
<tr>
<td></td>
<td>Video on Demand (VoD)</td>
<td>3-6 Mbps</td>
</tr>
<tr>
<td></td>
<td>Personal Video Recorder</td>
<td>Up to 6 Mbps</td>
</tr>
<tr>
<td></td>
<td>High Speed Internet (WEB Surfing)</td>
<td>Up to 2 Mbps</td>
</tr>
<tr>
<td></td>
<td>Interactive Gaming</td>
<td>1-5 Mbps</td>
</tr>
<tr>
<td></td>
<td>Video on PC</td>
<td>4-12 Mbps</td>
</tr>
<tr>
<td>Telephone</td>
<td>Voice over IP (VoIP)</td>
<td>20-64 kbps</td>
</tr>
<tr>
<td></td>
<td>Voice over DSL (VoDSL)</td>
<td>40-64 kbps/chn</td>
</tr>
</tbody>
</table>

Table 2-1: Video services over the internet and bandwidth required [23]
As shown in the Table 2-1 above, there are different types of video services, which require different amount of bandwidth while voice over IP requires lesser bandwidth than video over IP. Futuristic studies also indicate that availability, accessibility and reliability of VVoIP on all types of electronic devices will be on demand. Therefore, VVoIP would require more IP addresses in order to permit larger numbers of devices to be connected over the internet. Several other concerns are expected to arise and video over IP has to deal with related issues, in order to enhance the performance of video over IP. Issues like video packet size for mobile devices and quality over next generation networks (NGN) are yet to be resolved.

Currently video over IP is mainly being transmitted over IPv4 networks (Internet). However, according to researchers a greater challenge exists for transmitting video over IP using IPv6 infrastructure. This leads to the same scenario as VoIP where both technologies have to perform on IPv4 and IPv6 networks [24]. Next section covers issues that arise when voice and video traffic is transmitted over IPv4 and IPv6.

2.3. **IPv4 (Internet Protocol version 4)**

The growth of internet users around the globe is rapidly increasing in most of the countries. Currently the Internet Protocol version 4 is mostly being used to communicate over the internet world. Usually an IP address holds two different types of data such as host address and the network address. IPv4 address was structured based on 32-bit values and normally expressed in dotted decimal notation with 4 octets separated by decimals, for example 192.168.120.80. IPv4 addresses were divided into five different classes; however, class A, B & C is generally used. Class A provides highest amount of IP addresses, while class B provides less than class A and class C provides least amount of IP addresses. The following is an example of the IP address classes [25 & 26].
Class | Network ID | Network Host ID | Number of Available Networks | Number of Hosts per Network
---|---|---|---|---
A (1-126) | W | x.y.z | 126 | 16,777,214
B (128-191) | w.x | y.z | 16,384 | 65,534
C (192-223) | w.x.y | Z | 2,097,151 | 254

Table 2-2: Category of IPv4 addresses [26]

The Internet Protocol version 4 was designed with limited functionalities like the necessary services/functions to deliver packets from source to a destination over the internet. Based upon the structure of IPv4; it has to process the packets in the same sequence from one host to another. Table 2-3 below illustrates the architecture of IP version 4 packet.

<table>
<thead>
<tr>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>IHL</td>
<td>Type of service</td>
<td>Total length</td>
</tr>
<tr>
<td>Identification</td>
<td>Flags</td>
<td>Fragment offset</td>
<td></td>
</tr>
</tbody>
</table>

Table 2-3: IP version 4 architecture [27]

### 2.3.1. IPv4 limitations

As mentioned in the sections above that IPv4 is capable of supporting approximately 4,294,967,296 (2^32bit) IP addresses. Initially it was expected that IPv4 will provide large number of IP addresses to the network devices and internet users. However, the number of IP addresses available in IPv4 will no longer be able to provide enough space for all the users and devices around the world because over the past decade huge number of internet users in each country has grown. According to researchers IPv4 address space will be exhausted in the next few years as the numbers of users are rapidly rising. In recent years so many mobile devices have been developed which require an IP address. Furthermore, futuristic studies indicate that in the next few years most of the electronic devices will require an IP address to communicate and contact for
example: TV, DVD player, refrigerator, oven, washing-machine and other household items [28]. Figure 2-2 below describes the growth of IP version 4 over the last 20 years.

![Population of IPv4 Hosts](image)

Figure 2-2: The population of IPv4 internet hosts around the world [29]

### 2.4. IPv6 (Internet Protocol version 6)

Internet protocol version 6 is a new version of Internet Protocol that was designed to replace older version (IPv4). IPv4 has been used in the internet world for approximately 2 decades and it was expected that IPv4 had capacity of hosting $2^{32}$ bit addresses and will run out of IP space in next few years. Therefore, IPv6 was introduced before IPv4 is completely exhausted. IPv6 was planned and structured with enhanced features to provide better services than IPv4 over the internet. Most important feature that IPv6 is capable of providing is, its ability to support $2^{128}$ bit addresses. IPv6 eliminates the use of NAT (Network Address Translation) and VLSM (Variable Length Subnet Mask), since it has enough IP addresses for all the users around the globe. Table 2-4 below illustrates the architecture of IPv6 [30].
The development of IPv6 was designed to overcome the IPv4 limitations. However, IPv6 architecture is different from IPv4 architecture as shown in the Table 2-4 above. “IPv6 addresses are expressed in hexadecimal format (base 16) which allows not only numerals (0-9) but a few characters as well (a-f)” [31]. Currently Internet service providers (ISP) have realised that IPv4 may not be available in the future for all the users; so they started to invest in IPv6 networks and some of them already have adopted IPv6 infrastructure. According to Wu & Wang, 2011 [32] USA is currently holding 70% of IPv4 addresses. However, USA still needs more addresses to fulfil their requirements whereas companies in China are already ahead and have established IPv6 networks for their users. Next section compares the features of IPv6 and IPv4 in detail.

### 2.4.1. Comparison between IPv4 and IPv6

IPv6 has many new features that will resolve the issues that came up with IPv4. First of all IPv4 has capacity of $2^{32}$ bit address space while IPv6 provides $2^{128}$ bit. IP configuration in IPv4 was either manually or by DHCP leasing addresses from a given range, whereas IPv6 has auto-configuration and DHCP. Another feature of security is IPSec which was optional in IPv4 while IPv6 has inbuilt IPSec services. There are additional features and modifications incorporated in IPv6 architecture. Table 2-5 below contains features of both IP versions (IPv4 & IPv6) and their comparison [33 & 34].

<table>
<thead>
<tr>
<th>32Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>Version =6</td>
</tr>
<tr>
<td>Payload length</td>
</tr>
<tr>
<td>Source address (128 bits)</td>
</tr>
<tr>
<td>Destination address (128 bits)</td>
</tr>
</tbody>
</table>

Table 2-4: IPv6 architecture [30]
<table>
<thead>
<tr>
<th>Features</th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>32 bits</td>
<td>128 bits</td>
</tr>
<tr>
<td>Checksum in header</td>
<td>Included</td>
<td>No checksum</td>
</tr>
<tr>
<td>Header includes options</td>
<td>Required</td>
<td>Moved to IPv6 extension headers</td>
</tr>
<tr>
<td>Quality of Services (QoS)</td>
<td>Differentiated Services</td>
<td>Use traffic classes &amp; flow labels</td>
</tr>
<tr>
<td>Fragmentation</td>
<td>Done by routers &amp; source node</td>
<td>Only by the source node</td>
</tr>
<tr>
<td>IP configuration</td>
<td>Manually or DHCP</td>
<td>Auto-configuration or DHCP</td>
</tr>
<tr>
<td>IPSec support</td>
<td>Optional</td>
<td>Required</td>
</tr>
<tr>
<td>Unicast, multicast and broadcast</td>
<td>Uses all</td>
<td>Uses unicast, multicast and anycast</td>
</tr>
<tr>
<td>Address Resolution Protocol (ARP)</td>
<td>Used to resolve an IPv4 address</td>
<td>Replaced by Neighbour Discovery</td>
</tr>
<tr>
<td>Internet Group Management Protocol (IGMP)</td>
<td>Used to manage local subnet group</td>
<td>Replaced with Multicast Listener Discovery (MLD)</td>
</tr>
<tr>
<td>Domain Name System (DNS)</td>
<td>Used host address (A) resource records</td>
<td>Uses host address (AAAA) resource records</td>
</tr>
<tr>
<td>Mobility</td>
<td>Uses Mobile IPv4 (MIPv4)</td>
<td>MIPv6 with faster handover, routing and hierarchical mobility</td>
</tr>
</tbody>
</table>

Table 2-5: IPv4 versus IPv6 [34]

2.4.2. IPv6 limitations

Comparison between IPv4 and IPv6 indicates that IPv6 has many new features and also provides solution to the issues which ensue using IPv4. However, even with enhanced features IPv6 still has major issues, as it is not backward compatible. In others words IPv6 cannot communicate with IPv4 hosts directly and neither can IPv4 communicate with IPv6 hosts. According to experts IPv6 cannot be universally implemented and configured within a week globally or even in large corporate organisations because it is both expensive and complex. Next section describes the solution proposed by experts that allows IPv6 hosts to communicate with IPv4 hosts during migration period and would be only provisionally.
2.5. IP Transition Mechanisms

As mentioned in the section above IPv6 was designed with enhanced features to provide better performance than IPv4 (internet protocol) and to replace the existing IPv4 infrastructure. However, IPv6 is not able to communicate IPv4 and neither is IPv4 compatible with IPv6. It would be a long time before IPv6 is universally adopted. According to research it would take several years before IPv6 is fully established and ready to replace IPv4. Therefore, during migration period both IP versions will coexist provisionally. In order to permit both IP versions to coexist, different methods and techniques exist and may be designed for use. Researchers have already designed a few methods such as Tunnelling, Dual-Stack and Translation mechanisms. The following sections will specify the usage of different types of IP transition mechanisms in detail.

2.5.1. IPv6-to-4

IPv6to4 or also known as 6to4 is a tunnelling mechanism, which was mainly designed for IPv6 users. It provides IPv6 based network users to communicate with other IPv6 based networks through IPv4 cloud (Internet). According to Beijnum, 2006 [35] “every system that holds a valid, routable IPv4 address can automatically create a 6to4 prefix for itself by combining its IPv4 address with the 16-bit value 2002 (hexadecimal). The resulting prefix is 48 bits long, leaving enough bits for 65536 64-bit subnets.” Figure 2-3 below illustrates the structure of 6to4 address.

![Figure 2-3: Architecture of IPv6to4 tunnel [35]](image)

When an IPv6 based host wants to send a packet to another IPv6 based host via IPv4 cloud, it initially sends IPv6 packet to 6to4 tunnel. 6to4 tunnel encapsulates the IPv6 packet within IPv4 packet by adding IPv4 header and then sends it through IPv4 cloud.
Once packet reached at the other 6to4 tunnel endpoint, it de-capsulates the packet and removes IPv4 header and carries on the standard procedure of IPv6 to its destination. Figure 2-4 below illustrates, how IPv6 based networks establish connectivity using 6to4 tunnel.

IPv6to4 is considered as automatic tunnel and it is used by IPv6-capable hosts. It establishes communication between 6to4 sites by creating a direct tunnel; however, connectivity among other IPv6 based networks is obtained via 6to4 relay routers. “6to4 routers or hosts often default to using the global anycast address, 192.88.99.1; this way, 6to4 routers or hosts do not need configuration, always using the closest advertising relay to reach the rest of the IPv6 networks” [36]. Next section will discuss IPv6-in-4 tunnelling mechanism in detail.

### 2.5.2. IPv6-in-4

IPv6-in-4 is a tunnelling mechanism which is also known as “Configured Tunnel”. It is configured between hosts manually and it does not require any prefixed IP addresses, unlike 6to4 tunnel. Configured tunnel is generally considered bidirectional and it has the ability to communicate with any IP address range. According to Stockebrand, 2007 [37]
“Configured tunnels are widely used since the bandwidth overhead they require is very small.” Figure 2-5 below illustrates an infrastructure of IPv6-in-4 tunnelling mechanism.

IPv6-in-4 tunnel is based on virtual point-to-point links between sites or hosts. In this case IPv6-capable workstations are connected to a router with its first interface using IPv6 while second interface of a router is configured with IPv4 address which is connected to another router with IPv4 node. Both routers connected with IPv4 based addresses plus they have virtual interface called IPv6-in-4. IPv6-in-4 interface creates a Configured Tunnel between routers and allows IPv6-capable hosts to communicate to each other. The procedure of packet flow is similar as 6to4 tunnel as tunnel endpoints encapsulate the packet with IPv4 header and de-capsulate at the other endpoint and remove the IPv4 header [38 & 39]. The main difference between 6to4 and 6in4 tunnels is that 6to4 required prefixed IP address while 6in4 doesn’t. Next section will specify Dual-Stack Transition Mechanism in detail.

2.5.3. DSTM

Dual-Stack Transition Mechanism also referred to as Dual-Stack is another way to allow IPv6 based hosts to communicate to other IPv6 based networks. However, user’s workstations, routers and operating systems must be compatible with DSTM in order to
establish a Dual-Stack based connection between networks. Old operating systems are not compatible with IPv6. However, software based patches are available which enables them to use DSTM. In this case both IP versions (IPv4 & IPv6) are enabled on a single network interface and have to perform concurrently. According to Xia & Bound [40] in the next few years IPv4 address allocation will be on demand and will be available with a short lifetime. Moreover, when the DSTM based client will send a request to the DSTM server for a dynamic IPv4 address. "This IPv4 address will be reclaimed by the server if its lifetime expires. Thus, the DSTM server can serve the need of IPv4 address for a large number of IPv6 nodes by keeping a small IPv4 address pool" [40]. Figure 2-6 below illustrates the network based on DSTM architecture.

![Figure 2-6: Network based on Dual-Stack transition mechanism](image)

As shown in Figure 2-6 that a connection was established by enabling DSTM. When IPv6 based network sends a packet to DSTM gateway (which has both versions of IP operating concurrently) it uses IPv6 stack and forwards the IPv6 packet to IPv6 network. Moreover, it only allows IPv4 based nodes particularly to communicate with IPv4 based nodes while IPv6 based nodes specifically communicate with IPv6 based nodes. However, IPv6 nodes cannot directly communicate with IPv4 nodes [41 & 42]. Next section will cover Teredo Tunnelling Mechanism in detail.
2.5.4. Teredo

Teredo is another IP transition mechanism which is used to enable IPv6-capable hosts to communicate with IPv4 hosts. It was specially designed for hosts that are located behind the NAT (Network Address Translation). It has different IP address structure than other IP transition mechanisms as it encapsulates IPv6 packet with a UDP port rather than just the IPv4 header. Following is an example of the address procedure which is used by Teredo.

<table>
<thead>
<tr>
<th>32 bits</th>
<th>32 bits</th>
<th>16 bits</th>
<th>16 bits</th>
<th>32 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teredo Prefix</td>
<td>IPv4 Address of Teredo Server</td>
<td>Flags</td>
<td>Mapped Client UDP Port</td>
<td>Mapped Client IPv4 Address</td>
</tr>
</tbody>
</table>

Table 2-6: Teredo tunnel’s address structure [43]

Teredo transition mechanism is considered automatic tunnel as it was designed for IPv6 and NAT users. However, it does not require any permanent IPv4 address. According to Dunmore, 2005 [43] “it is a host-to-host automatic tunnelling mechanism that provides IPv6 connectivity, when Dual-Stack hosts are located behind one or multiple NATs by encapsulating IPv6 packets in IPv4-based User Datagram Protocol (UDP) messages”. Figure 2-7 below illustrates Teredo clients and their connectivity to IPv6 hosts.

![Figure 2-7: An example of a Teredo based network [43]](image-url)
Teredo uses two types of services such as Teredo Server and a Teredo Relay in order to send IPv4 packets to IPv6 network. Initially Teredo tunnel encapsulates IPv6 packet in UDP (User Datagram Protocol) and then UDP port known as 3544 is selected by Teredo Server to obtain a request from Teredo client. Then Teredo Server forwards the packets to “Teredo Relay Router”, which is intended for IPv6 host [44]. Next section will discuss another IP transition mechanism known as NAT-PT.

2.5.5. NAT-PT

NAT-PT (Network Address Translation- Protocol Translation) is an IP translation mechanism, which was especially designed to translate IPv6 address into IPv4 address and IPv4 address into IPv6 address without any encapsulation procedure. NAT-PT is implemented on devices like Server or a router which acts as NAT-PT translator. NAT-PT server is placed between IPv4 network and IPv6 network and when IPv4 network sends a packet to NAT-PT server it translates it into IPv6 and forwards it to IPv6 network. When a packet is sent by IPv6 host it does the opposite process by translating IPv6 into IPv4. NAT-PT does not require any tunnelling system or a Dual-Stack system to be performed in order to get packets forwarded. Figure 2-8 below illustrates a network based on NAT-PT [45 & 46].
NAT-PT architecture is different from tunnelling mechanisms and proposes a simple solution to IPv4 and IPv6 networks. However, NAT-PT translator “eliminates the assumption in SIIT that each IPv6 host has a global IPv4 address. Basically a pool of IPv4 addresses is used for assignment to IPv6 hosts” [47]. If only one IPv4 address is available in the pool, then it is called NAPT-PT (Network Address Port Translation - Protocol Translation). Figure 2-9 below illustrates NAT-PT translation process.

![Figure 2-9: NAT-PT translation processing system [48]](image)

The “Address management” deals with IPv4, IPv6 address and communicates to “Translation Engine”. The Translation engine performs translation process among different protocols (TCP & UDP) and it also informs the “High-reliance base”. The “high-reliance base” ensures all the translation procedure is correctly performed, according to the standard [48].

Selection of IP transition mechanism was determined based on compatibility and availability. Three IP transition mechanisms were selected, known as IPv6-to-4, IPv6-in-4 and Dual-Stack. However, NAT-PT and Teredo were not selected as these two mechanisms were not compatible with the hardware, software and testing tools. Selected mechanisms are available on most of the new operating systems and do not require any additional hardware or software. Next sections will cover different range of voice over IP & video over IP CODECS and protocols in detail.
2.6. Voice over IP Protocols

This section describes the VoIP protocols which are used to carry the voice packets over the internet. These protocols have different features that affect the voice quality during the transmission. Some of the most common and well-known VoIP protocols are covered in the sections below.

2.6.1. SIP

SIP stands for Session Initiation Protocol that was developed by IETF (Internet Engineering Task Force) and it is an application-layer signalling protocol. This is used for different applications such as instant messaging, video conferencing, online gaming and voice communication over the internet. SIP has the ability to redirect calls through UDP and TCP; however, other VoIP protocols are not capable of supporting TCP and only support UDP. It has various features for example it allows media to be added or removed during the sessions and it also permits multicast conferences in the existing sessions [49, 50 & 51]. “SIP transparently supports name mapping and redirection services, which supports personal mobility – users can maintain a single externally visible identifier regardless of their network location”[52].

2.6.2. RTP

RTP stands for Real-time Transport Protocol. This protocol is used to provide end-to-end delivery to various services such as audio, video and multimedia over multicast and unicast networks. RTP Applications characteristically operate on top of UDP (User Datagram Protocol) to provide services like multiplexing and checksum. RTP does not provide resource reservation and it also does not guarantee QoS (Quality of Services) for real time applications. However, it expects lower layer protocols to perform those services. “RTP does not either guarantee delivery through the network or prevent out-of-order delivery, and it does not assume that the underlying network is reliable and delivers datagrams in sequence to the receiving machine” [53].
2.6.3. **H.323**

H.323 is a suite of protocol which was introduced by ITU-T. H.323 is mostly used in VoIP communication over the internet and it is mainly compatible with Megaco (Media Gateway Control Protocol) and SIP. This is also known as an umbrella protocol because it includes a range of other ITU standards. H.323 architecture involves number of other functionalities such as terminal, gatekeeper, gateway and MCUs (Multipoint Control Units).

H.323 protocol was designed for peer-to-peer networks and it had the ability to isolate the devices running H.323 from other devices completely. This function permits each H.323 device to be configured uniquely and to avoid dependence on other devices for standard procedure. This process enables voice devices to avoid losing calls when H.323 gateway gets disconnected from other routers. However, H.323 requires a lot more configuration manually as it was especially designed for peer-to-peer networks [54].

2.7. **Voice over IP CODECS**

There is a collection of voice CODECS developed by ITU (International Telecommunications Union) such as G.711, G.721, G.722, G.723, G.726, G.727, G.728, G.729 for audio compression and de-compression. Each codec has different packet size and performs differently. In this study some of the most commonly used voice CODECS will be covered such as G.711.1, G.711.2, G.723.1, G.729.2 and G.729.3 CODECS. Sections below describe these CODECS in detail.

2.7.1. **G.711**

G.711 codec is a suite of the ITU (International Telecommunications Union) standards and it is used for voice communication and telephony networks. Theoretically G.711 codec provides good quality of voice and requires higher processing, as it has higher bit
rate as compared to other ITU CODECS [55]. Accordingly to Jeong & Kakumanu et al, [56] G.711 voice codec is used for encoding voice and 80 byte frames are sent after 10 ms interval. Thus the rate of a call in one direction is 64 Kbps. Furthermore, enhanced features were added into G.711 such as G.711.1 also known as Wideband Embedded Extension for G.711 pulse code modulation. According to ITU [57] G.711.1 contains an annexure that addresses the usage of G.711.0 with G.711.1. G.711.1 spans bit-rates range of 64, 80 and 96 Kbit/s.

2.7.2. G.723

G.723 codec is another ITU-T standard that was designed for voice & multimedia communication over standard phone system. It is an extension of G.721 codec, which was modified to provide real-time coding and suitable voice quality. Theoretically G.723 codec is not suitable for music and provides lower quality output than other CODECS [58]. According to Menth & Binzenhofer et al, 2009 [59] “it was specially designed for voice encoding at low bandwidth and is mostly used in VoIP applications, e.g., in Netmeeting or Picophone. G.723.1 can operate in two different modes generating 6.4 kbit/s with 24 byte chunks or 5.3 kbit/s with 20 byte chunks every 30 ms.”

2.7.3. G.729

G.729 codec is another ITU-T standard and it has the ability to compress the payload for low bit rate by using an algorithm known as CS-ACELP (Conjugate-structure algebraic-code-excited linear-prediction). Theoretically G.729 codec provides reasonably less delay and high speech quality [60]. According to Varga & Proust et al, 2009 [61] G.729.1 codec is the first speech codec with “an embedded scalable structure built as an extension of an already existing standard. It offers full backward bitstream interoperability at 8 kb/s with the much used G.729 standard in voice over IP (VoIP) infrastructures.”
Table 2-7 below describes the features of three selected ITU-T (International Telecommunications Union) standard voice CODECS such as G.711, G.723 and G.729. The comparison of these CODECS illustrates that each codec has different frame size, payload, speed and processing time, which affect the quality of voice [62].

<table>
<thead>
<tr>
<th>Codec</th>
<th>G.711</th>
<th>G.723.1</th>
<th>G.729</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding speed (Kbps)</td>
<td>64</td>
<td>5.3/6.3</td>
<td>8</td>
</tr>
<tr>
<td>Frame size (ms)</td>
<td>20</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>Processing Delay (ms)</td>
<td>20</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>Look ahead Delay (ms)</td>
<td>0</td>
<td>7.5</td>
<td>5</td>
</tr>
<tr>
<td>DSP MIPS</td>
<td>0.34</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>Payload (bytes)</td>
<td>160</td>
<td>20/24</td>
<td>20</td>
</tr>
<tr>
<td>Number of flows</td>
<td>7</td>
<td>84/71</td>
<td>56</td>
</tr>
<tr>
<td>Subscribed Rate packet time (ms)</td>
<td>20</td>
<td>30.2/30.5</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 2-7: VoIP CODECS specification [62]
2.8. **VVoIP (Video & Voice over Internet Protocol) Protocols**

This section describes five different video protocols namely MPEG-1, MPEG-2, MPEG-4, MKV and FLV.

### 2.8.1. MPEG-1

MPEG-1 (Moving Picture Experts Group 1) is a multimedia standard which is used to broadcast audio and video packets or signals over different networks such as Ethernet, xDSL, ATM and microwave networks. “In MPEG-1, video can be represented as a sequence of digitised images or frames. To decode the frames it is necessary to apply the motion compensation technique which produces a predicted frame using the reference frame(s) and the motion vector obtained during the compression process” [63]. MPEG-1 was the first MM (Multimedia) compression method, which had a speed at approximately 1.5 Mega bits per second (ISO/IEC 11172). Considering the low bit rate of 1.5Mbps for MM services, this standard provides lower sampling rate for the images and uses lower picture rates of 24-30 Hz [64].

### 2.8.2. MPEG-2

MPEG-2 (Moving Picture Experts Group 2) was designed for high quality video, especially for DVD and TV transmission. MPEG-2 standard is capable of supporting SDTV (Standard Definition Television) and HDTV (High Definition Television). MPEG-2 frames are divided into three categories such as I, P & B. “Frames encoded without considering previous or following frames are called I frames. Coded frames with reference to previous I or P frames are called P frames. The B frames are coded with reference to both previous and next frames” [65]. Each I frame is separated by (8×8) blocks. Moreover, MPEG-2 standard uses a fixed frame rate of 29.97 framess/sec (NTSC) and 25 frames/sec (PAL) because this level is suitable for human’s eyes. Improving the quality of video higher than this level would have no affect as human's eyes cannot discern above this level [66].
2.8.3. MPEG-4

MPEG-4 (Moving Picture Experts Group 4) is another standard that was developed by MPEG after MPEG-1, MPEG-2 and MPEG-3 standards. It is different than previous standards and has different compression system than other MPEG standards. “The compression and decompression of information is a small component of the MPEG-4 functionality. During the development of MPEG-4 it became evident that there was a need for the standard to cover the streaming of interactive multimedia content over low bandwidth networks and Internet connections” [67]. MPEG-4 is capable of broadcasting different bit-rates ranging from approximately 10 Kbit/s to 1.5Mbit/s. Moreover, MPEG-4 has the capability to store media elements as objects and media element could be audio or video. It also aids transport of application level MM (multimedia) like computer graphics, animation and regular video files. In some cases MPEG-4 decoder is capable of describing three dimensional pictures and surfaces for files with .MP4 file extension [68].

2.8.4. MKV

MKV stands for Matroska Multimedia Container, which is an open standard multimedia container file format that can hold an unlimited number of video, audio, picture or subtitle tracks inside a single file. Unlike other similar formats, such as MP4, AVI and ASF, MKV has an open specification (open standard) and most of its code is open source. The formats are .MKA for audio only, .MKS for subtitles only, .MKV for audio, video, pictures and subtitles and .MK3D for stereoscopic/3D video. Matroska is also the basis for .webm (WebM) files. Matroska is based on a binary derivative of XML, called the Extensible Binary Meta Language (EBML) which bestows future format extensibility, without breaking file support in old parsers [69, 70, 71 & 72].
2.8.5. FLV

FLV stands for Flash Video which can be watched on most operating systems, using the web browser plug-ins. It is very popular for embedded video on the web and used by YouTube, Google Video, Metacafe, Reuters.com, and many other news providers. FLV use Adobe Flash Player (versions 6 to 10) and “can be delivered in the following ways: (1) using embedded video within SWF files; (2) using progressive download FLV files; and (3) streaming video from own Flash Media Server or from a hosted server using Flash Video Streaming Services” [73].

There are two different video file formats known as Flash Video: FLV and F4V. FLV was originally developed by Macromedia. The audio and video data within FLV files are encoded in the same way as they are within SWF files. However, Flash Video content may also be embedded within SWF files. The F4V file format is based on the ISO based media file format. Flash Video FLV files usually contain material encoded with CODECS following the Sorenson Spark or VP6 video compression formats. The most recent public releases of Flash Player (collaboration between Adobe Systems and MainConcept) also support H.264 video and HE-AAC audio [73 & 74].

2.9. Selection of Voice and Video CODECS & Protocols

This section describes the VoIP CODECS and protocols which were selected for the experiments in this research. It also mentions about video protocols which were chosen for video over IP tests. For VoIP tests different types of voice CODECS were selected such as G.711.1, G.711.2, G.723.1, G.729.2 and G.729.3. For video over IP tests various types of video protocols were selected such as MPEG-1, MPEG-2, MPEG-4, MKV and FLV protocols.
2.9.1. Voice CODECS
Selection of VoIP CODECS was determined based on the aim we had for this research. A goal was to select CODECS, which were completely different from each other and unique. Therefore, different CODECS can clarify the impact of each IP transition mechanisms. Second aim was to select a tool that could support multiple voice CODECS because usage of multiple tools for each of the tested codec would require more time, as a researcher has to learn and deploy the new tools and perform the tests. Thus, the time scope of the research would increase if number of tools were included. Hence, D-ITG tool that had the capability to support five different voice CODECS covering, G.711.1, G.711.2, G.723.1, G.729.2 and G.729.3 CODECS were selected. These CODECS have completely different frame-size, and different range of packets per second as illustrated in the Appendix H.

2.9.2. Video protocols
Selection of video protocols was determined based on the same goal as for VoIP (mentioned in the section above). Selected protocols for video tests were MPEG-1, MPEG-2, MPEG-4, MKV and FLV. These protocols were selected as MPEG group of standards is most commonly used video period standard over the internet plus MPEG standard has made several improvements starting from MPEG-1, MPEG-2, MPEG-3, and MPEG-4 and even more. However, the aim of experiment was to clarify the impact caused by IP transition mechanisms on these protocols which are commonly used over the internet. Moreover, protocols like MKV and FLV were selected because they are being commonly used by the internet users. Next sections will specify the traffic generating and measuring tools in detail.
2.10. The Traffic Generating and Calculating Tools

To measure the network performance, it is essential to select the right tool in order to capture required and accurate results. There are many different sorts of tools that are available to measure voice and video performance over IP networks. However, each tool has its limitation which may not be compatible with the testing environment. The five different kinds of tools that were selected are listed below followed by a brief description of each:

- Zumara
- D-ITG (Distributed Internet Traffic Generator)
- VLC (Video LAN Client)
- SNT (Simple Network Tester)
- Gnome

2.10.1. Zumara

Zumara is a tool that was designed for commercial purpose and it is capable of measuring the performance of IP networks. It has the ability to perform multiple tasks simultaneously and is available in GUI mode. It supports both version of IP (IPv4 & IPv6) and works on Windows operating systems and requires additional plug-ins to operate. It supports multiple traffic types and measures them in many different ways. It is capable of generating many different kinds of CODECS and protocols, such as RTP, SIP, G.711, G.723 and many more. However, it is very expensive as it was especially designed for businesses but it is available as a trial version. Furthermore parameters covered by this tool are delay, jitter, MOS (Mean Opinion Score), throughput and packet loss. However, it has limitations as it does not work across networks, it only measures the performance between two interfaces installed on a single machine [75].
2.10.2. D-ITG

D-ITG (Distributed Internet Traffic Generator) is a network traffic generator and analyser tool, which allows a user to generate different types of traffic such as data, gaming and voice. Multiple metrics are supported by D-ITG such as throughput, jitter, packet loss, standard deviation and delay (one way and round trip time). Moreover, it measures the metrics at multiple levels for example; it calculates three different types of delay results such as “Minimum Delay”, “Average Delay” and “Maximum Delay”. It supports various operating systems such as Windows Server 2008, Windows 7, Linux Ubuntu, Fedora and OpenSuse.

It has the ability to send traffic via UDP and TCP nodes and allows a user to change packet size, and the quantity of the packets. There are two types of D-ITG versions that are available such as GUI and Command mode. However, GUI (Graphical User Interface) mode has some limitation as compared to Command mode.

It has the capacity to generate VoIP traffic and supports different voice CODECS such as G.711.1, G.711.2, G.723.1, G.729.2 and G.729.3 codec. Packet sizes for these CODECS are prefixed and cannot be changed; however number of simultaneous CODECS can be generated. Table 2-8 describes supported voice CODECS and their frame sizes.

<table>
<thead>
<tr>
<th>CODECS</th>
<th>Samples</th>
<th>Frame size</th>
<th>Packets (per sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.711.1</td>
<td>1</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>G.711.2</td>
<td>2</td>
<td>80</td>
<td>50</td>
</tr>
<tr>
<td>G.723.1</td>
<td>1</td>
<td>30</td>
<td>26</td>
</tr>
<tr>
<td>G.729.2</td>
<td>2</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>G.729.3</td>
<td>3</td>
<td>10</td>
<td>33</td>
</tr>
</tbody>
</table>

Table 2-8: VoIP CODECS supported by D-ITG & their specification [76]

It usually needs two types of platforms to measure the performance on the network. D-ITG sender is configured on a client machine and D-ITG Receiver is configured on another client machine. D-ITG sender sends the data to D-ITG receiver and stores the
log files on selected client machine. To measure one way delay requires time synchronization between clients while RTT (round trip time) does not require any additional settings. GUI mode only allows 1 flow at a time while command mode version supports more than 60 concurrent flows. Its features include both versions of IP such as Internet version 4 (IPv4) and internet version 6 (IPv6). Figure 2-10 below illustrates D-ITG suite.

![Figure 2-10: Architecture of D-ITG and its suite](image)

### 2.10.3. VLC

VLC (Video LAN Client) is an open-source media player that allows a user to play audio, video files and it also allows users to stream a live audio and video over network. This tool supports multiple operating systems such as Windows, Linux and Mac and has the ability to support both versions of IP (IPv4 & IPv6). Its latest feature includes streaming over IP version 6, which enables IPv6 based users to stream a video via IPv6 infrastructure.

VLC platform is installed on a client machine and is configured to stream a video file via multiple nodes or ports such as HTTP, UDP and RTP. Once video is up and running, other clients can use the given setting and IP address to watch that video application over the LAN or internet [78].
2.10.4. **SNT**

SNT (Simple Network Tester) is a network performance tester tool, which has many features. SNT is a GUI based commercial tool, which is available for free trial. This tool allows users to test the performance of their entire network. It has the capability to generate and measure different types of traffic such as data, voice and video over the network. It also has the ability to measure bandwidth, latency, jitter, packet loss and MOS. Furthermore it auto synchronizes the time between sender and receiver platforms and is easy to operate. There are many more features which can be used to measure the performance of a network. Moreover it allows changing packet sizes, number of packets per second, as well as number of concurrent flows. However, it has a limitation that it is not compatible with IP version 6 [79].

2.10.5. **Gnome**

Gnome is an application which was designed to monitor the system. This application is mostly available in Linux based operating systems. The output of the captured information is displayed in a graphical format with units. It is capable of auditing various types of services at the same time and presents their results concurrently. The following is a list of the features this application provides.

- Linux Kernel version
- GNOME version
- Hardware
- Installed memory
- Processors and speeds
- System Status
- Currently available disk space
- Processes
- Memory and swap space
- Network usage
- File Systems
Lists all mounted file-systems along with basic information about each.

It observes the behaviour of all the CPU processor separately which are used in the system. The outcome results are presented in a line chart for each processor and percentage of CPU processing power usage. It also monitors memory status which includes memory usage and memory swap as well. Gnome is capable of measuring network history and it calculates the activity of each network card separately. Moreover, it shows received and sent data per second and total amount of data received and sent data. The output of the network results is presented in graphical format and KiB/s (Kilo bytes per second). Furthermore, it has some more features for example file systems monitoring which shows the capacity of hard disk on the systems [80].

2.11. Selection of Traffic Generating and Calculating Tools

For this experimental research, one or more tools were required to test the performance of voice and video over IP. The tools mentioned above were very close to the requirement. SNT was the best tool which could perform all the activities covering both voice and video. However, it had a limitation that it is not compatible with IPv6. Therefore, D-ITG tool was selected to test the performance of voice over IP. It is compatible with IPv4 and IPv6 and allows multiple measurements such as jitter, delay, packet loss and throughput. It also works across multiple operating systems and provides output results in different formats.

The aim for video tests was the same as for VoIP to find a tool that could support multiple protocols. However, finding a single tool for video over IP wasn’t easy, because there wasn’t one single tool that could perform all the tasks required for the test by itself. Therefore, two different tools were selected to fulfil the requirements of video over IP such as VLC and Gnome. VLC tool was selected to generate video traffic over IPv4 and IPv6 using multiple protocols, while Gnome tool was selected to measure the performance of the traffic generated by VLC. Unlike VoIP wherein a single tool was
used, selection of both tools resolved this issue and provided the required testing facilities for this research.

2.12. Related Research Studies

In this section, five closely related studies have been selected and reviewed. There are large number of authors who have carried out numerous studies regarding voice over IP (VoIP) and video over IP using IPv4 and IPv6. However, there are not many studies that identify performance of VoIP & video over IP using IP transition mechanisms in credible resources. Studies in this area that exist at the time of working are listed below:

- Study 1: Evaluating Performance Characteristics of SIP over IPv6
- Study 2: Adaptable Packet Significance Determination Mechanism for H.264 Videos over IP Dual Stack Networks
- Study 3: Impact of IPSec and 6to4 on VoIP Quality over IPv6
- Study 4: Performance Evaluation of IPv6/IPv4 Deployment over Dedicated Data Links
- Study 5: Compressed High Definition Television (HDTV) over IPv6
2.12.1. Study 1

Evaluating Performance Characteristics of SIP over IPv6

In this study Hoeher & Petraschek, et al, 2007 [81] have carried out the research on VoIP using NGN (Next Generation Networking). Main focus of this study was to conduct experiments to evaluate the performance of VoIP on IPv4, IPv6 and IP transition mechanisms. Six different experiments were conducted in order to measure and clarify the impact of each mechanism on SIP based VoIP. The technologies and techniques considered in this test-bed were “Native IPv4, IPv6” and “Tunnelling mechanisms” and “Proxying”.

Design of IP transition scenario:

**Native** - The native scenario needs only the two UAs (User Agents) and their corresponding proxy servers operating uniformly with either IPv4 or IPv6.

**Tunnelling** - For the tunnelling setup the user agent “n7argon” possesses merely an IPv4 address. To interwork with native IPv6 nodes “n7argon” establishes a tunnel (6to4/Teredo) with “n7neon” which acts as terminating tunnel end-point. All other nodes including both proxy servers run IPv6.

**Proxying** - In this scenario the left domain (n7xenon and n7radon) deploys native IPv4 and the right one (n7helium and n7argon) runs only IPv6. The “Proxying” setup introduces an interconnecting proxy gateway that translates signalling information and a media gateway responsible for media translation. Both functionalities are implemented on n7neon. Figure 2-11 below shows the structure of the network test-bed used:
The network setup included had six different types of network test-beds and SIP (Session Initiation Protocol) based VoIP traffic was generated over these networks. They used a tool known as SIPp to generate VoIP calls. This tool allowed transmitting 10 calls per second and they scheduled this tool to perform for 12 seconds in order to generate 100 calls. The metric covered in this study was delay which was measured in a one-way delay scenario. However, two types of “Delay” were analysed such as “Signalling Delay” and “Media Data Delay”. One way delay required time synchronization between hosts; therefore NTP (Network Time Protocol) was used to synchronize the time between sender and receiver platforms, in order to measure accurate one-way delay results.

**Performance results:**

Three different types of results were obtained and compared such as native IPv4/IPv6, Tunnelling and Proxying. These results were analysed in two different categories such as Signalling Delay and Media data delay.
Signalling Delay:

Native IPv4 & IPv6: The outcome shows that Native IPv4 performed marginally better than Native IPv6. IPv4 set up 80% calls within 4 milliseconds whereas IPv6 provided 80% call setup approximately 5 milliseconds. IPv4 has more advantages than IPv6 as most of the applications were designed for IPv4 based network infrastructures. However, IPv6 had 25 percents longer delay than IPv4.

Tunnelling: 6to4 & Teredo: The results analysed for both tunnelling mechanisms indicates that 6to4 performed better than Teredo tunnel. Teredo tunnel provided more delay than 6to4 tunnel at approximately 1.0 millisecond. The authors stated that 6to4 is an automatic tunnel that is easily established between hosts where as Teredo is more complicated as it has to connect hosts located behind NAT.

Proxying: SER & MSP: The SIP Express Router (SER) performed much better than Mini SIP Proxy (MSP). According to authors MSP was designed for low traffic networks. However, in this scenario 100 calls were generated which clearly impacted the MSP.

Media Data Delay:

The results obtained for “media data delay” are mentioned in the Table 2-9 below:

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Mean</th>
<th>Median</th>
<th>Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native IPv4</td>
<td>0.188ms</td>
<td>0.132ms</td>
<td>0.146ms</td>
</tr>
<tr>
<td>Native IPv6</td>
<td>0.190ms</td>
<td>0.139ms</td>
<td>0.230ms</td>
</tr>
<tr>
<td>6to4 Tunnelling</td>
<td>0.200ms</td>
<td>0.180ms</td>
<td>0.292ms</td>
</tr>
<tr>
<td>Teredo</td>
<td>0.964ms</td>
<td>0.742ms</td>
<td>0.818ms</td>
</tr>
<tr>
<td>RTP Proxy</td>
<td>10.017ms</td>
<td>10.660ms</td>
<td>6.130ms</td>
</tr>
<tr>
<td>Ufwdd</td>
<td>0.201ms</td>
<td>0.169ms</td>
<td>0.158ms</td>
</tr>
</tbody>
</table>

Table 2-9: All scenarios tested and their results [81]
Discussion:

The results observed for first scenario showed that Native IPv4 performed better than Native IPv6 and the authors stated that it was due the packet structure of IPv4 \((2^{32})\) and IPv6 \((2^{128})\). Second scenario indicated that 6to4 tunnel performed better than Teredo tunnel. The Teredo tunnel is based on open-source and runs at “user-space” while 6to4 runs in “kernel-space”. Another observation on Teredo was that it is sophisticated and allows NAT users with same IP address to communicate across the internet with IPv6 network whereas 6to4 is straightforward connect two IPv6 hosts via IPv4 network. Third scenario shows that SIP Express Router (SER) performed better than Mini SIP Proxy (MSP) because SER is capable of managing high traffic load while MSP was designed for low traffic load.

2.12.2. Study 2

Adaptable Packet Significance Determination Mechanism for H.264 Videos over IP Dual Stack Networks

Lee and Yu et al, 2009 [82] conducted research on video over IP (Internet Protocol) using DSTM (Dual-Stack Transition Mechanism). The purpose of study was to identify the quality of video using H.264 protocol over Dual-Stack. In this study they developed a new method to resolve the issues and enhance the quality of H.264 video protocol over Dual-Stack Transition Mechanism. The video packet loss issue was undertaken in depth and multiple video protocols were involved as each protocol is based on different characteristics and experiences different errors during transmission. Furthermore, a model which used fixed packets for video traffic and prioritised video packets progression differently is ineffective and reduces the quality of video packets due to significant packet loss in the process of transmission.

To enhance the quality of video protocols a new method called ASDM-TS (Adaptive Significance Determination Mechanism in Temporal and Spatial domains) was designed and tested which purposed a solution to packet loss issue in video transmission. The
method was used for H.264 video protocols using IP Dual-Stack infrastructure with DiffServ model. However, using this new method (ASDM-TS) indicated that the quality of packet loss in video transmission especially when it is broadcast over IP Dual-Stack Transition Mechanism can be improved.

To conduct the experiment a tool known as NS-2 (Network Simulator version 2) was used to simulate and measure the video traffic. The H.264 protocol and JM10.2 codec was used, which compress video at a target rate of 1 Mega bit per second. The video sequence length was set at 300 frames and IPPP video format was selected with size 15 frames. Figure 2-12 below illustrates a flowchart which was designed for ASDM-TS method.
Simulated Results:

Three different network conditions were setup in the following simulation scenarios, which included low, medium and high available bandwidth cases (LAB, MAB and HAB). They also considered three different types of DiffServ levels in these experiments. The results obtained from the experiments were compared with FBC (Frame Based Classification). Three Tables (LAB, MAB & HAB) in Figure 2-13 below present the results for LAB, MAB and HAB.

<table>
<thead>
<tr>
<th>LAB:</th>
<th>FBC</th>
<th>ASC-TS-directly</th>
<th>ASC-TS</th>
<th>ASC-TS-upbound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo</td>
<td>85.68</td>
<td>93.57</td>
<td>93.91</td>
<td>94.89</td>
</tr>
<tr>
<td>Container</td>
<td>85.07</td>
<td>93.78</td>
<td>94.17</td>
<td>95.33</td>
</tr>
<tr>
<td>Foreman</td>
<td>76.61</td>
<td>88.54</td>
<td>89.11</td>
<td>90.58</td>
</tr>
<tr>
<td>News</td>
<td>88.13</td>
<td>93.41</td>
<td>93.7</td>
<td>94.67</td>
</tr>
<tr>
<td>Stefan</td>
<td>80.7</td>
<td>91.81</td>
<td>92.78</td>
<td>94.87</td>
</tr>
<tr>
<td>Bus</td>
<td>78.38</td>
<td>91.67</td>
<td>92.13</td>
<td>93.98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MAB:</th>
<th>FBC</th>
<th>ASC-TS-directly</th>
<th>ASC-TS</th>
<th>ASC-TS-upbound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo</td>
<td>84.39</td>
<td>93.65</td>
<td>94.13</td>
<td>95.56</td>
</tr>
<tr>
<td>Container</td>
<td>84.67</td>
<td>93.33</td>
<td>93.56</td>
<td>94.87</td>
</tr>
<tr>
<td>Foreman</td>
<td>75.44</td>
<td>89.76</td>
<td>90.13</td>
<td>91.54</td>
</tr>
<tr>
<td>News</td>
<td>90.2</td>
<td>94.39</td>
<td>95.17</td>
<td>96.15</td>
</tr>
<tr>
<td>Stefan</td>
<td>80.07</td>
<td>91.31</td>
<td>91.78</td>
<td>94.41</td>
</tr>
<tr>
<td>Bus</td>
<td>75.79</td>
<td>91.85</td>
<td>91.94</td>
<td>93.29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HAB:</th>
<th>FBC</th>
<th>ASC-TS-directly</th>
<th>ASC-TS</th>
<th>ASC-TS-upbound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo</td>
<td>84.57</td>
<td>93.96</td>
<td>94.37</td>
<td>95.41</td>
</tr>
<tr>
<td>Container</td>
<td>85.46</td>
<td>92.33</td>
<td>93.07</td>
<td>94.52</td>
</tr>
<tr>
<td>Foreman</td>
<td>76.94</td>
<td>90.28</td>
<td>90.93</td>
<td>93.02</td>
</tr>
<tr>
<td>News</td>
<td>91.76</td>
<td>93.72</td>
<td>94.65</td>
<td>96.06</td>
</tr>
<tr>
<td>Stefan</td>
<td>82</td>
<td>92.56</td>
<td>93.19</td>
<td>95.22</td>
</tr>
<tr>
<td>Bus</td>
<td>75.37</td>
<td>92.18</td>
<td>92.27</td>
<td>93.7</td>
</tr>
</tbody>
</table>

Figure 2-13: The results of LAB, MAB and HAB [82]

The results analysis indicated that proposed ASDM-TS mechanism successfully resolves the problems “by evaluating the significance of video packets in temporal and spatial domains simultaneously with a self-learning process.” The comparison between traditional FBC scheme and the proposed mechanism (ASDM-TS) showed that ASDM-TS have the ability to enhance the quality of video as it provided the accuracy of significance classification up to 15%. Furthermore, “delivering video data with ASDM-TS
on IP Dual-Stack DiffServ network outperforms FBC priority strategy up to 0.7dB in
PSNR"[71].

Discussion:

According to Lee and Yu et al, 2009 [82] the proposed method (ASDM-TS) can provide
better quality in video communication system up to 15% when it is transmitted over IP
Dual-Stack Transition Mechanism. The results were compared against FBC (Frame
Based Classification) method and stated that ASDM-TS method is better than FBC
method. However the results were performed in simulation environment, whereas
realistic results may have a different outcome comparing to FBC. The authors did not
mention about future work or condusing an experiment to apply these results on realistic
networks to clarify the results.

2.12.3. Study 3
Impact of IPSec and 6to4 on VoIP Quality over IPv6

Yasinovskyy and Wijesinha et al, 2009 [83] conducted research on VoIP (Voice over IP)
to clarify the impact of IPSec and IPv6to4 tunnel on voice quality in LAN environment.
The purpose of study was to measure the impact of NAT (Network Address Transition),
IPSec and 6to4 tunnel on VoIP using IPv6 infrastructure. Hence, five different networks
were setup based on these mechanism mentioned above to perform these tests in a
local area network. The metrics considered in this experiment were delta (packet inter-
arrival time), jitter, packet loss, MOS (Mean Opinion Score) and throughput.

Main focus on IPSec was to measure the additional impact caused by IPSec on VoIP.
Therefore, an IPSec connection was setup on a LAN network using IPv6 and VoIP
communication was established on it. Second test was for IPv6to4 tunnel mechanism. It
was established between IPv6 based networks in order to enable IPv6 based network to
communicate via IPv4 based networks. Additional impact of encapsulation process
(which is essential in 6to4 tunnelling mechanism) on VoIP was measured. For third and
forth experimental test NAT (Network Address Transition) was added into IPSec and 6to4 based networks and the impact of combined mechanisms was tested.

To conduct these tests multiple tools were selected such as Openswan, Linphone (soft-phone) and MGEN. The Openswan was a tool used to setup IPSec connection between networks while Linphone is software which was used as a soft-phone to communicate. Moreover, MGEN is a tool that generates data traffic and it was used to generate background traffic between networks. Figure 2-14 below illustrates a network diagram which was designed for these four scenarios.

![Network Diagram](image)

Figure 2-14: Network diagram based on four scenarios [83]

VoIP calls were generated for 2 minutes using Linphone while only 20 ms of conversation was measured to avoid the start-up noise. The tests were repeated 3 times and background traffic was set at the rate 50, 100, 150 and 200 Mega bits per second using MGEN tool. Furthermore, Linux machines were configured to act as routers and also act as an IPSec/6to4 gateway.
Results:

Overall, it was stated that “effects on VoIP quality are not significant as long as the load does not exceed the capacity of the network; under heavily overloaded conditions (200 Mbps load over 100 Mbps links and passed through several routers), VoIP quality degrades significantly” [83].

The outcome for jitter indicates that 13 ms (milliseconds) jitter was observed on IPv4 while 24ms when traffic load was increased at 150 Mbps. However, IPv6 had 13 ms with no background traffic while 21 ms when traffic load was added at 200 Mbps. IPv6to4 had 13 ms range of jitter without traffic load while 26 ms when traffic load was added at 100Mbps.

The packet loss results showed that no packet loss was noticed when background traffic load was lower than 50 Mbps for all IPSec scenarios with IP versions and 6to4 tunnel. However, when traffic load was increased to 100 Mbps, packet loss was noticed using IPv4, approximately 2% for no-security. The highest amount of packet loss was observed on 6to4 at approximately 55% when traffic load was increased to 200 Mbps. As for IPv6 packet loss ranged from 4-16% while for 6to4 it was from 4-27%.

The result for MOS showed that maximum amount of MOS was measured at approximately 4.41 with 50 Mbps traffic load for all the scenarios tested. However, the outcome of MOS was unacceptable when background traffic was increased to 150 and 200 Mbps. Furthermore, the outcome for throughput stated that all scenarios had approximately the same throughput for a given rate of background traffic using both versions of IP and 6to4 tunnel. “This implies that the additional overhead due to the extra headers and processing due to IPSec and 6to4 does not significantly affect the voice throughput” [83].
Discussion:

Yasinovskyy and Wijesinha et al, 2009 [83] stated that the usage of IPv6, 6to4 and IPSec had no impact on VoIP quality while background traffic was below 50 Mbps. However, significant amount of impact was noticed when traffic load was set over 100 Mbps. 6to4 had impact on voice quality because it encapsulated and de-capsulated the packets between IPv4 and IPv6. Furthermore, IPSec also enabled an encapsulation to secure the packets while being transmitted which caused impact on voice quality. These tests were carried out on Linux based machines; however for more clarification in the results, authors could select different operating systems to compare the tests. Future work was not mentioned regarding these tests to be conducted under different environments.

2.12.4. Study 4

Performance Evaluation of IPv6/IPv4 Deployment over Dedicated Data Links

In this paper, [84] the authors had setup a network based on Dual-Stack transition mechanism (DSTM) to evaluate the performance of four different types of traffic including voice and video over Dual-Stack. The performance parameters considered were packet loss, throughput, bandwidth and delay (End to End). The traffic selected and measured over Dual-Stack was “MPEG-4 IPv6”, “FTP IPv6”, “Internet IPv4” and “VoIP IPv4”. All these tests were simulated using a well-known tool, NS-2 (Network Simulator version 2), which had the ability to conduct all these tests without any additional requirements. Figure 2-15 below illustrates the network setup, which was used for the experiments.
This network included four hosts and three routers. A router on right-side was based on IPv4 while router on left-side was based on IPv6; however, a router between these two routers was setup to act as a Dual-Stack transition mechanism. Each host was tasked to process different type of traffic as illustrated in the diagram above.

**Simulation Cases:**

Four different types of cases were involved using four types of traffic. The time limit set for traffic simulation was 60 seconds. Table 2-10 below describes the four cases used in these experiments.
<table>
<thead>
<tr>
<th>Case</th>
<th>VoIP IPv4 (bytes)</th>
<th>Internet IPv4 (bytes)</th>
<th>FTP IPv6 (bytes)</th>
<th>MPEG-4 (Video)</th>
<th>MPEG-4 (Audio) (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200-500 increment step 100</td>
<td>100</td>
<td>1,000</td>
<td>Rate Factor = 1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>300-600 increment Step 100</td>
<td>1,000</td>
<td>Rate Factor = 1</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>200</td>
<td>1,200-1,500 increment Step 100</td>
<td>Rate Factor = 1</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>200</td>
<td>1,000</td>
<td>Rate Factor = 3-6 with Step 1</td>
<td>300-600 increment Step 100</td>
</tr>
</tbody>
</table>

Table 2-10: Simulation cases [84]

**Results:**

The observation for case 1 showed that when VoIP traffic load was increased up to 400-500 bytes, it impacted the bandwidth and throughput. The traffic load over 400 bytes caused more packet loss. It was also noticed that when traffic load was increased higher than 400 bytes, more packet loss occurred but “Mean End-to-End Delay” was not affected by higher traffic load.

The outcome for case 2 indicated that “Internet IPv4 traffic” affected bandwidth, throughput and packet loss. However, it had no affect on “mean delay”. The “FTP IPv6” had very slight impact when “Internet IPv4 Traffic” packet size was increased.

The results calculated for case 3 showed that various packet size of “FTP IPv6 Traffic” had slight affect on bandwidth, throughput and packet loss. However, “mean delay” was not impacted at all.
The results measured from case 4 indicated that video traffic with “MPEG-4 IPv6” affected the bandwidth and throughput. It also had higher impact on packet loss because video traffic has heavy traffic load than other traffic which caused higher packet drop when congestion occurred. On the other side “Mean End-to-End Delay” was not affected by higher video traffic.

**Discussion:**

In this study authors setup a network based on IPv6-to-IPv4 and IPv4-to-IPv6 transition on Dual-Stack. They also covered four cases on different traffic categories. The results from case 1 & 2 showed that IPv6 performed better than IPv4. It also clarified that IPv6 provided more throughput and bandwidth. According to Sanguankotchakorn & Somrobru, [84] the amount of bandwidth and throughput increased when packet size was increased using IPv6. However, sample method was applied on IPv4 which produced opposite results because IPv4 has limited bandwidth and throughput. These tests were conducted in a simulated environment but authors did not mention about carrying these tests on a real network to clarify the actual impact of IPv6 on different types of traffic.

### 2.12.5. Study 5

**Compressed High Definition Television (HDTV) over IPv6**

In this study, [85] the HDTV (High Definition Television) quality over IP networks was tested and the authors setup a real network across different countries to measure more accurate results. Main focus of this experiment was to implement a two-way network between China and Korea and transmit live video conference using IPv4, IPv6 and Dual-Stack transition mechanism. The parameters considered in this experiment were packet loss, Jitter with buffering and Jitter without buffering. Figure 2-16 below illustrates network design.
As shown in Figure 2-16 above that two High Definition cameras and two High Definition TVs were setup to broadcast and receive video over the IP network. Figure 2-17 below illustrates the locations and capacity of Dual-Stack systems, which was available to be used for this experiment.
This experiment was configured between KAIST and Tsinghua universities as shown in Figure 2-17 above. The video type broadcast over IPv6 was compressed HDTV and both Server and clients were capable managing HD video. Table 2-11 below describes the format, pixels and frame rate of the SDTV (Standard Definition Television) and HDTV (High Definition Television).

<table>
<thead>
<tr>
<th>Format</th>
<th>Vertical Lines</th>
<th>Horizontal Pixels</th>
<th>Aspect Ratio</th>
<th>Scan Mode</th>
<th>Frame Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDTV</td>
<td>1080i</td>
<td>1080</td>
<td>1920</td>
<td>Interlaced</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>720p</td>
<td>720</td>
<td>1280</td>
<td>Progressive</td>
<td>60</td>
</tr>
<tr>
<td>EDTV(DVD)</td>
<td>480p</td>
<td>480</td>
<td>704</td>
<td>Progressive</td>
<td>60</td>
</tr>
<tr>
<td>SDTV</td>
<td>480i</td>
<td>480</td>
<td>640</td>
<td>Interlaced</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 2-11: The detail of standard definition TV and high definition TV [85]

**Configuration:**

To setup a two-way communication system between two countries, following router and devices were used. A lab located in Korea used Gigabit switch and another Gigabit switch was located in KAIST, which was connected to GSR12008 router. GSR 12008 router was connected to another router, which was connected to Cisco 7609 router. Cisco 7609 router was then connected to GSR12410 router using 155Mbps Dual-Stack link. GSR12410 router was connected to 7507 Cisco router, which was then connected to GSR12410 router. It was connected to three more GSR 12016 routers and then it was connected to a switch in laboratory which was located in China. Overall 9 routers and 3 switches were connected to establish this connection between two countries. Figure 2-18 below illustrates the infrastructure of these devices.
Results:

Two different types of metrics were considered in this experiment such as jitter and packet loss and their average results were obtained. The analysis of results indicated that HDTV over IPv4 from China to Korea performed reasonably. However, Dual-Stack mechanism did not perform well, when both versions of IP (IPv4 & IPv6) were used concurrently. The outcome of HDTV over IPv6 was analysed and it showed that IPv6 created high amount of packet loss.

Packet loss:

The authors did not use any FEC (Forward Error Correction) process during the experiments. The packet loss for IPv6 using one-way traffic indicated that it produced very minimal packet loss at approximately 0.1%. However, IPv6 using a two-way communication between sites caused large amount of packet loss at approximately 44%.
Jitter:

Two different types of jitter results were measured such as jitter with buffering and jitter without buffering. The outcome for jitter without buffering showed that it has higher jitter than jitter with buffering. The jitter without buffering had a range between 7 to 15 milliseconds while jitter with buffering had a range between 1 to 5 milliseconds.

Discussion:

The video transmission over IPv4 stated that there is no major concern while using one-way and two-way video communication and outcome is stable for both. However, results for IPv6 indicated that using two-way transmission has caused significant impact on packet loss (44%) due to the network devices. Overall it was concluded that devices used in the infrastructure of IPv6 have caused this major packet loss as these device were not compatible with each other in regards to IPv6 traffic forwarding. The authors discussed this issue and concluded that performance of IPv6 using two-way transmission can be improved if all the routers between sites are fully capable of managing IPv6 packet forwarding. They also stated that to identify the problem within routers required measuring the performance of each router using IPv6 to clarify the results and pinpoint the faulty router, which caused large amount of packet loss. The jitter results were calculated with and without the buffering system. However, no quality difference was noticed for jitter with and without buffering. The buffering system had less impact on HDTV but it had higher impact on MPEG-2 protocol.

2.13. Identified Gaps

This section discusses research gaps which were identified in previous studies. Table 2-12 below describes the summary of related studies which were reviewed in the above sections. These studies have covered multiple tools to generate and measure different types of traffic using various IP transition mechanisms.
### Table 2-12: Summary of related studies

<table>
<thead>
<tr>
<th>Authors</th>
<th>Traffic &amp; Tools</th>
<th>IP Transition Mechanisms</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>T. Hoeher, M. Petraschek, S. Tomic and M. Hirschbichler</td>
<td>VoIP traffic with SIPp tool</td>
<td>IPv4, IPv6to4 tunnel &amp; Teredo tunnel</td>
<td>2007</td>
</tr>
<tr>
<td>C. Lee, Y. Yu, and P. Chang</td>
<td>Video traffic with NS-2 tool</td>
<td>Dual-Stack Transition Mechanism</td>
<td>2009</td>
</tr>
<tr>
<td>R. Yasinovskyy, A.L. Wijesinha and R. Karne</td>
<td>VoIP traffic with Openswan &amp; MGEN tools</td>
<td>IPv6-to-4 tunnel with IPSec &amp; NAT</td>
<td>2009</td>
</tr>
<tr>
<td>T. Sanguankotchakorn &amp; M. Somrobru</td>
<td>VoIP, Video, FTP traffic with NS-2 tool</td>
<td>Dual-Stack Transition Mechanism</td>
<td>2005</td>
</tr>
<tr>
<td>J. Lee &amp; K. Chon</td>
<td>Video traffic with video camera</td>
<td>Pure IPv4 &amp; IPv6 across the globe</td>
<td>2006</td>
</tr>
</tbody>
</table>

Firstly, Table 2-12 above stated the IP transition mechanisms which were covered in previous studies. As mentioned above that Dual-Stack, IPv6-to-4 and Teredo Transition mechanisms were tested under different environments. The first gap which was identified is a different IP transition mechanism. Current study will cover IPv6-in-4 tunnelling mechanism which had been tested by previous studies. Another gap is the test environment as noticed that Dual-Stack Transition Mechanism was tested under simulation environment whereas current study will have implementation of Dual-Stack on real networks.

Secondly, difference in performance of tools used in previous studies was identified. The tools covered in previous studies were SIPp, NS-2, Openswan and MGEN. However, the current study being undertaken will use a well-known tool D-ITG to perform all the tasks for VoIP tests. To generate and measure video traffic on IP transition mechanisms multiple tools will be covered which are called Gnome and VLC client as against the usage of NS-2 tool used in previous studies which only simulated.
the traffic under virtual environment. Current study is aimed to use the tools which will generate actual traffic for obtaining results.

Thirdly, current study will cover different platforms as compared to previous studies. The platforms and operating systems covered in previous studies were Microsoft Server 2003 and Linux operating systems and NS-2 platform. However, current study will include five different and new released operating systems such as Microsoft Windows Server 2008, Microsoft Windows 7, Linux (Ubuntu 10.10), Fedora 14 and OpenSUSE 11.3.

Fourthly, another gap was identified which is usage of different protocols. The previous studies covered SIP protocol for VoIP where as current study will consider five different voice CODECS namely G.711.1, G.711.2, G.723.1, G.729.2 and G.729.3. Another gap between video tests noticed was, while previous studies used MPEG-4 protocol to test video traffic, the current study will cover five different video protocols such as MPEG-1, MPEG-2, MPEG-4, MKV and FLV.

Fifthly, another research gap was noticed with various measurement metrics. The previous research studies had covered multiple metrics such as delay (one-way), jitter, packet loss, throughput, bandwidth and MOS. Current study will include delay (Round Trip Time), jitter, throughput and packet loss for VoIP but actual gap was identified between video tests. Current study will consider CPU utilisation and “Impacted-Throughput” in other words additional bandwidth wastage; this was caused by IP transition mechanisms during video transmission. This metric was not covered by any previous studies, which clearly identify the impact caused by IP transition mechanisms on five different video protocols. Next sections will cover metrics in detail which were selected for current study.
2.14. Metrics Tested

This section describes the performance metrics which will be covered for the experiments. Two types of tests will be conducted such as voice over IP and video over IP. Each part of the experiment will focus on different metrics. Voice over IP part will consider delay (Round Trip Time), jitter, throughput and packet loss. Video over IP part will consider actual-throughput, impacted-throughput and CPU utilisation.

2.14.1. Delay

Delay is the time taken for end-to-end transit between sender and receiver which is “elapsed time for a packet to pass from the sender through the network to the receiver.” In this case if voice/video packets take longer time from sender to receiver hosts then the delay would be higher, in other words, quality of voice and video will be degraded. Performance of network will be better if the time taken between sender and receiver hosts is lesser [86].

2.14.2. Jitter

Jitter is the variation in delay. Jitter is vital in voice and video communication because if transmission delay varies too widely during a VoIP conversation or video conference, the quality of voice and/or video will be degraded [87].

2.14.3. Packet loss

Packet loss occurs during the communication between two or more hosts across the network. When two hosts exchange packets between their operating systems some of the packets get dropped during the transmission due to overload which is called packet loss. Losing packets during voice communication degrades the quality of voice [88].
2.14.4. Throughput

"The throughput of a network represents the amount of network bandwidth available for a network application at any given moment, across the network links." Performance of the throughput between networks can be impacted/affected by some activities such as network LAN cards, switches, routers and the network design etc. [89].

2.14.5. Actual-throughput

This is an original throughput, which is required in order to establish a video conference between two hosts. This is calculated when a user sends video packets from one platform to another. Actual-throughput is an essential numbers of packets per second to establish a video conference without having any bad quality or packet loss in transmission. In this case it will be measured over pure IPv4 and pure IPv6 networks using multiple video protocols to identify the Actual-throughput of each video protocol.

2.14.6. Impacted-throughput

This is an additional usage of bandwidth while actual-throughput is transmitted over the IP transition mechanisms. This mainly occurs due to encapsulation and de-capsulation procedure required by IPv6to4 and IPv6in4 tunnelling mechanisms. However, encapsulation and de-capsulation system is essential for tunnelling mechanisms as they are based on these functions. This additional usage of bandwidth is a waste of bandwidth because it establishes a video conference between two hosts using more bandwidth due to encapsulation and de-capsulation procedure during the transmission. In this case video packets with small packet size are sent from one platform to another and more bandwidth is used, which is called impacted-throughput.
2.14.7. CPU utilisation

CPU utilisation performance test is conducted when an application runs on a workstation and CPU (Central Processing Unit) power is allocated for that particular application. It is normally calculated as a percentage, when an application is executed for the amount of CPU resource used. There are two possibilities; firstly, if large quantity of packets is transmitted over the network, more CPU resources will be used. Secondly, if lesser quantity of packets is transmitted over the network and more CPU power is used for a specific transition mechanism. This performance test establishes the overhead as a result of the transition mechanism. In this case different types of IP networks will be setup using IPv4, IPv6, IPv6to4, IPv6in4 and Dual-Stack mechanisms. CPU utilisation tests will be conducted during video transmission over these mechanisms.

2.15. Chapter Summary

This chapter covered introduction to VoIP and video over IP and explained how these technologies are growing and their limitations. It also covered multiple VoIP CODECS and protocols. Video protocols selected in this study were also described in detail such as MPEG-1, MPEG-2, MPEG-4, MKV and FLV. Moreover, multiple metrics considered namely jitter, packet-loss, delay, throughput, actual-throughput, impacted-throughput and CPU utilisation. Finally, it mentioned the related studies to current study, which explains the issues with these technologies and the level of research recently being carried out by other authors. Next chapter will cover the methods and techniques used for this research study.
Chapter 3: Methodology

This chapter covers the methods and techniques involved to drive this research. Initially it describes hypotheses used in this research and then it specifies methodologies, which were adopted for the research. Finally, it illustrates the steps involved in data collection process.

3.1. Research Hypotheses

The following hypotheses will be used to manage the research activities and tasks.

**Hypothesis 1:** It is expected that quality of the voice will reduce as large amount of simultaneous VoIP calls are transmitted on pure IPv6 network as compared to pure IPv4.

**Hypothesis 2:** Voice CODECS with small packet size are expected to support more calls than the CODECS with large packet size over IP tunnelling mechanisms.

**Hypothesis 3:** The performance of VoIP over IP transition mechanisms are expected to be even lower than pure IPv6 network.

**Hypothesis 4:** Video quality is expected to be impacted by the IP tunnelling mechanisms.

**Hypothesis 5:** Video protocols with smaller packet size are expected to perform better on IP transition mechanisms from quality of service point of view.
3.2. Method of Study

There are different types of methods, which are used to manage a research such as quantitative and qualitative. In this study, a quantitative method was adopted to manage the research. According to Daniel, 2004 [90] quantitative research includes “explaining phenomena by collecting numerical data that are analysed using mathematically based methods (in particular statistics).” Main focus of this study was to collect data by carrying out different experiments. Therefore, in this study, quantitative research method was used to manage the research.

According to Gray and Williamson et al, 2007 [91] there are two main types of quantitative research designs, which are used to manage the research such as experimental designs and non-experimental designs. Experimental based research involves tests and numerical statistic whereas Non-experimental based research includes survey based information. In this study data was gathered based on experimental research in computer laboratory environment. Three different experimental designs were planned to measure data. Aim of the first design was to identify the performance of VoIP on IP transition mechanisms using five different computing platforms. Second design aimed to identify the quality of VoIP on IP versions and IP transition mechanisms. Final design was aimed to clarify the impact caused by IP transition mechanisms on video applications. In these experiments appropriate traffic generating and measuring tools were used to collect and measure the results, the details of these are covered later.
3.3. Methodology for Experiments

To manage the research in an efficient way, a water fall methodology was used. The water fall methodology is based on steps, which helps researchers to carry out the research steps accordingly. The water fall, allows a researcher to define the tasks and organise them in a step by step flow. Figure 3-1 below illustrates the water fall methodology:

The water fall methodology is a step by step methodology with each step to be completed in order to start the next step. As shown in Figure 3-1 above, first of all, information was gathered during literature review which also covered the network traffic generating and measuring tools research. Once tools were selected the next step was to implement the networks which were designed for this research. Once networks were fully implemented with the help of tools, data was generated and various outputs captured. Data analysis was initiated, once the data was collected and the results were documented in the final report. The final report also includes all the steps taken to
implement the networks and the scripts used to configure them, including the configuration for the tunnelling mechanisms used.

3.4. Data Collection Process

This section describes the data collection process, which was used to obtain the data for this research. A quantitative research method was selected as this research generally focuses on experiments. Mainly there are two types of data collection methods that were involved in this research such as literature review and experimental observation. Literature review based data was gathered from credible conference papers, books and Journal papers while experimental observation based data was gathered by carrying out multiple tests.

3.5. Literature Review Process

Initial step to gather information about the research was through study of relevant literature. It was expected to enhance the researcher intellectually and empower him/her with the knowledge related to his/her research while reading, analysing and evaluating other's research works. It also provided the background and innovative ideas regarding research dilemma which was being investigated. In this research different credible resources were used in order to get essential information to drive this research and appropriately affirm or negate the research hypotheses.

Literature was gathered from journal papers, conference papers, books and appropriate credible association websites. Following is a list of the credible resources, where the information for this research was retrieved.
Online digital resources: IEEE (The Institute of Electrical and Electronics Engineers) [93] and ACM (The Association for Computing Machinery) [94]

Books: Google search engine for books [95] and Unitec library [96]

Search: Google scholar [97] and I seek education [98]

The resources mentioned above provided lead to research papers relevant to the study, as they are well-known and credible for information technology based researches. Once the literature is reviewed and critically analysed and synthesized, it leads to the next step which is data gathering from experiments. Further sections cover observation segment, which involves experimental tests to help this research to achieve its goals.

3.6. Experimental Observations

Main resource of data gathering for this research was collecting data from the tests. Multiple network setups were established and numerous of tests were run in order to achieve the results. Generally two different types of data collection processes were used as this research design includes two different types of tests which were voice over IP and video over IP. Overall, three kinds of research designs were planned as discussed in the sections below. However, part 1 and part 2 were on the same structure while part 3 was based on different structure. Therefore two different data collection processes were required.

The first two parts (1 & 2) of the experiment involved voice over IP tests. To gather information for voice over IP (VoIP) experiments, a well-known tool, D-ITG (Distributed Internet Traffic Generator) was selected. This tool has the ability to generate, measure and evaluate the voice performance over the IP networks and store the information collected into binary log files. To retrieve data from those binary log files “ITGDec” is used, which is part of D-ITG tool and it allows a user to decode binary files and display them. Figure 3-2 below illustrates an example of a decoded file and shows the results retrieved from binary log file.
Figure 3-2: An example of D-ITG output

decode the receiver log file on the destination:  
host:  [donato@catarella tmp]$ ./ITGDec  
recv_log_file

Flow number: 1
From 10.0.0.4:33029
To 10.0.0.4:10002

Total time = 9.998991 s  
Total packets = 1000
Minimum delay = 0.000150 s  
Maximum delay = 0.000200 s
Average delay = 0.000175 s
Average jitter = 0.000110 s
Delay standard deviation = 0.000136 s
Bytes received = 2482
Average bitrate = 1.985800 Kbit/s
Average packet rate = 101.911 pkt/s
Packets dropped = 0 ( 0 %)

Figure 3-2 above shows detail of binary log file after ITGDec decoded it. ITGDec is part of D-ITG tool which output the results tested. As visible at the top a number indicates the “Flow” which is 1 in this case and the address of host and destination workstations using IP version 4. Second sector presents total time which is 9.998991 seconds followed by total packets sent “1000”. A row below “Total packets” shows “Delay”, which is in three type’s minimum, maximum and average followed by “average jitter” which is 0.00110 seconds. The row below “Delay standard deviation” shows “Average bitrate” which is a throughput and the last row “Packets dropped” show the numbers of lost packets in percentage format.

Next step is to transfer data from binary log files to MS Excel sheets and to present these as graphs. However, DIT-G cannot directly convert binary log files into MS Excel sheet, which requires additional processes. Firstly binary log files are converted into notepad files and then, manually the data is selected and is added into MS Excel sheets. The results compiled for RTT, jitter and packet loss were presented as bar
graphs while throughput was presented in tables. Figure 3-3 below illustrates the example of a bar graph, which was used to present the data for ease of comparison.

![Bar chart example](image)

Figure 3-3: An example of bar chart:

Top heading of the Bar graph lists the name of the voice codec used and at the top right corner the name of the IP transition mechanisms which were tested. On the left side along the Y-Axis of the graph the metric under test is visible, which in this example is “jitter” along with units in milliseconds. Along the X-Axis of the graph names of the platforms tested are mentioned.

Second procedure of data collection includes different steps to gather data and it was for video over IP tests. For this part of the experiment multiple tools were selected to generate and measure the data features. Firstly, a tool known as VLC player [78] was used to broadcast the video transmission over the IP networks then another tool known as Gnome was selected to capture the data. A Gnome is a tool that has the capability to measure multiple traffic types such as voice data and video traffic on IP transition mechanisms. This tool also has the ability to monitor the performance of resources used such as CPU, Memory and Network History. Below is a dumped screenshot, which was captured during the test and it illustrates how data is captured. Initially data is processed.
as a line graph and shows values at bottom along the X-Axis of the diagram below. Overall, Figure 3-4 below shows the format of Gnome tool during the data capturing mode.

A snapshot of Gnome tool, which was captured during tests is visible in Figure 3-4 above. The top of the figure shows heading of the metric while the Y-Axis shows units as kilobytes per seconds (KiB/s). At the bottom along the X-Axis it shows sent traffic in KiB/s which is 255.8KiB/s and in the mid-bottom it shows received data in KiB/s (242.8KiB/s). The line graph depicts two flows of traffic in two different colours. Blue line represents received traffic and purple line represents sent traffic. The screen displays sixty (60) seconds detail of network traffic, which is divided by 10 seconds per column.

Next step after data collecting is to manage and store the data into MS Excel sheets [99]. Following is an example of data stored and measuring after collecting and Figure 3-5 below depicts this:
A diagram shown above presents an example of how data was measured and stored. MS Excel sheet was used to store data and to compute the average results captured as shown under each column. The first column shows number of runs which were repeated for each test and next five columns define the data for IPv4, IPv6, 6to4, 6in4 and Dual-Stack headings. The bottom readings highlighted in green depict the average results measured, which were used to create the charts. In Figure 3-6 below is an example of area chart, which was used to present video over IP data with throughput on the Y-Axis and repressed in kilo bytes per second.
3.7. Chapter Summary

In this chapter hypotheses and the research methods were described. A quantitative research method was used to manage the research and to collect data. Mainly two types of data collection procedures were described. Firstly, literature review based data was collected via credible resources while primary data was gathered by conducting experiments. Two different types of experiments were conducted and each network step had different procedure for gathering data and information from the experiments. Mainly traffic generating and measuring tools were used to capture the data from the experiments and multiple tools were used to obtain different types of data. Next chapter is Experimental Design, which explains clearly how the setup for network design was implemented and configured using different mechanisms and techniques.
Chapter 4: Experimental Network Design

This chapter describes a structure of the experimental network design used in this research to obtain the results. It also covers software and hardware detail and a network test-bed diagram. In this design 5 different operating systems were involved and three IP transition mechanisms were configured on these operating systems. Main focus of the research was to clarify the impact caused by each IP transition mechanism for VoIP and video traffic. Thus, VoIP and video traffic was generated from one network to another via IP transition mechanisms and average results were compiled.

4.1. Network Test-Bed Design

In this research a structure of networks was designed that contains 4 workstations. The network infrastructure was based on 3 different networks, where two client machines and two routers machines were deployed. First network was based on IPv6 configurations and was connected to 2nd network which was based on IPv4 configurations. This IPv4 cloud (2nd network) was then connected to 3rd network, which was based on IPv6 configurations. Figure 4-1 illustrates the network infrastructure:
In this network design Cat5 cable was used and 1 Gigabit NIC card was installed on both client machines and dual Gigabit NIC cards were installed on both routers' machines. Both client machines had Microsoft Windows 7 operating system installed during all the tests conducted while router machines had to switch to different operating systems each time. Therefore five different operating systems used such as Fedora-14, OpenSuse-11.3, Ubuntu 10.10, Windows Server 2008, and Windows 7.

To generate and measure VoIP traffic over IP transition mechanisms, a tool known as D-ITG was used. D-ITG-Sender which is part of D-ITG tool was installed on a client machine and D-ITG-Receiver which is also part of D-ITG tool was installed on another client machine. The aim for this experiment was to measure end-to-end performance of VoIP, in other words VoIP traffic was generated from client one and was received at 2nd client machine via both routers. All tests in this experiment for VoIP were based on round-trip-time (RTT). Furthermore, parameters considered were jitter, delay, throughput & packet loss).

To generate and measure video traffic over IP transition mechanisms, two different tools were selected such as VLC and Gnome. A tool called VLC (Video LAN Client) was used to broadcast VVoIP traffic over the networks. However, “Gnome” tool was used to measure the performance of VVoIP which was installed on “Router 1” machine. The aim of this experiment was to measure the impact caused by IP transition mechanisms on video traffic. Therefore, all tests were measured on “Router 1” as encapsulation and de-capsulation of packets is processed on routers where IP transition mechanisms were deployed. The video protocols selected for this experiment were (MPEG-1, MPEG-2, MPEG-4, MKV & FLV) and parameters considered were CPU utilisation, actual-throughput and impacted-throughput.
4.2. Hardware Specification

To measure the quality of VoIP and video on various IP transition mechanisms, identical hardware components were used in order to measure accurate performance of IP transition mechanisms. Table 4-1 below describes the hardware details:

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>an Intel® Core™ 2 Duo 6300 1.87 GHz processor</td>
</tr>
<tr>
<td>Memory</td>
<td>4.00 GB DDR2 RAM</td>
</tr>
<tr>
<td>Hard-disk</td>
<td>Western Digital Caviar SE 160 GB hard-drive</td>
</tr>
<tr>
<td>Cable</td>
<td>Fast Ethernet cables</td>
</tr>
<tr>
<td>LAN Card</td>
<td>Broadcom NetXtreme Gigabit Ethernet</td>
</tr>
<tr>
<td>Switch</td>
<td>Layer 3 Cisco switch</td>
</tr>
</tbody>
</table>

Table 4-1: Hardware details

In this experiment four different workstations were involved. Two workstations acted as routers, with two additional NIC cards and other two workstations were used as clients at both ends of the networks.

4.3. Software Specification

In this experiment various operating systems were involved in order to measure the performance of different IP transition mechanisms on each operating system. Two types of operating systems were included such as Microsoft based operating systems and Linux based operating systems. Table 4-2 below describes operating systems detail:
4.4. **Configuration of IP Transition Mechanisms**

In this research three different types of experimental test-beds were setup. Purpose of first test-bed contained three IP transition mechanisms (IPv6-to-4, IPv6-in-4 and Dual-Stack) on five different platforms known as Windows Server 2008, Windows 7, Linux Ubuntu 10.10, Fedora 14 and OpenSuse 11.3. These five operating systems were installed on routers machines and IP transition mechanisms were configured on them. Furthermore, client machines were identical and had Windows 7 OS installed on them. Second test-bed aimed to measure the performance of VoIP on pure IPv4, pure IPv6 and compare the results against first test-bed results which were tested over IPv6-to-4, IPv6-in-4 and Dual-Stack configurations. Only two operating systems were selected for second test-bed such as Windows and Linux. The operating system selected for Windows environment was Windows Server 2008 and for Linux it was Ubuntu 10.10 operating system. These two operating systems were installed on both routers machines and Windows 7 OS was installed on both clients' machines.

Final test-bed involved video over IP transition mechanisms, which includes only Linux platform knows as Ubuntu 10.10. The metrics measured includes CPU utilisation, actual-throughput and impacted-throughput. These metrics were measured on “pure IPv4”, “pure IPv6”, IPv6-to-4, IPv6-in-4 and Dual-Stack configurations.

<table>
<thead>
<tr>
<th>Software</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows 7</td>
<td>Service pack 1</td>
</tr>
<tr>
<td>Microsoft Windows Server 2008</td>
<td>Service pack 1</td>
</tr>
<tr>
<td>Linux Ubuntu</td>
<td>Version 10.10</td>
</tr>
<tr>
<td>Linux Fedora</td>
<td>Version 14</td>
</tr>
<tr>
<td>Linux OpenSUSE</td>
<td>Version 11.3</td>
</tr>
</tbody>
</table>

Table 4-2: Software details
Note: Each router contains two network interfaces and in this experiment they were named “Private” and “Public”. Private interface is connected to a client whereas Public is connected to IPv4 cloud.


Note: To run the commands described below, a user must be logged-in as administrator or select command prompt to run as administrator:

Note: In Windows 7 OS to enable routing, a change was made in Windows registry (see Appendix G).

Configuration for Router 1:

- To create and enable 6-in-4 tunnel:
  
  ```
  # netsh int ipv6 add v6v4tunnel "IPv6-in-4-Tunnel" 2.16.1.1 2.16.2.2
  ```

- To configure IPv6 address in a tunnel interface:
  
  ```
  # netsh int ipv6 add address "IPv6-in-4-Tunnel" 2001:210:110:11::1
  ```

- To configure static routing for a tunnel interface:
  
  ```
  # netsh int ipv6 add route 2001::/0 "IPv6-in-4-Tunnel" 2001:210:110:11::2
  ```

- To setup packet forwarding for interfaces:
  
  ```
  # netsh int ipv6 add route 2001:210:110:11::/64 "Private" Set packet forwarding
  # netsh int ipv6 set int "IPv6-in-4-Tunnel" forwarding=enable
  # netsh int ipv6 set int "Private" forwarding=enable advertise=enable
  ```

Configuration for Router 2:

- To create and enable 6in4 tunnel:
  
  ```
  # netsh int ipv6 add v6v4tunnel "IPv6-in-4-Tunnel" 2.16.2.2 2.16.1.1
  ```
To configure IPv6 address in a tunnel interface:
# netsh int ipv6 add address "IPv6-in-4-Tunnel" 2001:210:110:11::2

To configure static routing for a tunnel interface:
# netsh int ipv6 add route 2001::/0 "IPv6-in-4-Tunnel" 2001:210:110:11::1

To setup packet forwarding for interfaces:
# netsh int ipv6 add route 2001:210:110:11::/64 "Private" Set packet forwarding
# netsh int ipv6 set int "IPv6-in-4-Tunnel" forwarding=enable
# netsh int ipv6 set int "Private" forwarding=enable advertise=enable

---

4.4.2. Configuration of IPv6-to-4 on Win 7 and Win Server 2008

Note: To select IP address for 6to4 mechanism a special calculator is used to calculate IP addresses. Selected IP addresses for IPv6 and IPv4 must be compatible; otherwise they will not communicate with each other. IPv4 address for "Public" interfaces in a router #1 is 2.16.0.16 and in a router #2 is 2.16.1.1.

Note: In Windows 7 OS to enable routing protocol, a change was made in Windows registry (see Appendix G).

Configuration for Router 1:

- To create and enable 6to4 tunnel:
  # netsh int ipv6 6to4 set state enabled

- To configure IPv6 address in a "Private" interface:
  # netsh int ipv6 add address "Private" 2001:210:10:1::1

- To configure static routing for 6to4 tunnel interface:
  # netsh int ipv6 add route 2001:210:10:1::/64 "Private"
  # netsh int ipv6 add route 2001:210:101:1::/64 "Private"

- To set packet forwarding:
  # netsh int ipv6 set int "Private" forwarding=enable advertise=enable
  # netsh int ipv6 set int "6to4" forwarding=enabled
Configuration for Router 2:

- To create and enable 6to4 tunnel:
  
  # netsh int ipv6 6to4 set state enabled

- To configure IPv6 address in a Private interface:
  
  # netsh int ipv6 add address "Private" 2001:210:101:1::1

- To configure static routing for 6to4 tunnel interface:
  
  # netsh int ipv6 add route 2001:210:101:1::/64 "Private"
  
  # netsh int ipv6 add route 2001:210:10:1::/64 "Private"

- To set packet forwarding:
  
  # netsh int ipv6 set int "Private" forwarding=enable advertise=enable
  
  # netsh int ipv6 set int "6to4" forwarding=enabled

4.4.3. Configuration of Dual-Stack on Win 7 and Win Server 2008

On both operating systems, IPv4 and IPv6 were enabled on each interface and were configured with normal settings. Routing protocol for IPv4 was RIPv2 and for IPv6 had static configuration enabled.

Note: In Windows 7 OS to enable routing, a change was made in Windows registry (see Appendix G).

Configuration for Router 1:

- IPv4 Interface settings:
  
  # IP address 2.16.2.2 sub-net mask 255.0.0.0 default gateway 2.16.1.1

- Enabled RIPv2 protocol:
  
  # for IPv4 it was enabled from GUI mode and for IPv6 it was configured statically

- IPv6 Interface settings:
  
Configuration for Router 2:

- IPv4 Interface settings:
  # IP address 2.16.1.1 sub-net mask 255.0.0.0 default gateway 2.16.2.2
- Enabled RIPv2 protocol:
  # for IPv4 it was enabled from GUI mode and for IPv6 it was configured statically
- IPv6 Interface settings:
  # IP address 2001:210:110:11::2/64 default gateway 2001:210:110:11::1

4.4.4. Configuration of IPv6-in-4 on Fedora, OpenSUSE and Ubuntu

Note: These Linux based operating systems had the same settings for each mechanism except for administrator privilege:

Configuration for Router 1:

- To create IPv6-in-4 tunnel a sit interface is used:
  # ip tunnel add sit1 local 2.16.0.16 remote 2.16.1.1
- To enable and start up a sit1 interface:
  # ip link set sit1 up
- To configure IPv6 address in a sit1 interface:
  # ip addr add address 2001:210:101:1::1/64 dev sit1
- To configure static routing for 6in4 tunnel interface:
  # ip -6 route add 2001::/16 dev sit1
  # ip route add default via 2001:210:101:1::2 dev sit1
- To enable packet forwarding:
  # sysctl –w net.ipv6.conf.default.forwarding=1

Configuration for Router 2:

- To create IPv6-in-4 tunnel a sit interface is used:
  # ip tunnel add sit1 local 2.16.1.1 remote 2.16.0.16
To enable and start up a sit1 interface:
   # ip link set sit1 up

To configure IPv6 address in a sit1 interface:
   # ip addr add address 2001:210:101:1::2/64 dev sit1

To configure static routing for 6in4 tunnel interface:
   # ip -6 route add 2001::/16 dev sit1
   # ip route add default via 2001:210:101:1::1 dev sit1

To enable packet forwarding:
   # sysctl –w net.ipv6.conf.default.forwarding=1

4.4.5. Configuration of IPv6-to-4 on Fedora, OpenSUSE and Ubuntu

Configuration for Router 1:

- To create IPv6-to-4 tunnel:
  # ip tunnel add tun6to4 mode local 2.16.0.16 remote 2.16.1.1

- To enable and start up a 6to4 tunnel interface:
  # ip link set dev tun6to4 up

- To configure IPv6 address in a 6to4 tunnel interface:
  # ip addr add address 2001:210:10:1::1/64 dev tun6to4

- To configure static routing for 6in4 tunnel interface:
  # ip -6 route add 2001::/16 dev tun6to4
  # ip route add default via 2001:210:101:1::1 dev tun6to4

- To enable packet forwarding:
  # sysctl –w net.ipv6.conf.default.forwarding=1

Configuration for Router 2:

- To create IPv6-to-4 tunnel:
  # ip tunnel add tun6to4 mode local 2.16.1.1 remote 2.16.0.16
To enable and start up a 6to4 tunnel interface:
   # ip link set dev tun6to4 up

To configure IPv6 address in a 6to4 tunnel interface:
   # ip addr add address 2001:210:101:1::1/64 dev tun6to4

To configure static routing for 6in4 tunnel interface:
   # ip -6 route add 2001::/16 dev tun6to4
   # ip route add default via 2001:210:10:1::1 dev tun6to4

To enable packet forwarding:
   # sysctl –w net.ipv6.conf.default.forwarding=1

4.4.6. Configuration of Dual-Stack on Fedora, OpenSUSE and Ubuntu

Configuration for Router 1:

IPv4 interface commands:

- To add IPv4 address
  # Ifconfig eth0 2.16.2.2 netmask 255.0.0.0

- To add default gateway
  # Route add default gw 2.16.1.1 eth0

- IP packet forwarding
  # Echo 1 > /proc/sys/net/ipv4/ip_forward

IPv6 interface commands:

- To add IPv6 address
  # IP addr add 2001:210:110:11::1/64 dev eth0

- To add default gateway
  # Ip route add default via 2001:210:110:11::2 dev eth0

- IP packet forwarding
  # Sysctl –w net.ipv6.conf.default.forwarding=1
Configuration for Router 2:

**IPv4 interface commands:**

- To add IPv4 address
  
  ```
  # Ifconfig eth0 2.16.1.1 netmask 255.0.0.0
  ```
- To add default gateway
  
  ```
  # Route add default gw 2.16.2.2 eth0
  ```
- IP packet forwarding
  
  ```
  # Echo 1 > /proc/sys/net/ipv4/ip_forward
  ```

**IPv6 interface commands:**

- To add IPv6 address
  
  ```
  # IP addr add 2001:210:110:11::2/64 dev eth0
  ```
- To add default gateway
  
  ```
  # Ip route add default via 2001:210:110:11::1 dev eth0
  ```
- IP packet forwarding
  
  ```
  # Sysctl –w net.ipv6.conf.default.forwarding=1
  ```

4.5. **Chapter Summary**

This chapter included the experimental network setup design and network test-bed diagram. It explained the structure of the network which was used to obtain the results. It also included network and IP transition mechanism configurations, as well as software & hardware specifications. Next chapter describes the data analysis of the VoIP and video tests obtained from these experiments.
Chapter 5: Data Analysis

This chapter covers analysis of the results gathered from experiments. This chapter contains 3 different sections that cover the results. First part includes performance of VoIP on three different IP transition mechanisms using five different platforms. Second part involves performance of VoIP on both IP versions and three IP transition mechanisms. Final part contains video over IP on IP transition mechanisms. The analyses of the data are presented in this chapter as area, bar graphs and tables. The parameters measured in the experiments were, RTT (Round Trip Time), jitter, packet loss, throughput, impacted-throughput, actual-throughput and CPU utilisation.

5.1. Performance of VoIP on IP Transition Mechanisms using 5 Platforms

This is a first part of data analysis chapter and this section presents performance of VoIP on three various IP transition mechanisms known as IPv6-to-4, IPv6-in-4 and Dual-Stack. These mechanisms were configured on five different operating systems, which were Fedora 14, Ubuntu 10.10, OpenSuse 11.3, Windows Server 2008 and Windows 7 and their average performance results were compared. The traffic load measured on these platforms comprised of 200 simultaneous calls using five different VoIP CODECS namely G.711.1, G.711.2, G.723.1, G.729.2 and G.729.3.

The results for delay and jitter were analysed in milliseconds (ms) and are presented in bar charts. Packet loss was calculated as a percentage (%) and is presented in bar graphs. However, throughput was calculated in Mega bits per second (Mbps) and is presented in tables. The raw data of these measurements are placed at the end of this thesis (Appendix A, B & C).
5.2. Results for Delay

This section covers results for delay (RTT). The results analysed for RTT on three IP transition mechanisms using five different platforms are presented in bar charts.

5.2.1. Delay for G.711.1 codec over IP transition mechanisms

Figure 5-1 below illustrates the delay measured for G.711.1 codec using three IP transition mechanisms on five different platforms.

![Figure 5-1: Delay on five platforms using G.711.1 codec](chart)

Performance measured for G.711.1 codec shows that Windows 7 OS using all three IP transition mechanisms had produced lower amount of delay than three other operating systems (OpenSuse, Ubuntu and Windows Server 2008). Performance calculated on Fedora OS shows that 6to4 and 6in4 produced delay below 0.8 milliseconds while Dual-Stack had a considerable delay of 6.5 milliseconds. Highest delay was observed on Windows Server 2008 using 6in4 tunnel at approximately 7.7 milliseconds. Second highest delay was measured on Ubuntu OS using Dual-Stack at approximately 7.3 milliseconds. Lowest delay was observed on Fedora OS using 6in4 tunnel at
approximately 1.1 milliseconds. Performance noticed on OpenSuse OS shows that three of the IP transition mechanisms had marginally close delay approximately 5.9 milliseconds.

Comparisons between IP transition mechanisms indicate that Dual-Stack performed slightly better than other two IP transition mechanisms using OpenSuse, Windows Server 2008 and Windows 7 operating systems. However, 6in4 performed marginally better on Fedora and Ubuntu operating systems.

Moreover it seems that these IP transition mechanisms have impact on different platforms. As seen in Figure 5-1 above that Windows 7 OS using all three IP transition mechanisms provided delay averaged below 1.7 milliseconds while OpenSuse OS had average results for three IP transition mechanisms ranging from 5.9 milliseconds to 6.0 milliseconds. Overall there is approximately 4.1 milliseconds difference, which is a high amount of additional delay from the voice quality perspective.

### 5.2.2. Delay for G.711.2 codec over IP transition mechanisms

Figure 5-2 below illustrates the delay measured for G.711.2 codec using three IP transition mechanisms on five different platforms.

![Figure 5-2: Delay on five platforms using G.711.2 codec](image-url)
The results obtained from tests and presented in this graph above shows that 6to4 tunnel had lower delay than other two IP transition mechanisms on all operating systems tested except for OpenSuse OS. Second lowest delay was measured on 6in4 tunnel on all operating systems except for Windows 7 OS.

Least amount of delay was measured on Windows 7 operating system, where three of the IP transition mechanisms had produced almost the same amount of delay ranging between 1.1 to 1.2 milliseconds.

Highest delay was measured on Ubuntu operating system using Dual-Stack mechanism at approximately 4.7 milliseconds.

Windows based operating systems performance showed that Windows Server 2008 OS had produced delay ranging between 4.1 to 4.5 milliseconds for all IP transition mechanisms tested. However, difference measured between Windows Server 2008 OS and Windows 7 OS is approximately 3.3 milliseconds.

Linux based operating systems indicate that Fedora OS using 6to4 and 6in4 had less delay below 0.8 milliseconds while Dual-Stack had greater delay at approximately 3.6 milliseconds. Comparison between all three Linux based platforms showed that Ubuntu OS had the worst performance using three IP transition mechanisms as it produced higher delay ranging between 4.0 to 4.7 milliseconds.

Furthermore, it was observed from the above results that Windows 7 and Fedora operating systems fares better with 6to4 and 6in4 tunnelling mechanisms, while other three Windows Server 2008, OpenSuse and Ubuntu OSs provided greater delay for VoIP communication. Dual-Stack mechanism performed marginally better than other two mechanisms (6to4 & 6in4) on all operating systems except for Fedora OS. It is clear that Dual-Stack does not perform as well as other two mechanisms (6to4 & 6in4) on Fedora OS because it produced 3.6 milliseconds more delay than other two mechanisms. Moreover Dual-Stack is causing marginally extra delay because of processing both IP addresses at the same time, which requires slightly more processing power than normal.
5.2.3. Delay for G.723.1 codec over IP transition mechanisms

Figure 5-3 below illustrates the delay measured for G.723.1 codec using three IP transition mechanisms on five different platforms (as before).

The results observed for G.723.1 codec illustrate that Dual-Stack mechanism provided lowest RTT on Windows 7 and Ubuntu operating systems while highest amount of RTT was on Fedora and Windows Server 2008 operating systems.

Least amount of RTT was measured on Window 7 OS using Dual-Stack at approximately 0.6 milliseconds closely followed by 6to4 and 6in4 mechanisms which was approximately 0.7 milliseconds.

Highest amount of RTT was calculated on Dual-Stack using OpenSuse and Windows Server 2008 operating systems at approximately 2.1 milliseconds.

IP transition mechanisms on Windows based operating systems indicate that Windows 7 OS can provide lesser delay than Windows Server 2008 OS using all three IP transition mechanisms.
Performance of IP transition mechanisms on Linux based operating systems indicated that Fedora OS is much better than OpenSuse OS and Ubuntu OS from RTT point of view.

5.2.4. Delay for G.729.2 codec over IP transition mechanisms

Figure 5-4 below illustrates the delay measured for G.729.2 codec using three IP transition mechanisms on five different platforms.

The observation for G.729.2 codec shows that IP transition mechanisms tested on Windows 7 OS are better than other four operating systems.

Least amount of delay was calculated on Fedora OS using 6to4 and 6in4 mechanisms and both produced almost same amount of delay at approximately 0.7 milliseconds.

Highest amount of delay was observed on OpenSuse OS using 6to4 tunnelling mechanism at approximately 4.7 milliseconds while second highest was calculated on Windows Server 2008 OS using Dual-Stack at approximately 4.5 milliseconds.
Comparison between IP transition mechanisms show that 6in4 marginally performed better than other two IP transition mechanisms on OpenSuse, Ubuntu and Windows Server 2008 operating systems.

Overall it is clear from these results that IP transition mechanisms have different impact on each operating system. Moreover, all Windows based operating systems provided different results. Measurement over Windows 7 OS performed far better than the performance measured on Windows Server 2008 OS.

5.2.5. Delay for G.729.3 codec over IP transition mechanisms

Figure 5-5 below illustrates the delay measured for G.729.3 codec using three IP transition mechanisms on five different platforms.

![Figure 5-5: Delay on five platforms using G.729.3 codec](image)

The bar graph above illustrates the results of G.729.3 codec. Performance measured for G.729.3 codec shows that 6to4 mechanism performed marginally better on OpenSuse, Ubuntu and Windows Server 2008 OSs. Second best performance was
observed with 6in4 mechanism on all operating systems except for Windows 7 operating system.

Lowest amount of delay for G.729.3 codec was calculated on Windows 7 OS using Dual-Stack mechanism at approximately 0.7 milliseconds while highest amount of delay was measured on OpenSuse, Ubuntu and Windows Server 2008 OSs using Dual-Stack at approximately 2.2 milliseconds.

Comparison among operating systems show that Windows 7 OS performed better than other four operating systems. Windows 7 OS using three IP transition mechanisms produced RTT ranging from 0.7 to 0.8 milliseconds while OpenSuse, Ubuntu and Windows Server 2008 OSs produced RTT ranging from 1.8 to 2.0 milliseconds. Overall difference identified between Windows 7 OS and other three operating systems (OpenSuse, Ubuntu and Windows Server 2008) was approximately 1.0 millisecond.

5.3. Results for Jitter

In this section results for jitter are presented in bar charts. The results analysis compare jitter on three IP transition mechanisms using five different platforms.

5.3.1. Jitter for G.711.1 codec over IP transition mechanisms
Figure 5-6 below illustrates the jitter calculated for G.711.1 codec using three IP transition mechanisms on five different platforms.
Performance measured for G.711.1 codec on three IP transition mechanisms showed that 6to4 tunnelling mechanism provided marginally lesser jitter than other IP transition mechanisms using three operating systems (OpenSuse, Ubuntu and Windows Server 2008).

Highest amount of jitter was observed on Windows Server 2008 using 6in4 tunnelling mechanism at approximately 2.3 milliseconds. Second highest amount of jitter was calculated on Dual-Stack using Ubuntu operating system at approximately 2.1 milliseconds. Lowest amount of jitter was measured on Fedora operating system using 6in4 tunnelling mechanism at approximately 0.5 milliseconds.

Comparison between operating systems showed that Windows 7 OS had the least amount of jitter over the three IP transition mechanisms in the range of 0.5 to 0.6 milliseconds.

Three IP transition mechanisms comparison was observed and the results obtained show that Dual-Stack had the highest jitter from the three IP transition mechanisms on all operating systems except for Windows 7 OS.
5.3.2. Jitter for G.711.2 codec over IP transition mechanisms

Figure 5-7 below illustrates the jitter calculated for G.711.2 codec using three IP transition mechanisms on five different platforms.

![Graph showing jitter on five platforms using G.711.2 codec](image)

The results obtained for G.711.2 codec specify that 6to4 mechanism produced slightly lesser jitter than other IP transition mechanisms tested on all operating systems except OpenSuse OS. Dual-Stack performance measured shows that it produced highest amount of jitter on all platforms besides Windows 7 OS.

Lowest jitter was calculated on Fedora operating system using 6to4 mechanism at approximately 0.4 milliseconds. However, highest amount of jitter was identified on Ubuntu operating system using Dual-Stack at approximately 2.2 milliseconds.

Overall Windows 7 OS performance was much better than other operating systems for all the IP transition mechanisms tested. It provided least amount of jitter ranging from 0.4 to 0.5 milliseconds. Moreover, second best performance was measured on Fedora operating system.
5.3.3. Jitter for G.723.1 codec over IP transition mechanisms

Figure 5-8 below illustrates the jitter calculated for G.723.1 codec using three IP transition mechanisms on five different platforms.

![Figure 5-8: Jitter on five platforms using G.723.1 codec](image)

Figure 5-8 illustrated above shows that Windows 7 OS performance was marginally better than other operating systems using three IP transition mechanisms. Windows 7 operating system results reveal that all three IP transition mechanisms had produced almost the same amount of jitter at approximately 0.4 milliseconds.

Least amount of jitter was tested on 6to4 mechanism using Fedora operating system at approximately 0.39 milliseconds. Highest jitter performance was calculated on Windows Server 2008 OS using Dual-Stack at approximately 1.1 milliseconds.

The quality difference measured between two Windows operating systems (Windows 7 & Windows Server 2008) showed that Windows 7 OS had 0.4 milliseconds jitter average while Server 2008 OS had jitter range between 0.9 to 1.1 milliseconds. Overall 6to4 and 6in4 using Fedora OS and Windows 7 OS had provided better and nearly comparable results.
5.3.4. Jitter for G.729.2 codec over IP transition mechanisms

Figure 5-9 below illustrates the jitter calculated for G.729.2 codec using three IP transition mechanisms on five different platforms.

The graph showed above presents the results of G.729.2 codec. It illustrates that 6to4 using OpenSuse OS had produced highest amount of jitter at approximately 2.3 milliseconds while lowest amount of jitter was measured on Fedora OS using 6to4 at approximately 0.45 milliseconds. Comparison between IP transition mechanisms indicate that 6in4 performance was much better on all the platforms tested.

Performance evaluated on Windows 7 OS shows that it had the best performance comparing to other 4 operating systems using three of the IP transition mechanisms. The range of jitter observed on Windows 7 OS using three IP transition mechanisms was between 0.4 to 0.46 milliseconds.

Dual-Stack results show that it performed marginally better on OpenSuse and Ubuntu OSs while 6to4 was second best, as it performed better on Fedora, Windows Server 2008 and Windows 7 operating systems.
Overall performance measured for 6to4 and 6in4 was marginally close on Windows 7 and Fedora operating systems. These two OSs also had the lower jitter than other three operating systems using 6to4 and 6in4 tunnelling mechanisms. However, using Dual-Stack on Fedora OS provided higher jitter at approximately 1.9 milliseconds.

5.3.5. Jitter for G.729.3 codec over IP transition mechanisms

Figure 5-10 below illustrates the jitter calculated for G.729.3 codec using three IP transition mechanisms on five different platforms.

![Figure 5-10: Jitter on five platforms using G.729.3 codec](image)

The G.729.3 codec performance over five operating systems using three IP transition mechanisms show that 6to4 performed much better on OpenSuse, Ubuntu and Windows Server 2008 OSs comparing to other 2 IP transition mechanisms. Second best IP transition mechanism was 6in4 as it performed better on all operating systems.

Highest jitter was observed on Windows Server 2008 OS using Dual-Stack at approximately 0.7 milliseconds while second highest amount of jitter was identified on Dual-Stack using Ubuntu OS at approximately 0.69 milliseconds. However, lowest
amount of jitter was measured on Windows 7 OS using Dual-Stack at approximately 0.34 milliseconds.

Comparison between operating systems shows that Windows 7 OS had much better performance than other four operating systems while Fedora OS was second best. Performance calculated on Windows Server 2008 OS was the worst as it had the highest amount of jitter for three of the IP transition mechanisms tested.

Overall, Windows 7 OS had stable results for jitter on three IP transition mechanisms tested.

5.4. Results for Packet Loss

This section presents packet loss over three IP transition mechanisms using five VoIP CODECS. Performance measured for five different CODECS using three IP transition mechanisms on five different platforms. The results are presented as bar graph in sections below.

5.4.1. Packet loss for G.711.1 codec over IP transition mechanisms

Figure 5-11 below illustrates the packet loss analysed for G.711.1 codec using three IP transition mechanisms on five different platforms.
Figure 5-11 above illustrates the results of packet loss for G.711.1 codec. It shows that Dual-Stack had least amount of packet loss using all operating systems tested. IPv6in4 tunnel had lesser amount of packet loss than 6to4 tunnel except for Windows Server 2008 OS.

Highest packet loss was observed on Windows Server 2008 OS using 6in4 tunnelling mechanism at approximately 0.01%. Second highest packet loss was calculated on Windows 7 OS using 6to4 tunnel at approximately 0.008%. However, lowest amount of packet loss was calculated on Fedora OS using Dual-Stack at approximately 0.00075%.

Best operating system performance was measured on OpenSuse OS where all three IP transition mechanisms had lower packet losses, ranging between 0.0012 to 0.0022%. Second best performance was observed on Fedora operating system, where packet loss ranged between 0.00075 to 0.0027%. Performance measured over Windows 7 OS indicates that Windows 7 OS had the worst performance compared to other four operating systems.
5.4.2. Packet loss for G.711.2 codec over IP transition mechanisms

Figure 5-12 below illustrates the packet loss analysed for G.711.2 codec using three IP transition mechanisms on five different platforms.

The graph showed above in Figure 5-12 presents the results of G.711.2 codec. It depicts that Dual-Stack had better performance than other two IP transition mechanisms tested. It also illustrates that 6to4 tunnel performed better than 6in4 tunnel using OpenSuse OS and Windows Server 2008 OS whereas 6in4 tunnel performed better than 6to4 tunnel on Fedora and Ubuntu operating systems.

Best performance was measured on Windows 7 OS where both 6to4 and 6in4 had the same amount of packet loss at approximately 0.0012%. Windows Server 2008 OS performance was not efficient as it had the highest packet loss on three of the IP transition mechanisms.

Highest packet loss was observed on Windows Server 2008 OS using 6in4 tunnel at approximately 0.006% while second highest packet loss was also on Windows Server 2008 OS using 6to4 tunnel at approximately 0.003%. However, lowest amount of packet
loss was measured on Dual-Stack using three operating systems (Fedora, OpenSuse and Windows 7) at approximately 0.00075%.

5.4.3. Packet loss for G.723.1 codec over IP transition mechanisms

Figure 5-13 below illustrates the packet loss analysed for G.723.1 codec using three IP transition mechanisms on five different platforms.

Figure 5-13: Packet loss on five platforms using G.723.1 codec

Figure 5-13 above illustrates the results of packet loss for G.723.1 codec. It shows that there is a significant packet loss on each operating system tested over three of the IP transition mechanisms.

Highest packet loss was measured on Windows 7 OS using 6in4 tunnelling mechanism at approximately 0.00375% while lowest packet loss was observed on Windows Server 2008 OS at approximately 0.0001%.

Performance among operating systems showed that Fedora OS had performed better than other operating systems using 6to4 tunnelling mechanism. The OpenSuse OS and
Ubuntu OS had same amount of packet loss for 6to4 and 6in4 tunnelling mechanism at approximately 0.00225.

Performance comparison for IP transition mechanisms indicated that 6in4 had more packet loss than 6to4 and Dual-Stack on all operating systems except for Windows Server 2008 OS. Highest packet loss difference was observed between Dual-Stack and 6in4 using Windows 7 OS, where margin was 0.003%. Second highest packet loss difference was calculated on Windows 7 OS using 6in4 and 6to4 mechanisms at approximately 0.0015%.

5.4.4. Packet loss for G.729.2 codec over IP transition mechanisms

Figure 5-14 below illustrates the packet loss analysed for G.729.2 codec using three IP transition mechanisms on five different platforms.

![Packet loss graph](image)

The results observed for G.729.2 codec are presented in the graph above. It illustrates that 6in4 tunnel on three of the operating systems have created higher amount of packet loss. Highest packet loss was measured on Fedora OS using 6to4 tunnelling
mechanism at approximately 0.0025% while same amount of packet loss was calculated on Ubuntu OS using 6in4 tunnelling mechanism. Least amount of packet loss was observed on OpenSuse OS using Dual-Stack at approximately 0.0075% and same amount of value was also observed on Windows 7 OS using Dual-Stack.

Performance comparison for IP transition mechanisms indicated that Dual-Stack had lesser packet loss than other two IP transition mechanisms on all operating systems except for Fedora OS. Second best performance was measured on 6to4 tunnelling mechanism using Ubuntu, Windows Server 2008 and Windows 7 platforms.

Performance comparison for operating systems shows that OpenSuse OS had better performance than other four operating systems over three IP transition mechanisms tested. Dual-Stack had lowest amount of packet loss on all operating systems tested except Fedora OS, while 6in4 and 6to4 are not the lowest but they are stable as both have provided same amount of packet loss at approximately 0.00175%.

5.4.5. Packet loss for G.729.3 codec over IP transition mechanisms

Figure 5-15 below illustrates the packet loss analysed for G.729.3 codec using three IP transition mechanisms on five different platforms.
The results measured for G.729.3 codec illustrated that IPv6in4 tunnel produced more packet loss than other two IP transition mechanisms tested on all five operating systems. Highest packet loss was calculated on Windows Server 2008 OS using 6in4 tunnelling mechanism at approximately 0.00425%. Lowest amount of packet loss was noticed on Ubuntu OS using Dual-Stack mechanism at approximately 0.00075%. Second lowest amount of packet loss was measured on Windows Server 2008 OS using Dual-Stack at approximately 0.001%.

Comparison among IP transition mechanisms showed that Dual-Stack marginally performed better than other two IP transition mechanisms on all five operating systems. IPv6to4 tunnelling mechanism had second best performance on all five operating systems.

Performance comparison among operating systems indicated that Ubuntu OS produced least amount of packet loss and had better performance than other four operating systems using all three IP transition mechanisms. Second best performance was calculated on Fedora OS as it provided lesser packet loss on three operating systems (Windows 7, Windows Server 2008 & OpenSuse).
5.5. Results for Throughput

This section presents throughput tested for five VoIP CODECS using three IP transition mechanisms on five different platforms. The results for throughput are analysed in Mbps (Mega bits per second) and are presented in a table.

5.5.1. Throughput for all 5 CODECS over IP transition mechanisms

Table 5-1 below illustrates the throughput (Mbps) for all five CODECS using three IP transition mechanisms on five different platforms.

<table>
<thead>
<tr>
<th>CODEC</th>
<th>6to4</th>
<th>Dual</th>
<th>6in4</th>
<th>6to4</th>
<th>Dual</th>
<th>6in4</th>
<th>6to4</th>
<th>Dual</th>
<th>6in4</th>
<th>6to4</th>
<th>Dual</th>
<th>6in4</th>
<th>6to4</th>
<th>Dual</th>
<th>6in4</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.723.1</td>
<td>1.46</td>
<td>1.46</td>
<td>1.46</td>
<td>1.46</td>
<td>1.46</td>
<td>1.46</td>
<td><strong>1.44</strong></td>
<td>1.46</td>
<td>1.45</td>
<td>1.45</td>
<td>1.47</td>
<td>1.46</td>
<td>1.46</td>
<td>1.46</td>
<td>1.47</td>
</tr>
<tr>
<td>G.729.2</td>
<td>2.01</td>
<td>2.05</td>
<td>2.01</td>
<td>2.05</td>
<td>2.058</td>
<td>2.05</td>
<td>2.02</td>
<td>2.05</td>
<td>2.05</td>
<td>2.03</td>
<td>2.03</td>
<td>2.06</td>
<td>2.00</td>
<td>2.02</td>
<td>1.996</td>
</tr>
<tr>
<td>G.729.3</td>
<td><strong>1.82</strong></td>
<td>1.85</td>
<td>1.81</td>
<td>1.85</td>
<td>1.857</td>
<td>1.856</td>
<td>1.83</td>
<td>1.84</td>
<td>1.84</td>
<td>1.83</td>
<td>1.85</td>
<td>1.86</td>
<td>1.84</td>
<td>1.85</td>
<td>1.84</td>
</tr>
</tbody>
</table>

Table 5-1: Throughput results for all 5 CODECS on five operating systems
Table 5-1 above shows the throughput measured on all five CODECS using five different operating systems on three various IP transaction mechanisms. Each codec produced different amount of throughput due to each having their fixed packet sizes. From the results, it was observed that the highest throughput is for the Windows 7 OS using codec G.711.1 over the 6to4 tunnel at approximately 12.38 Mbps. Least amount of throughput for G.711.1 codec was provided by Fedora OS using 6to4 tunnel at approximately 11.97 Mbps. Lowest throughput measured for G.711.2 codec was approximately 11.88 Mbps using Ubuntu OS on Dual-Stack. Highest throughput for G.711.2 codec was calculated on Windows Server 2008 at approximately 12.32 Mbps using IPv6in4 tunnelling mechanism.

For G.723.1 codec, Windows 7 OS and Windows Server 2008 OS using 6in4 tunnel had highest throughput at approximately 1.47 Mbps while lowest was on Ubuntu OS using 6to4 tunnel at approximately 1.44 Mbps. Performance evaluated for G.729.2 codec showed that the highest amount of throughput was measured on Windows Server 2008 OS using 6in4 tunnel at approximately 2.06 Mbps whereas lowest was on Windows 7 OS using 6in4 tunnel at approximately 1.996 Mbps. Finally the results for G.729.3 codec indicated that it has the lowest amount of throughput range comparing to other four CODECS. However, the highest throughput calculated for G.729.3 codec was approximately 1.857 Mbps using Dual-Stack on OpenSuse OS, while the lowest was approximately 1.82 Mbps using 6to4 tunnel on Fedora OS.

The performance over IP transition mechanisms show that Dual-Stack had more throughput than 6to4 and 6in4 using Fedora OS.

Overall it can be said that usage of Dual-Stack and two tunnelling mechanisms (6to4 & 6in4) have only slight impact on these five voice CODECS from the throughput perspective.
5.6. Comparison of VoIP on IP versions and IP Transition Mechanisms

This is a second part of data analysis chapter and it presents performance of voice over IP (VoIP) on both versions of IP (IPv4 & IPv6) and three IP transition mechanisms known as IPv6-to-4, IPv6-in-4 and Dual-Stack. In this experiment, pure IPv4 and pure IPv6 based networks were configured and the results obtained were compared against three IP transition mechanisms which were tested in first part (first part of data analysis showed). This part relates to performance comparison of VoIP over pure IP version 4 and IP version 6 with IP transition mechanisms using five voice CODECS. In this case two operating systems were selected such as Windows Server 2008 and Linux Ubuntu 10.10. Each operating system was setup to act as a router and client machines had Windows 7 OS installed during all tests. Furthermore, parameters considered were jitter, delay, packet loss and throughput. Traffic load tested was for 200 simultaneous VoIP calls using five different VoIP CODECS namely G.711.1, G.711.2, G.723.1, G.729.2 and G.729.3.

The results for jitter and delay were measured in milliseconds (ms) and are presented in bar charts. However, packet loss was measured as a percentage (%) while throughput was measured in Mega bits per second (Mbps) and there results have been averaged across all the range of observation for each metric measured. Furthermore packet loss results are presented in bar charts while throughput results are in table. The raw data for these measurements are placed at the end of this thesis (Appendix D, E & F).
5.7. Results for Delay

This section covers results for RTT (delay), which were obtained from the experiments and analysed for results. This section contains performance of VoIP CODECS on both IP versions (IPv4 & IPv6) and compared against three IP transition mechanisms using two operating systems (Windows Server 2008 & Linux Ubuntu 10.10). The results are presented in bar charts below.

5.7.1. Delay for G.711.1 codec using IP versions & IPTMs

Figure 5-16 below illustrates the delay calculated for G.711.1 using both versions of IP and three IP transition mechanisms.

![Codec G.711.1](chart.png)

Figure 5-16: Delay on IP versions & IP transition mechanisms using G.711.1 codec

Figure 5-16 shown above illustrates the results for delay measured over IP versions and IP transition mechanisms. Highest amount of delay using Windows OS was observed on IPv6in4 tunnel at approximately 7.7 milliseconds while lowest amount of delay was calculated on IPv4 at approximately 6.0 milliseconds. Highest delay on Linux OS was
noticed using IPv6 at approximately 7.4 milliseconds while closely followed by Dual-Stack at approximately 7.3 milliseconds. However, lowest amount of delay was measured on IPv6in4 tunnel using Linux OS at approximately 5.6 milliseconds.

Both versions of IP almost had the same amount of delay for both operating systems.

Performance of IP transition mechanisms showed that IPv6to4 was marginally better than IPv6in4 and Dual-Stack. IPv6in4 had higher delay for Windows OS while Dual-Stack had higher on Linux OS.

The outcome for operating systems indicated that Windows OS performed better than Linux OS on IPv4, IPv6 and Dual-Stack while Linux OS was better using IPv6to4 and IPv6in4 transition mechanisms.

5.7.2. Delay for G.711.2 codec using IP versions & IPTMs

Figure 5-17 below illustrates the delay calculated for G.711.2 using both versions of IP and three IP transition mechanisms.

![Codec G.711.2](image)

Figure 5-17: Delay on IP versions & IP transition mechanisms using G.711.2 codec
The results for G.711.2 codec are shown above in Figure 5-17. It shows that IPv4 had lesser delay than IPv6 using Windows operating system, while IPv6 performed marginally better than IPv4 using Linux operating system.

Highest RTT was observed on Dual-Stack and IPv4 using Linux OS at approximately 4.7 milliseconds while second highest RTT was measured on IPv6 using Linux OS at approximately 4.6 milliseconds. Least amount of RTT was calculated on IPv6to4 using Linux OS at approximately 4.0 milliseconds.

Comparison of RTT between IP transition mechanisms indicated that Dual-Stack mechanism had lower performance than other two tunnelling mechanisms on both operating systems. Dual-Stack over Windows OS produced approximately 4.4 milliseconds RTT while on Linux OS, it was 4.7 milliseconds. Performance difference identified between two tunnelling mechanisms showed IPv6to4 performed marginally better than IPv6in4 using both operating systems.

Performance comparison between Windows and Linux operating systems show that Windows performed better than Linux on IPv4, IPv6, IPv6in4 and Dual-Stack, whereas Linux OS performed marginally better on IPv6to4 tunnelling mechanism.

5.7.3. Delay for G.723.1 codec using IP versions & IPTMs

Figure 5-18 below illustrates the delay calculated for G.723.1 over both versions of IP and three IP transition mechanisms.
Figure 5-18: Delay on IP versions & IP transition mechanisms using G.723.1 codec

RTT results for G.723.1 codec show that IPv4 produced lesser amount of delay as compared to IPv6 using Windows OS while Linux OS was marginally better on IPv6.

Lowest amount of RTT was calculated on IPv4 using Windows OS at approximately 1.9 milliseconds while IPv6 had the second lowest RTT at approximately 1.92 milliseconds. Highest amount of RTT was measured on Dual-Stack using Windows OS at approximately 2.08 milliseconds while second highest was observed on IPv6to4 tunnel using Linux OS at approximately 2.07 milliseconds.

Comparison between Windows and Linux operating systems showed that Windows OS performed better than Linux on IPv4, IPv6, IPv6to4 and IPv6in4, whereas Linux OS was better on Dual-Stack mechanism.

Performance of three IP transition mechanisms indicated that IPv6in4 performed better than IPv6to4 and Dual-Stack. IPv6in4 had approximately 1.94 millisecond delay on Windows OS while IPv6to4 had 1.95 milliseconds delay. This only has 0.01 millisecond difference while comparison between Linux OS showed that the difference was 0.05 milliseconds.
5.7.4. Delay for G.729.2 codec using IP versions & IPTMs

Figure 5-19 below illustrates the delay calculated for G.729.2 using both versions of IP and three IP transition mechanisms.

![Codec G.729.2](image)

Figure 5-19: Delay on IP versions & IP transition mechanisms using G.729.2 codec

RTT outcome for G.729.2 codec indicates that IPv4 provided almost the same amount of delay as IPv6 using Windows OS at approximately 4.3 milliseconds. IPv4 using Linux OS was marginally better than IPv6 as it provided approximately 0.2 milliseconds lesser delay than IPv6.

Highest amount of RTT was measured on Dual-Stack using Windows OS at approximately 4.56 milliseconds whereas IPv6 using Linux OS had the second highest RTT at approximately 4.43 milliseconds. Lowest amount of RTT was measured on IPv6in4 using Linux OS at approximately 4.0 milliseconds while second lowest was observed on IPv4 using Linux OS at approximately 4.17 milliseconds.
Comparison between Linux and Windows operating systems indicated that Linux OS performed better than Windows OS on IPv4, IPv6in4 and Dual-Stack while Windows OS was better on IPv6 and IPv6to4 tunnelling mechanism.

Performance comparison among three IP transition mechanisms indicated that IPv6in4 performed better than IPv6to4 and Dual-Stack using Linux OS, as it provided approximately 0.3 milliseconds lesser delay. Dual-Stack had worst performance as compared to IPv6in4 and IPv6to4. As it provided approximately 3.80 milliseconds more delay on Windows OS while Linux OS was same as IPv6to4 at approximately 4.3 milliseconds. Overall IPv6in4 was marginally better than Dual-Stack and IPv6to4 while pure IPv4 was slightly better than pure IPv6.

5.7.5. Delay for G.729.3 codec using IP versions & IPTMs

Figure 5-20 below illustrates the delay calculated for G.729.3 using both versions of IP and three IP transition mechanisms.

![Figure 5-20: Delay on IP versions & IP transition mechanisms using G.729.3 codec](image-url)
The results for G.729.3 codec are visible in the graph above and it indicates that IPv4 using both operating systems (Linux & Windows) performed better than IPv6. Windows OS using IPv4 had approximately 0.15 milliseconds lesser delay than IPv6. Linux OS using IPv4 provided approximately 0.07 milliseconds lesser delay than IPv6.

Highest amount of delay was noticed on IPv6 using Windows OS at approximately 2.25 milliseconds while second highest RTT was observed on IPv6in4 at approximately 2.2 milliseconds. Least amount of delay was calculated on IPv6to4 using Windows OS at approximately 2.0 milliseconds whereas using Linux OS on IPv6to4 had the second lowest RTT at approximately 2.02 milliseconds.

Comparison analysis between Linux and Windows operating systems showed that Linux OS performed marginally better than Windows OS on IPv6, IPv6in4 and Dual-Stack while Windows OS was better on IPv4 and IPv6to4 tunnelling mechanism.

Performance analysis of three IP transition mechanisms indicated that IPv6to4 performed slightly better than IPv6in4 and Dual-Stack using both operating systems. Dual-Stack had the worst performance when compared to IPv6to4 and IPv6in4 as it had higher RTT over both operating systems at approximately 2.2 milliseconds. IPv6to4 had approximately 0.2 milliseconds lesser RTT than IPv6in4 using Windows OS. Overall IPv6to4 was marginally better than Dual-Stack and IPv6in4 whereas pure IPv4 was marginally better than pure IPv6 using both operating systems.
5.8. Results for Jitter

This section covers results analysis for jitter. This section contains performance of VoIP CODECS on both IP versions (IPv4 & IPv6) and three IP transition mechanisms over two operating systems (Windows & Linux). The results are presented in bar charts below for each of the CODECS tested.

5.8.1. Jitter for G.711.1 codec using IP versions & IPTMs

Figure 5-21 below illustrates the jitter measured for G.711.1 codec over both versions of IP and three IP transition mechanisms.

The results for G.711.1 codec are shown in the graph above. It indicates that IPv6in4 had the highest amount of jitter at approximately 2.4 milliseconds using Windows OS. Second highest jitter was observed on Dual-Stack at approximately 2.16 millisecond using Linux OS. Lowest amount of jitter was calculated on IPv6to4 tunnel at
approximately 1.8 milliseconds using Windows OS while second lowest jitter was measured on IPv6to4 using Linux OS at approximately 1.9 milliseconds.

Performance evaluation between two operating systems showed that Windows OS performed marginally better than Linux OS on IPv4, IPv6, IPv6to4 and Dual-Stack mechanism, while Linux OS performed better on IPv6in4 tunnel.

Comparison between two IP versions (IPv4 & IPv6) indicated that they both performed very close to each other. Both IP versions had approximately 2.0 milliseconds jitter using both operating systems (Windows & Linux).

Overall performance comparison among IP transition mechanisms showed that IPv6to4 marginally performed better than IPv6in4 and Dual-Stack mechanisms. Dual-Stack performed slightly better than IPv6in4 using Windows OS and provided approximately 0.4 milliseconds lesser jitter whereas results for Linux OS were opposite as Dual-Stack had approximately 0.2 milliseconds more jitter than IPv6in4.

### 5.8.2. Jitter for G.711.2 codec using IP versions & IPTMs

Figure 5-22 below illustrates the jitter measured for G.711.2 codec using both versions of IP and three IP transition mechanisms.

![Figure 5-22: Jitter on IP versions & IP transition mechanisms using G.711.2 codec](image)
The outcome for G.711.2 codec shows that IPv4 had produced highest amount of jitter at approximately 2.3 milliseconds using Linux OS. Second highest amount of jitter was noticed on Dual-Stack at approximately 2.28 milliseconds using Linux OS. Least amount of jitter was measured on IPv6to4 tunnel using Linux OS at approximately 1.9 milliseconds. However, using Windows OS it had provided second lowest jitter at approximately 1.97 milliseconds.

Performance analysis between Windows and Linux operating systems indicate that Windows had produced better results as compared to Linux OS. Windows OS performed slightly better on IPv4, IPv6, Dual-Stack and IPv6in4 tunnel. However, Linux OS marginally performed better with IPv6to4 tunnel and provided approximately 0.08 milliseconds lesser jitter than Windows OS.

Evaluation between IPv4 and IPv6 showed that IPv4 using Windows OS performed better than IPv6, while IPv6 using Linux OS performed better than IPv4. IPv4 using Windows OS had provided 0.12 milliseconds lesser jitter than IPv6 whereas IPv6 using Linux OS produced approximately 0.06 milliseconds lesser jitter than IPv4.

Furthermore, comparison between IP transition mechanisms indicated that IPv6to4 performed slightly better than Dual-Stack and IPv6in4 mechanisms. The worst performance was measured on Dual-Stack mechanism as it produced more jitter than two tunnelling mechanisms using both operating systems. IPv6in4 tunnel performance range was between IPv6to4 and Dual-Stack mechanisms using both operating systems. Moreover, IPv6to4 had provided approximately 0.21 milliseconds lesser jitter than IPv6in4 tunnel, while IPv6in4 had approximately 0.1 milliseconds lesser jitter than Dual-Stack mechanism using Linux OS.
5.8.3. Jitter for G.723.1 codec using IP versions & IPTMs

Figure 5-23 below illustrates the jitter measured for G.723.1 codec using both versions of IP and three IP transition mechanisms.

![Figure 5-23: Jitter on IP versions & IP transition mechanisms using G.723.1 codec](image)

The results analysed for G.723.1 codec indicates that IPv6 performed better than IPv4 using both operating systems. IPv6 using Linux OS had provided approximately 0.06 milliseconds lesser jitter while IPv6 using Windows OS had approximately 0.027 milliseconds lesser jitter than IPv4. Highest amount of jitter was noticed on Dual-Stack using Windows OS at approximately 1.08 milliseconds while second highest jitter was observed on IPv6to4 tunnel using Linux OS at approximately 1.05 milliseconds. Lowest value of jitter was calculated on IPv6in4 using Windows OS at approximately 0.93 milliseconds. However, IPv6 performed very close to IPv6in4 and provided approximately 0.01 milliseconds more jitter than IPv6in4 tunnel.

Comparison between two operating systems (Windows and Linux) showed that Windows OS provided better results than Linux OS on IPv4, IPv6, IPv6in4 and IPv6to4.
However, Linux OS performed better than Windows OS on Dual-Stack and provided approximately 0.05 milliseconds lesser jitter.

Performance comparison among IP transition mechanisms showed that Dual-Stack provided the highest jitter using Windows OS while IPv6to4 provided the highest jitter using Linux OS. IPv6in4 using Windows OS provided the least amount of jitter at approximately 0.93 milliseconds. These three IP transition mechanisms had almost the same amount of jitter using Linux operating system at approximately 1.05 milliseconds. The worst performance was measured on Dual-Stack as both operating systems had produced high amount of jitter as compared to other two tunnelling mechanisms. Best performance was observed on IPv6in4 as it had lower jitter than Dual-Stack and IPv6to4 tunnel. IPv6in4 provided approximately 0.16 milliseconds lesser jitter than Dual-Stack using Windows OS.

5.8.4. Jitter for G.729.2 codec using IP versions & IPTMs

Figure 5-24 below illustrates the jitter measured for G.729.2 codec using both versions of IP and three IP transition mechanisms.

![Jitter on IP versions & IP transition mechanisms using G.729.2 codec](image-url)
The graph shown above illustrates the results for G.729.2 codec. It indicates that IPv6in4 tunnel provided the least amount of jitter at approximately 1.84 milliseconds using Linux OS while IPv4 provided the second lowest amount of jitter at approximately 1.93 milliseconds using Linux OS. The highest amount of jitter was calculated on Dual-Stack at approximately 2.23 milliseconds using Windows OS while IPv6 provided the second highest amount of jitter using Linux OS at approximately 2.1 milliseconds.

The results analysis between two operating systems (Windows and Linux) indicated that Linux OS had better performance than Windows OS. The Linux OS performed marginally better on IPv4, IPv6in4 and Dual-Stack while Windows OS marginally performed better with IPv6 and IPv6to4 tunnel. However, both operating systems had only 0.1 milliseconds difference between them on both version of IPs and two tunnels while Dual-Stack had approximately 0.2 milliseconds.

Comparison between IPv4 and IPv6 indicated that IPv4 using Linux OS had better performance than IPv6. However, using Windows OS both IP versions (IPv6 and IPv4) provided almost the same amount of jitter at approximately 2.05 milliseconds. The performance difference identified between IPv4 and IPv6 was at approximately 0.22 milliseconds using Linux OS.

Another comparison among IP transition mechanisms showed that IPv6in4 tunnel had better results than IPv6to4 and Dual-Stack mechanisms. The second best performance was observed on IPv6to4 as it provided more jitter than IPv6in4 and lesser jitter than Dual-Stack at approximately 2.0 milliseconds using Windows OS. The highest jitter was identified on Dual-Stack using Windows OS at approximately 2.2 milliseconds. Dual-Stack performance wasn’t efficient as compared to two tunnelling mechanisms and it provided approximately 0.27 milliseconds more jitter than IPv6in4 using Windows OS.
5.8.5. Jitter for G.729.3 codec using IP versions & IPTMs

Figure 5-25 below illustrates the jitter measured for G.729.3 codec using both versions of IP and three IP transition mechanisms.

![Figure 5-25: Jitter on IP versions & IP transition mechanisms using G.729.3 codec](image)

The results for G.729.3 codec are illustrated above in Figure 5-25. It shows that IPv6to4 had the lowest jitter as compared to both versions of IP and two IP transition mechanisms over both operating systems. IPv6to4 using Linux OS had the least amount of jitter at approximately 0.6 milliseconds while second lowest amount of jitter was on IPv6to4 using Windows OS at approximately 0.61 milliseconds. The highest jitter value was identified on IPv4 and Dual-Stack using Windows OS and also on IPv6 using Linux OS at approximately 0.7 milliseconds. The second highest jitter value was observed on IPv6 using Windows OS and on Dual-Stack using Linux OS at approximately 0.69 milliseconds.

The performance difference between two operating systems (Windows and Linux) showed that Linux OS performed marginally better than Windows OS. Both operating systems performed almost the same on IPv6in4 tunnel at approximately 0.675
milliseconds. The Linux OS had better results on IPv4, IPv6to4 and Dual-Stack while Windows OS marginally performed better on IPv6.

The results difference between both versions of IP (IPv4 and IPv6) indicated that IPv4 using Linux OS had better performance than IPv6, as it provided approximately 0.028 milliseconds lesser jitter. However, IPv6 using Windows OS had better performance than IPv4 as it provided approximately 0.01 milliseconds lesser jitter.

The results analysis for IP transition mechanisms indicated that IPv6to4 tunnel had much better performance than IPv6in4 and Dual-Stack mechanisms using both operating systems. IPv6in4 had second best performance using both operating systems as it provided more jitter than IPv6to4 tunnel and lesser jitter than Dual-Stack at approximately 0.675 milliseconds. Dual-Stack had the worst performance as it provided the highest jitter values at approximately 0.7 milliseconds using Windows OS. Dual-Stack using Linux OS had slightly lesser jitter than Windows OS at approximately 0.01 milliseconds. Overall IPv6to4 tunnel had almost 0.1 milliseconds lesser jitter than Dual-Stack and approximately 0.075 milliseconds lesser than IPv6in4 tunnel.

5.9. Results for Packet loss

This section presents packet loss measured on IP versions and IP transition mechanisms using five VoIP CODECS. The results are presented in bar charts below:
5.9.1. Packet loss for G.711.1 codec over IP versions & IPTMs

Figure 5-26 below illustrates the jitter analysed for G.711.1 codec using both versions of IP and three IP transition mechanisms.

![Codec G.711.1](image)

As visible in Figure 5-26 above that G.711.1 codec didn’t have any packet loss using IPv4 with Windows operating system.

Highest amount of packet loss was calculated on IPv6in4 tunnel using Windows OS at approximately 0.004 percent. Second highest packet loss was observed on IPv6to4 tunnel using Linux OS at approximately 0.003 percent. IPv4 produced no packet loss at all while second lowest amount of packet loss was measured on IPv4 using Linux OS approximately 0.00025 percent.

Performance comparison between Windows and Linux operating systems showed that Windows OS had much better performance than Linux OS as it produced lesser packet loss on IPv4, IPv6, IPv6to4 and Dual-Stack.
The results measured for three IP transition mechanisms indicated that Dual-Stack performed better than two tunnelling mechanisms using Windows OS. However, Dual-Stack using Linux OS had similar performance as IPv6in4 at approximately 0.00175 percent. Moreover, IPv6to4 using Linux OS had approximately 0.003 percent packet loss. Overall it is clear that IPv4 is much better from packet loss perspective as it had very low packet loss. Furthermore Dual-Stack mechanism is slightly better than the two tunnelling mechanisms.

5.9.2. Packet loss for G.711.2 codec over IP versions & IPTMs

Figure 5-27 below illustrates the jitter analysed for G.711.2 codec using both versions of IP and three IP transition mechanisms.

Packet loss results for G.711.2 codec are shown above in Figure 5-27. It illustrates that IPv4 using Linux OS had no packet loss at all.
Performance measured and compared between Windows and Linux operating systems showed that Linux OS had lesser packet loss than Windows OS using IPv4, IPv6to4, IPv6in4 and Dual-Stack. However, Windows OS was marginally better on IPv6.

Highest packet loss was measured on IPv6in4 tunnel using Windows OS at approximately 0.02 percent while lowest amount of packet loss was noticed on IPv4 using Linux OS approximately 0 percent. Second lowest packet loss was observed on IPv4 using Windows OS at approximately 0.00025 percent.

Dual-Stack performance was much better than the two tunnelling mechanisms using both operating systems. As it produced 0.0015 percent packet loss on Windows OS while 0.001 percent packet loss on Linux OS, having 0.0005 percent difference. IPv6in4 using Windows OS produced significant packet loss at approximately 0.02 percent while Linux OS had 0.00125 percent, having 0.01875 percent more packet loss than Linux OS.

Two tunnelling mechanisms comparison showed that using Windows OS on IPv6to4 was much better than IPv6in4 as it produced 0.017 percent more packet loss. However, using Linux OS on both (IPv6to4 & IPv6in4) indicated that IPv6in4 had lesser packet loss than IPv6to4, having approximately 0.001 percent difference.

Furthermore, it can be concluded from the results shown above, that IPv4 performed better than IPv6, while Dual-Stack mechanism was stable on both operating systems (Windows & Linux) as compared to two tunnelling mechanisms.

5.9.3. Packet loss for G.723.1 codec over IP versions & IPTMs

Figure 5-28 below illustrates the jitter analysed for G.723.1 codec using both versions of IP and three IP transition mechanisms.
The outcome measured for G.723.1 codec shows that IPv4 had no packet loss using both operating systems (Windows & Linux). However, performance of IPv6 shows that it has produced packet loss using both operating systems.

Highest packet loss was calculated on IPv6in4 and IPv6to4 tunnel. This shows that IPv6to4 produced packet loss at approximately 0.00225 percent using Windows OS while same amount of packet loss was measured on IPv6in4 using Linux OS.

Performance difference between two operating systems indicated that Windows OS had no packet loss on IPv4 and Dual-Stack while Linux OS had no packet loss on IPv4. Overall Linux OS performed better than Windows OS on IPv6 and IPv6to4 whereas as Windows OS performed better than Linux OS on IPv6in4 and Dual-Stack mechanisms.

Comparison between IP transition mechanisms showed that Dual-Stack performed marginally better than other two tunnelling mechanisms. Performance analysis for IPv6to4 and IPv6in4 shows that IPv6in4 had better performance with Windows OS while IPv6to4 had better with Linux OS.
5.9.4. Packet loss for G.729.2 codec over IP versions & IPTMs

Figure 5-29 below illustrates the jitter analysed for G.729.2 codec using both versions of IP and three IP transition mechanisms.

Packet loss for G.729.2 codec shows that IPv4 using both operating systems (Windows & Linux) had no packet loss. Performance measured on IPv6 indicates that both operating systems provided almost same amount of packet loss at approximately 0.00025 percent.

Highest amount of packet loss was observed on IPv6in4 tunnel using Linux OS at approximately 0.0025 percent, while Windows OS on IPv6in4 had second highest packet loss at approximately 0.0022 percent.

Performance comparison between two operating systems showed that Windows OS and Linux OS provided equal amount of packet loss on IPv4, IPv6, IPv6to4 and Dual-Stack. However, slight difference was calculated on IPv6in4 where Windows OS had 0.0003 percent lesser packet loss than Linux OS.
Performance analysis between IP transition mechanisms indicated that Dual-Stack had the best performance comparing to both tunnelling mechanisms, as it provided approximately 0.001 percent lesser packet loss than IPv6to4 tunnel and at approximately 0.0005 percent lesser packet loss than IPv6in4 tunnel. Comparison between IPv6to4 and IPv6in4 indicates that IPv6to4 marginally performed better than IPv6in4 as it provided 0.0005 percent lesser packet loss.

5.9.5. Packet loss for G.729.3 codec over IP versions & IPTMs

Figure 5-30 below illustrates the jitter analysed for G.729.3 codec using both versions of IP and three IP transition mechanisms.

![Figure 5-30: Packet loss on IP versions & IP transition mechanisms using G.729.3](image)

Packet loss results for G.729.3 codec are visible in the graph above. It shows that IPv4 had zero percent packet loss using both operating systems (Windows & Linux). The results measured for IPv6 shows that it produced higher amount of packet loss for both operating systems. Windows OS produced approximately 0.0023 percent packet loss while Linux OS had little higher at approximately 0.0032 percent packet loss.
Highest amount of packet loss was calculated on IPv6in4 tunnel using Windows OS at approximately 0.0042 percent, whereas using Linux OS on IPv6 had the second highest amount of packet loss at approximately 0.0032 percent.

Comparison between Windows and Linux operating systems showed that Linux OS performed better than Windows OS on IPv6in4, IPv6to4 and Dual-Stack mechanisms while Windows OS was better on IPv6. Highest difference was measured between Windows and Linux OSs on IPv6in4 at approximately 0.0018 percent. Lowest difference was calculated between operating systems using Dual-Stack at approximately 0.0002 percent.

Comparison among IP transition mechanisms indicated that Dual-Stack performed better than both tunnelling mechanisms. It had packet loss range approximately 0.0008 to 0.001 percent. However, IPv6to4 performed better than IPv6in4, as it had packet loss range approximately 0.0017 to 0.0028 percent. Overall Dual-Stack was marginally better, as it had least amount of packet loss. It had approximately 0.0032 percent lesser packet loss than IPv6in4 tunnel using Windows OS.
5.10. Results for Throughput

This section covers throughput results, which were measured on both versions of IP and three IP transition mechanisms using five VoIP CODECS. The results were analysed in Mbps (Mega bits per second) and are presented in a table.

5.10.1. Throughput of 5 CODECS on IP versions & IP transition mechanisms

Table 5-2 below illustrates the throughput calculated for five CODECS using both versions of IP and three IP transition mechanisms.

<table>
<thead>
<tr>
<th>CODECS</th>
<th>IPv4 Win</th>
<th>IPv4 Linux</th>
<th>IPv6 Win</th>
<th>IPv6 Linux</th>
<th>Dual-Stack Win</th>
<th>Dual-Stack Linux</th>
<th>IPv6to4 Win</th>
<th>IPv6to4 Linux</th>
<th>IPv6in4 Win</th>
<th>IPv6in4 Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.723.1</td>
<td>1.46</td>
<td>1.45</td>
<td>1.46</td>
<td>1.45</td>
<td>1.45</td>
<td>1.46</td>
<td>1.45</td>
<td>1.44</td>
<td>1.47</td>
<td>1.45</td>
</tr>
<tr>
<td>G.729.2</td>
<td>2.04</td>
<td>2.04</td>
<td>2.09</td>
<td>2.04</td>
<td>2.03</td>
<td>2.06</td>
<td>2.04</td>
<td>2.02</td>
<td>2.06</td>
<td>2.05</td>
</tr>
<tr>
<td>G.729.3</td>
<td>1.84</td>
<td>1.83</td>
<td>1.86</td>
<td>1.84</td>
<td>1.85</td>
<td>1.84</td>
<td>1.83</td>
<td>1.83</td>
<td>1.86</td>
<td>1.83</td>
</tr>
</tbody>
</table>

Table 5-2: Throughput results for five VoIP CODECS

Table 5-2 above shows the throughput measured on all five CODECS using both versions of IP and three IP transition mechanisms. Each codec provided different throughput according to their packet sizes. However, comparison between Windows and Linux operating systems showed that throughput calculated over both operating systems is marginally close. The results for G.723.1, G.729.2 and G.729.3 CODECS showed that 0.01 to 0.05 Mbps difference was noticed on these three CODECS. However, G.711.1 and G.711.2 CODECS had higher difference between operating systems ranging between 0.1 to 0.39 Mbps.
The comparison between IPv4 and IPv6 showed that IPv6 provided higher throughput than IPv4.

The performance over IP transition mechanisms was different from both versions of pure IP (IPv4 & IPv6). Dual-Stack mechanism provided lesser throughput as compared to IPv6 using G.711.1 codec at approximately 0.5 Mbps for Windows OS while 0.38 Mbps for Linux OS.

IPv6to4 and IPv6in4 tunnelling mechanisms had insignificant impact on throughput for G.723.1, G.729.2 and G.729.3 CODECS while slight impact was noticed for G.711.1 and G.711.2 CODECS. Overall it can be said that usage of Dual-Stack and two tunnelling mechanisms have slight impact on these five voice CODECS from throughput perspective.
5.11. Performance of VVoIP & Impact of IPTMs on Video Protocols

This is a third part of data analysis’s chapter which includes two sections. First section presents performance of video over IP using three IP transition mechanisms. Second section covers impact of IP transition mechanisms on five different video protocols. The video protocols tested are MPEG-1, MPEG-2, MPEG-4, MKV and FLV and IP transition mechanisms selected were IPv6-to-4 tunnelling, IPv6-in-4 tunnelling and Dual-Stack mechanisms. These IP transition mechanisms were setup on Ubuntu 10.10 platform and metrics covered are actual-throughput, impacted-throughput and CPU utilisation. A two-way video conference was established between networks and results were obtained over IPv4, IPv6, IPv6to4, IPv6in4 and Dual-Stack. The outcome obtained is presented in two types of graphs such as bar and area.

**Actual-Throughput:** is original throughput of each video protocol, which was transmitted over pure IPv4 based networks and pure IPv6 based networks. It is a real throughput which is essential to establish a two-way video conference without any additional impact on actual throughput.

**Impacted-Throughput:** is additional bandwidth required over the original throughput due to encapsulation and de-capsulation process, which is vital in IPv6to4 and IPv6in4 tunnelling mechanisms. This additional throughput is a waste of bandwidth because it establishes a two-way video conference between users and requires more bandwidth in order for encapsulation and de-capsulation to operate during the transmission.
5.12. Performance of Video over IP using IP Transition Mechanisms

This section presents results of video over both versions of IP (IPv4 & IPv6) and three IP transition mechanisms. The five video protocols involved were MPEG-1, MPEG-2, MPEG-4, MKV and FLV. The results are analysed in Kilo bytes per second and are presented in area charts below.

5.12.1. MPEG-1 protocol

Figure 5-31 below illustrates performance of MPEG-1 on both versions of IP and three IP transition mechanisms. This enables viewing of the transition impact and the actual throughput at the same time.

![Figure 5-31: Performance of MPEG-1 on IP versions & IP transition mechanisms](image)

Figure 5-31 above illustrates the performance of MPEG-1 on IP versions and IP transition mechanisms. Throughput measured over IPv4 was approximately 248 kilobytes per second while IPv6 was approximately 257 kilobytes per second.
Dual-Sack has produced marginally close throughput to IPv6 at approximately 261 kilobytes per second. The impact on the bandwidth was very slight, due to both IP versions being processed concurrently in Dual-Stack mechanism.

IPv6to4 and IPv6in4 tunnelling mechanisms had higher impact on video traffic and wasted large amount of bandwidth while video traffic was being transmitted. IPv6to4 had better performance than IPv6in4 as it wasted 3 kilobytes per second lesser bandwidth than IPv6in4. In this case tunnelling mechanisms require additional bandwidth in order to process video traffic, as mentioned above; using IPv4 only uses 248 kilobytes per second bandwidth to provide 248 kilobytes per second throughput while usage of IPv6to4 tunnel used approximately 370 kilobytes per second in order to provide 248 kilobytes per second throughput. It is mainly due to the encapsulation and de-capsulation process due to the tunnelling mechanisms. Therefore it is clear that IPv6to4 and IPv6in4 caused significant impact on bandwidth and wasted approximately 110 kilobytes per second.

5.12.2. MPEG-2 protocol

Figure 5-32 below illustrates performance of MPEG-2 on both versions of IP and three IP transition mechanisms.

![Video Protocol MPEG-2](image)

Figure 5-32: Performance of MPEG-2 on IP versions & IP transition mechanisms
Performance of MPEG-2 shows that it has larger packet size compared to other protocols tested in this experiment. Throughput calculated on IPv4 was approximately 530 kilobytes per second while IPv6 had approximately 536 kilobytes per second.

The outcome for Dual-Stack is stable as it provided slightly higher throughput than IPv4 and IPv6 at approximately 540 kilobytes per second. Dual-Stack has slight impact on the bandwidth; however it is approximately 5 kilobytes per second. Large amount of difference was observed between Dual-Stack and two tunnelling mechanisms. IPv6to4 wasted significant amount of bandwidth at approximately 239 kilobytes per second more as compared to Dual-Stack. However, IPv6in4 had still more impact on the bandwidth than IPv6to4 and wasted approximately 4 kilobytes per second more than IPv6to4.

The results complied for IPv6in4 indicated that it had worst performance as it produced more bandwidth wastage. Overall Dual-Stack is better than two tunnelling mechanisms as it had the least impact on bandwidth since it caused very little bandwidth wastage. However, establishment of Dual-Stack will take several years before it is ready to be used around the world. Thus two tunnelling mechanisms will be used in the interim period. However, the bandwidth wastage caused by these mechanisms will require at least double the amount of internet bandwidth in order to have video communication over internet. Performance of IPv6to4 was marginally better than IPv6in4 as it caused lesser impact on bandwidth.

5.12.3. MPEG-4 protocol

Figure 5-33 below illustrates performance of MPEG-4 on both versions of IP and three IP transition mechanisms.
The results observed for MPEG-4 show that it has smaller packet size than MPEG-1 and MPEG-2. Throughput performance tested on MPEG-4 using IPv4 provided approximately 107 kilobytes per second while IPv6 provided 123 kilobytes per second.

Dual-Stack performance indicates that it used approximately 4 kilobytes per second more bandwidth than IPv6 to establish a two-way video conference. IPv6to4 tunnel used approximately 175 kilobytes per second bandwidth to deliver approximately 123 kilobytes per second throughput. Overall it wasted approximately 52 kilobytes per second. The outcome for IPv6in4 shows that it used 194 kilobytes per second bandwidth in order to deliver 123 kilobytes per second throughput. Overall it wasted approximately 71 kilobytes per second bandwidth.

Comparison between two tunnelling mechanisms showed that IPv6in4 tunnel wasted more bandwidth than IPv6to4 tunnel at approximately 20 kilobytes per second. Furthermore, it can be said that usage of IPv6to4 is much better than IPv6in4 for MPEG-4 protocol as it had lesser impact compared to IPv6to4.
5.12.4. MKV protocol

Figure 5-34 below illustrates performance of MKV on both versions of IP and three IP transition mechanisms.

![Figure 5-34: Performance of MKV on IP versions & IP transition mechanisms](image)

The results for MKV protocol revealed that IPv4 used approximately 251 kilobytes per second bandwidth to establish a two-way video communication while IPv6 used higher bandwidth at approximately 253 kilobytes per second.

Outcome over Dual-Stack indicates that it used 265 kilobytes per second bandwidth in order to deliver a two-way video conference. It used almost 10 kilobytes per second more bandwidth than IPv4 and IPv6. This additional use of bandwidth was due to the processing of both IP stacks by the system simultaneously.

Highest impact on bandwidth was identified using IPv6in4 tunnelling mechanism as it used approximately 383 kilobytes per second whereas IPv6to4 was marginally close as it used approximately 380 kilobytes per second to establish a two-way video conference. IPv6in4 tunnel wasted approximately 130 kilobytes per second bandwidth while IPv6to4 tunnel had 3 kilobytes per second lesser impact on bandwidth than IPv6in4 tunnel.
Overall MKV performance measured shows that it was much better than MPEG-1 and MPEG-2 as it had lesser impact on IP transition mechanisms. However performance of Dual-Stack on MKV had higher impact than the Dual-Stack on MPEG-1 and MPEG-2 protocols.

5.12.5. FLV protocol

Figure 5-35 below illustrates performance of FLV on both versions of IP and three IP transition mechanisms.

The results for FLV protocol are shown in the graph above. It is visible from Figure 5-35 above that FLV has smaller packet size than the previous four protocols tested in the experiment. Performance over pure IPv4 was calculated at approximately 32.7 kilobytes per second while pure IPv6 had 32.9 kilobytes per second. The difference identified between IPv4 and IPv6 was insignificant at approximately 0.2 kilobytes per second.

The outcome measured over Dual-Stack was slightly higher than both versions of IP. It used approximately 33.8 kilobytes per second to establish a two-way video conference.
Usage of FLV over Dual-Stack is reasonable as it only wasted 1.1 kilobytes per second bandwidth.

Comparison between the two tunnelling mechanisms showed that IPv6in4 used 49.9 kilobytes per second bandwidth to deliver a two-way video conference while IPv6to4 used almost the same amount of bandwidth at approximately 50 kilobytes per second to deliver a two-way video conference. The impact calculated between IP versions and tunnelling mechanisms indicated that tunnelling mechanisms wasted approximately 18 kilobytes per second bandwidth. Usage of 18 kilobytes per second more bandwidth than IP versions, which is the least amount of bandwidth wastage as compared to other four video protocols, tested namely MPEG-1 MPEG-2 MPEG-4 and MKV (visible in Figure 5-31, 5-32, 5-33 & 5-34 above).

5.13. Actual-throughput versus Impacted-throughput

This section presents tests for Impacted-throughput and Actual-throughput. These were conducted in order to differentiate the performance of pure IPv6 and IPv6 with three IP transition mechanisms (IPv6to4, IPv6in4 & Dual-Stack) using five well-known video protocols (MPEG-1 MPEG-2 MPEG-4, MKV and FLV). The values were analysed in Kilobytes per second and are presented in bar charts below.

5.13.1. Dual-Stack

Figure 5-36 below illustrates performance of five video protocols on IPv6 and Dual-Stack mechanism.
Figure 5-36: Actual-throughput VS impacted-throughput over Dual-Stack

As visible in Figure 5-36 above that actual-throughput was observed over IPv6 networks while impacted-throughput was observed over Dual-Stack mechanism using five video protocols. Overall it is clear that slight impact was measured over Dual-Stack comparing to its counterpart using all five protocols. MPEG-1 had approximately 4 kilobytes per second difference between IPv6 and Dual-Stack. The 4 kilobytes per second impacted-throughput was identified as it wasted the bandwidth. Dual-Stack on MPEG-2 had approximately 3.6 kilobytes per second impacted-throughput.

Least amount of impact was discovered on FLV protocol at approximately 0.9 kilobytes per second while highest impact was identified on MKV at approximately 15 kilobytes per second. Performance of MPEG-4 was reasonable as it produced 3.8 kilobytes per second impact.

It can be concluded from these results that usage of a Dual-Stack mechanism will have only a small amount of impact on video protocols, in other words Dual-Stack will have slight bandwidth wastage during video traffic transmission. Moreover, protocols like FLV will have very slight impact on video streaming at approximately 0.4 kilobytes per second, which is reasonable for video streaming applications.
5.13.2. IPv6-to-4 tunnel

Figure 5-37 below illustrates performance of five video protocols on IPv6 and IPv6-to-4 mechanism.

![Graph Showing Throughput Comparison]

The results for IPv6 and IPv6to4 tunnel were measured and compared in the graph shown above. It is clear that IPv6to4 had impacted each video protocol significantly. The performance difference discovered between pure IPv6 and IPv6to4 tunnel on MPEG-1 shows that IPv6to4 caused approximately 106 kilobytes per second impact on the bandwidth. MPEG-2 test results indicated that it had been impacted by approximately 242 kilobytes per second, which is even more than MPEG-1.

Lowest impact was measured for FLV protocol at approximately 17 kilobytes per second whereas MPEG-4 was second best as it had approximately 51 kilobytes per second impact on bandwidth. The results of MKV were even worse than MEPG-1 as it impacted the bandwidth by approximately 136 kilobytes per second which is 30 kilobytes per second more than MPEG-1.

Overall FLV was least impacted and performed well on IPv6to4 tunnel while MPEG-4 was second best.
5.13.3. IPv6-in-4 tunnel

Figure 5-38 below illustrates performance of five video protocols on pure IPv6 and IPv6-in-4 mechanism.

The results for IPv6in4 were obtained using five different video protocols and compared with pure IPv6 as shown in Figure 5-38 above. It can be concluded that IPv6in4 has significant impact on throughput using all five protocols tested.

Highest impact was observed using MPEG-2 protocol at approximately 246 kilobytes per second while lowest was measured using FLV at approximately 17.1 kilobytes per second. MKV performance analysis shows that it had the second highest impact at approximately 126 kilobytes per second. MPEG-1 performance evaluation indicates that it was impacted at approximately 110 kilobytes per second; however it was much better than MPEG-2 and MKV protocols as these had considerable impact on throughput.

Overall the best performance on IPv6in4 was observed on FLV and second best performance was calculated on MPEG-4 protocol as it provided 70 kilobytes per second impact, which was much better than MPEG-1, MPEG-2 and MKV protocols.
5.14. CPU Utilisation

This section covers results for CPU utilisation. These tests were measured on “Router 1” during transmission of each video protocol (MPEG-1, MPEG-2, MPEG-4, MKV and FLV) over both versions of IP and three IP transition mechanisms. The results were analysed as percentage (%) CPU utilisation and are presented in the table below.

5.14.1. Impact of IP transition mechanisms on CPU processing power

The results observed from CPU utilisation are compiled in a Table 5-3 below. Each video protocol tested indicates different range of CPU processing power usage over both IP versions and IP transition mechanisms. The variation among five protocols was noticed and compared in the table below.

<table>
<thead>
<tr>
<th>Protocols</th>
<th>CPU Utilisation%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IPv4</td>
</tr>
<tr>
<td>MPEG-1</td>
<td>29.67</td>
</tr>
<tr>
<td>MPEG-2</td>
<td>31.42</td>
</tr>
<tr>
<td>MPEG-4</td>
<td>30.94</td>
</tr>
<tr>
<td>MKV</td>
<td>30.37</td>
</tr>
</tbody>
</table>

Table 5-3: CPU utilisation results for video protocols

FLV protocol had used least amount of CPU processing power while MPEG-4 had the highest usage of CPU processing power. FLV used approximately 26 % CPU processing power for IPv4, Dual-Stack, IPv6to4 and IPv6in4 while IPv6 used 23 %. MPEG-4 results showed that it used less CPU processing power for IPv4 and IPv6 ranged from 30.94 % to 31.46 %. However, it used more processing power over Dual-Stack, IPv6to4 and IPv6in4 mechanisms ranged from 31.73 % to 33.44 %.
The outcome for MKV showed that CPU processing power usage on an average ranged between 28.88 to 30.37 %. MPEG-1 and MPEG-2 provided different results than MPEG-4 as shown in the table above. MPEG-1 & MPEG-2 used less processing power over two tunnelling mechanisms while more processing power was used over IPv4 and IPv6. Furthermore, it is very clear that FLV had the best performance, as it used least amount of CPU processing power as compared to other four protocols tested.

5.15. Chapter Summary

This chapter presented data analysis from the tests conducted in this study. It has three different parts of data analysis. First part focused on performance of voice over IP using multiple platforms, while second part focused on voice over IP using pure IP versions and IP transition mechanisms over limited OSs (Windows Server 2008 & Linux Ubuntu 10.10). Finally the third part focused on video over IP using various IP transition mechanisms. First part contained the results for VoIP that were presented in bar graphs and tables. The comparison between platforms was identified and the metrics considered were RTT, jitter, throughput and packet loss. Second part covered the performance of VoIP on pure IP versions (4&6) and IP transition mechanisms and the results were compared. The outcome was presented in bar charts and tables. The third part for this chapter focused on video over IP using both versions of IP and IP transition mechanisms. The results were presented in bar and area graphs and compared their performance. It also covered performance comparison of various video protocols and differentiated the actual-throughput from impacted-throughput and arrived at the protocol with least impact for video over IP. Next chapter will cover the discussion of the results obtained and described in Chapter 5.
Chapter 6: Discussion & Findings

The previous chapter covered results analysis and this chapter describes discussions regarding the results obtained from the experiments and future work. Initially it describes the performance of VoIP over IP transition mechanisms using five different platforms. Secondly, it mentions performance comparison of VoIP over pure IP versions and IP transition mechanisms. Finally, it discusses the impact of IP transition and tunnelling mechanisms on video over IP quality and comparison among various video protocols. Performance metrics considered for this study were jitter, delay (RTT), throughput, packet loss, actual-throughput, impacted-throughput and CPU utilisation. The sections below discuss the results presented in Chapter 5 above in greater detail.

6.1. Performance of VoIP over IP Transition Mechanisms using 5 Platforms

The results for VoIP over IP transition mechanisms using different platforms were analysed in Chapter 5 above. The analysis for RTT (delay) shows that Windows 7 OS had better performance as compared to the other four platforms (OpenSuse, Ubuntu, Fedora and Windows Server 2008) using Dual-Stack transition mechanism. As observed previously (Chapter 5) five different VoIP CODECS were tested on these five operating systems using Dual-Stack transition mechanism. However, the result for Windows 7 OS was much better compared to other four platforms. Windows 7 OS had approximately 4.4 milliseconds lesser delay than other four platforms using G.711.1 codec. For G.711.2 codec it had approximately 3.2 milliseconds lesser delay while for G.723.1 codec it had approximately 1.3 milliseconds lesser delay using Dual-Stack. For G.729.2 it provided approximately 3.23 milliseconds lesser delay while for G.729.3 it produced approximately 1.4 milliseconds delay.

Overall, it was noticed that Windows 7 OS produced such efficient RTT results compared to other four platforms using five different CODECS on Dual-Stack transition
mechanism. Based on these results it can be concluded that Window 7 operating system is more compatible with Dual-Stack transition mechanism as compared to Windows Server 2008 and the three other Linux based operating systems (OpenSuse, Ubuntu and Fedora).

The results for jitter also show that Windows 7 OS outperformed other four platforms using Dual-Stack transition mechanism. All five CODECS were measured on five different platforms. However, Window 7 OS had consistent results on each codec and produced least amount of jitter. Furthermore, the outcome for packet loss was different as Windows Server 2008 OS had highest amount for packet loss using G.711.1, G.711.2 and G.729.3 CODECS on IPv6in4 tunnelling mechanism. It was noticed that Windows based operating systems had more packet loss than Linux based operating systems tested. The Windows 7 OS had second worst performance as it produced second highest amount of packet loss on all the CODECS tested expect for G.711.2 codec.

The comparison between IP transition mechanisms analysed in Chapter 5 above indicated that performance of Dual-Stack transition mechanism provided more delay and jitter. The results measured on Fedora OS for delay and jitter showed that Dual-Stack had the worst performance compared to IPv6to4 and IPv6in4 tunnelling mechanisms. This is due to both IP stacks working concurrently which caused more delay in the transmission. Another factor observed was that Dual-Stack created more jitter than the tunnelling mechanisms on most of the operating systems tested using five CODECS. This leads to another deduction that both tunnelling mechanisms operate by bridging IPv6 network via IPv4 cloud, whereas Dual-Stack operates by enabling IPv4 and IPv6 simultaneously.

The results obtained for five VoIP CODECS showed that G.711.1 had higher delay as compared to all the other four CODECS (G.711.2, G.723.1, G.729.2 & G.729.3) tested. However, G.723.1 produced least amount of delay. The outcome for jitter and packet loss also indicated that G.711.1 codec provided more packet loss and jitter than other CODECS on all five operating systems while G.723.1 codec provided minimal packet loss and better jitter performance than other four CODECS. There are few features
noticed about these results. Firstly, G.711.1 codec had a bigger packet size which caused higher delay, jitter and packet loss whereas G.723.1 codec which had smaller packet size among all CODECS tested, produced better results on these three IP transition mechanisms. Secondly it was noticed that these IP transition mechanisms do not perform well with bigger packet size.

6.2. Comparison of VoIP on IP versions and IP Transition Mechanisms

The performance of VoIP on both IP versions and three IP transition mechanisms was calculated using Windows and Linux operating systems and their results were analysed in Chapter 5 above. The results for pure IP versions and IP transition mechanisms were compared. This clarified the impact caused by (IPv4, IPv6, IPv6to4, IPv6in4 & Dual-Stack) on VoIP quality. The analysis of the results for delay (RTT) showed that Dual-Stack transition mechanism performed better on Windows OS using G.711.1 and G.711.2 CODECS. However, Dual-Stack performed better on Linux OS for G.723.1, G.729.2 and G.729.3 CODECS. In this experiment identical hardware and software were used for both tests, which were conducted using Windows and Linux operating systems but results obtained were different. The G.711.1 and G.711.2 CODECS have bigger packet size than G.723.1, G.729.2 and G.729.3 CODECS, which clarifies that Windows OS performed much better than Linux OS, when packets with bigger size are transmitted while Linux OS performed marginally better than Windows OS when packets with smaller size are transmitted.

Another factor observed was that IPv4 provided less delay than IPv6 using Windows OS for all five CODECS tested. However, using Linux OS IPv6 performed marginally better for G.711.2 and G.723.1 CODECS.

Comparison between IP transition mechanisms indicated that Dual-Stack transition mechanism produced more delay (RTT) than other two IP transition mechanisms using all five CODECS. This is due to the both IP stacks processing simultaneously and packets have to flow in two different IP stacks, which caused slight more delay than
single IP stack based packet flow. Moreover, IPv6to4 tunnelling mechanism showed that it performed better than IPv6in4 tunnelling mechanism on Windows OS using all five CODECS tested. The results analysis also show that IPv6in4 tunnelling mechanism performed slightly better than IPv6to4 tunnelling mechanism using Linux OS for G.711.1, G.723.1 and G.729.2 CODECS.

The outcome of measurement when analysed for jitter showed that IPv4 and Pv6 marginally performed close to each other on both operating systems (Windows & Linux) for all five CODECS tested. Performance among three IP transition mechanisms showed that Dual-Stack transition mechanism produced more jitter than IPv6to4 and IPv6in4 tunnelling mechanisms using all five CODECS tested.

The analysis for packet loss indicated that IPv4 using all five CODECS did not have any packet loss while IPv6 had produced small amount of packet loss for each codec tested. Another interesting feature noticed was that Dual-Stack transition mechanism provided least amount of packet loss compared to two tunnelling mechanisms (6to4 & 6in4). The possible reasons that allowed Dual-Stack to produce less packet loss is Dual-Stack operates in two separate IP stacks, in other words some packets flow using IPv4 and some packets flow using IPv6, which reduces the load on single “stack” and allows less packet loss in the transmission.

Comparison between two tunnelling mechanisms showed that IPv6to4 tunnel had better performance than IPv6in4 tunnel on Linux OS using all five CODECS from packet loss perspective. Overall, based upon all the test results observed, it can be concluded that IPv6to4 tunnelling mechanism was better than IPv6in4 tunnelling mechanism. As these two tunnelling mechanisms have different structure to encapsulate and de-capssulate the packets.
6.3. **Performance of Video over IP**

This section covers discussion over the performance of video over IP using IP transition mechanisms. It also discusses about wastage of bandwidth due to IP transition mechanisms and CPU processing power usage on different IP transition mechanisms.

### 6.3.1. Impact of IP versions & IP transition mechanisms on video protocols

The results evaluated and compared in Chapter 5 above showed that performance of all five video protocols (MPEG-1, MPEG-2, MPEG-4, MKV, & FLV) on both versions of IP and three IP transition mechanisms was different. Five video protocols had produced different throughput and bandwidth wastage due to the packet size of each protocol.

Moreover, packet size of the protocol is another factor, which was observed. It was noticed that protocols with larger packet size are impacted more and usage of those protocols will have major impact on bandwidth wastage. However, usage of protocol with smaller packet size had minimal impact on bandwidth wastage, which is more suitable to be used over IP tunnelling mechanisms. Furthermore, it was clarified that FLV had the best performance compared to the other four protocols on all the IP transition mechanisms tested as it had smaller packet size than other four protocol tested.

Performance of MPEG-4 protocol was much better than MPEG-1 and MPEG-2 protocols over IPv6to4 tunnelling mechanism. It was observed that MPEG-4 has better compression system in packets sequence [67 & 68], which is another factor that enabled it to perform better than other two protocols (MPEG-1 & MPEG-2). Moreover, it is due to the compression system used in the sequence of MPEG-4 structure that increased its performance on IPv6to4 tunnelling mechanism.
6.3.2. Actual-throughput versus impacted-throughput

The results analysed and compared between actual-throughput and impacted-throughput for Dual-Stack mechanism showed that pure IPv6 had slightly better performance than Dual-Stack using all five video protocols (MPEG-1, MPEG-2, MPEG-4, MKV & FLV). The Dual-Stack transition mechanism (DSTM) had slight impact on video protocols, as compared to pure IPv6, since in DSTM both IP versions operate simultaneously. Furthermore, protocols like FLV will have minimal impact on video streaming applications as it was noticed that FLV only wasted 0.4 kilobytes per second over Dual-Stack, which is reasonable for video streaming applications.

The comparison between pure IPv6 and IPv6to4 tunnelling mechanism indicated that IPv6to4 had significant additional impact on all five video protocols tested. IPv6to4 using MPEG-1 protocol caused approximately 106 kilobytes per second bandwidth wastage, while using MPEG-2 caused even more bandwidth wastage at approximately 242 kilobytes per second. IPv6to4 tunnelling mechanism requires encapsulation and de-capsulation process in order to transmit IPv6 packets through IPv4 network to another IPv6 network. The additional bandwidth wastage is caused by encapsulation and de-capsulation processes, which are essential and without them IPv6to4 will not perform. However, users would have to have extra bandwidth for video transmission when they use IPv6to4 tunnelling mechanism.

Performance analysis between pure IPv6 and IPv6in4 tunnelling mechanism showed that IPv6in4 had major impact on video protocols. The impact caused by IPv6in4 tunnel using MPEG-2 was approximately 246 kilobytes per second, which is even more than the impact caused by IPv6to4 tunnel. IPv6in4 tunnel operates in similar way as IPv6to4 tunnel but it has different structure than IPv6to4 tunnel [35, 36, 37, 38 & 39]. They both (6to4 & 6in4) are tunnelling mechanisms and perform encapsulation and de-capsulation process to allow IPv6 packets to flow through IPv4 cloud. However, IPv6in4 caused more impact on bandwidth wastage than IPv6to4 tunnel.
Overall, it was observed that Dual-Stack mechanism had minimal impact on bandwidth wastage compared to two tunnelling mechanisms (6to4 & 6in4). It was because Dual-Stack operates by enabling both versions of IP concurrently and caused less bandwidth wastage. However, both tunnelling mechanisms (6to4 & 6in4) perform encapsulation and de-capsulation process, which caused more bandwidth wastage due to encapsulation and de-capsulation process.

6.3.3. Impact of IP transition mechanisms on CPU processing power

The results for CPU power used by each IP transition mechanism indicated that Dual-Stack mechanism used more CPU processing power at approximately 30.3% using MPEG-1 protocol while IPv6to4 used 26.62% and IPv6in4 used 27.4%. This is due to both IP versions acting concurrently, which requires more CPU processing power. However, using IPv6to4 and IPv6in4 tunnelling mechanisms between IPv6 networks did not cause additional impact on CPU processing power. Hence, where CPU processing power is a consideration the use of these tunnelling mechanisms is preferable.

Highest CPU processing power was used during MPEG-4 transmission at approximately 33.44 over Dual-Stack mechanism. MPEG-4 also had higher impact on IPv6to4 at approximately 33.3% while the impact on IPv6in4 was 31.73%. This additional impact was caused by compression system, which is used for MPEG-4 protocol [67 & 68].

Lowest CPU processing power was used by FLV protocol at approximately 26%. FLV protocol used the same amount of CPU processing power on all three IP transition mechanisms.

Overall, it is clear that protocol like FLV requires lesser processing power than MPEG-1 protocol. Protocols with bigger packet size require more CPU processing power and protocol, which uses compression system, are also required higher CPU processing power.
6.4. Future Work

In this study three goals were achieved and presented. Main focus of this study was to identify the performance of voice and video over IP networks over IP transition and tunnelling mechanisms using different platforms. The first goal was to identify the performance of VoIP on three different IP transition mechanisms using five different platforms. Second goal was to identify the performance of five VoIP CODECS on both IP versions and compare their results against IP transition mechanisms results. Final goal was to clarify the impact caused by IP transition mechanisms on video quality. There is still a task for future research, which needs to be undertaken regarding NGN (Next Generation Network). Current study focused on some of the IP transition mechanisms; however, there are more IP transition mechanisms which can be studied. Few features for future studies are covered below:

- Include other IP transition mechanisms (NAT-PT, NAP-PT and Teredo)
- Include additional voice CODECS to compare with tested CODECS
- VoIP using SIP protocol over IP transition mechanisms is another research area
- Cover additional tests for video over IP using other video protocols (.ASF, .AVI, .WMV, .RM, .RAM, MPEG-5, MPEG-6 etc.) to identify the impact of IP transition mechanisms on these protocols
- Another area of study would be to measure packet loss of these video protocols over increased traffic loads.
Chapter 7: Conclusion

The purpose of this research study was to evaluate the quality of voice and video over IP transition mechanisms. This research study was based on experiments which involved five voice CODECS, five video protocols and five different operating systems. These experiments were divided into three different parts and each part of the experiment was aimed to cover different protocols for various metrics under study.

First part was conducted on voice over IP using three IP transition mechanisms on five various platforms. It involved five different voice CODECS (G.711.1, G.711.2, G.723.1, G.729.2 & G.729.3) and operating systems selected were Windows Server 2008, Windows 7, Linux Ubuntu 10.10, Fedora 14 and OpenSuse 11.3. The metrics covered in this part were delay, jitter, throughput and packet loss.

The aim of the second part was to identify the quality of VoIP on both IP versions (IPv4 & IPv6) under same environment as selected for first part of the experiment. However, in the second part of the experiment two operating systems were selected (Window Server 2008 & Linux Ubuntu 10.10). The results obtained from this part were compared with first part to clarify the difference.

The third part focused on the quality of video over Dual-Stack and two IP tunnelling mechanisms using five different video protocols such as MPEG-1, MPEG-2, MPEG-4, MKV and FLV. The metrics considered for this part were impacted-throughput, actual-throughput and CPU utilisation. The results observed clarified the impact caused by IP transition mechanisms on video protocols.

According to the related studies which were reviewed earlier in Chapter 2 indicated that this study has not been conducted prior to this research. It also indicated that similar work to part 1, and part 2 was studied before; however, part 3 (video over IP transition mechanisms) is completely new and has not been studied before. The following is the summary of the results which were obtained from the experiments.
• The performance of VoIP quality was much better on Windows 7 OS while Fedora 14 OS had second best performance as compared to other platforms (Windows Server 2008, Ubuntu 10.10 & OpenSUSE 11.3).
• Pure IPv4 and pure IPv6 performed better than Dual-Stack Transition Mechanism.
• IPv6-to-4 tunnelling mechanism performed better than IPv6-in-4 tunnelling mechanism.
• Quality of video was highly impacted by IPv6-in-4 tunnelling mechanism and wasted large amount of bandwidth.
• Performance of video protocols clarified that FLV was least impacted by IP transition mechanisms and provided the best performance comparing to other four protocols (MPEG-1, MPEG-2, MPEG-4 & MKV).

7.1. The Summary of Findings

Each of the hypotheses that were formulated helped this research study to arrive at the following conclusions.

**Hypothesis 1:** It is expected that quality of the voice will reduce as large amount of simultaneous VoIP calls are transmitted on pure IPv6 network as compared to pure IPv4.

It was observed from the results, that when 200 simultaneous calls were generated over pure IPv6 network, voice quality was degraded as compared to that over IPv4. The results clarified that performance of five VoIP CODECS tested over IPv4 produced zero percent packet loss, whereas all five CODECS tested on IPv6 produced 0.0007% to 0.0032%, which happens to be small packet loss for the range of VoIP CODECS tested.
Hypothesis 2: Voice CODECS with small packet size are expected to support more calls than the CODECS with large packet size over IP tunnelling mechanisms.

The results for packet loss support this hypothesis, since packet loss for voice CODECS with large packet size had greater packet loss. The results for delay also support this hypothesis, with greater delay for larger packet sized CODECS. However, the results for jitter were not consistent.

Hypothesis 3: The performance of VoIP over IP transition mechanisms are expected to be even lower than pure IPv6 network.

Overall performance of VoIP CODECS over IP transition mechanisms confirms that IP transition mechanisms had additional impact on VoIP quality as compared to pure IPv6 results. It was noticed that the two IP tunnelling mechanisms (6to4 & 6in4) impacted the VoIP quality due to their encapsulation and de-capsulation processes used for tunnelling. The results for Dual-Stack showed that it had lesser packet loss than the two tunnelling mechanisms but higher than pure IPv6.

Hypothesis 4: VoIP quality is expected to be impacted by the IP tunnelling mechanisms.

The results observed for video transmission over IP tunnelling mechanisms confirmed that irrespective of the video protocol used for transmission over the two IP tunnelling mechanisms, some sort of affect existed. In other words more bandwidth is required when VoIP over IP tunnelling mechanisms is transmitted because encapsulation and de-capsulation process in tunnelling mechanisms required more bandwidth when VoIP applications are broadcast over IP tunnelling mechanisms as compared to pure IPv4 and IPv6.

Hypothesis 5: Video protocols with smaller packet size are expected to perform better on IP transition mechanisms from quality of service point of view.

The analysis of the results clarified that video protocol known as FLV had smaller packet size compared to other 4 video protocols tested. It was noticed that FLV
was least impacted by IP transition mechanisms as it wasted lesser amount of bandwidth during transmission over IP transition mechanisms as compared to other 4 protocols tested (MPEG-1, MPEG-2, MPEG-4 & MKV). The bandwidth wastage noticed over IPv6to4 and IPv6in4 mechanisms ranged from 17 to 18 kilobytes per second using FLV protocol, whereas other protocols bandwidth wasted ranged from 70 to 240 kilobytes per second.
### Appendix A: Delay results for part 1 of data analysis

Table below shows the results for part 1: VoIP performance on three IP transition mechanisms using five platforms.

<table>
<thead>
<tr>
<th>Platform</th>
<th>IPv6to4</th>
<th>G.711.1</th>
<th>G.711.2</th>
<th>G.723.1</th>
<th>G.729.2</th>
<th>G.729.3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Federa 14</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPv6to4</td>
<td>0.00125995</td>
<td>0.00083145</td>
<td>0.000764175</td>
<td>0.000755775</td>
<td>0.000851225</td>
<td></td>
</tr>
<tr>
<td>IPv6in4</td>
<td>0.001118625</td>
<td>0.000855525</td>
<td>0.00076675</td>
<td>0.000762675</td>
<td>0.00084225</td>
<td></td>
</tr>
<tr>
<td>Dual-Stack</td>
<td>0.006452975</td>
<td>0.004369225</td>
<td>0.002040525</td>
<td>0.004065525</td>
<td>0.00211035</td>
<td></td>
</tr>
<tr>
<td><strong>Ubuntu 10.10</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPv6to4</td>
<td>0.005903975</td>
<td>0.00402595</td>
<td>0.002071725</td>
<td>0.0043016</td>
<td>0.002021925</td>
<td></td>
</tr>
<tr>
<td>IPv6in4</td>
<td>0.00568965</td>
<td>0.0045722</td>
<td>0.002007325</td>
<td>0.0039917</td>
<td>0.002100625</td>
<td></td>
</tr>
<tr>
<td>Dual-Stack</td>
<td>0.00738645</td>
<td>0.0047286</td>
<td>0.00200825</td>
<td>0.0042951</td>
<td>0.002195925</td>
<td></td>
</tr>
<tr>
<td><strong>OpenSUSE 11.3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPv6to4</td>
<td>0.0059992</td>
<td>0.004556025</td>
<td>0.001949425</td>
<td>0.004676225</td>
<td>0.00201425</td>
<td></td>
</tr>
<tr>
<td>IPv6in4</td>
<td>0.00591115</td>
<td>0.004283825</td>
<td>0.002104325</td>
<td>0.004301775</td>
<td>0.0021378</td>
<td></td>
</tr>
<tr>
<td>Dual-Stack</td>
<td>0.00593295</td>
<td>0.004568175</td>
<td>0.002101625</td>
<td>0.004488225</td>
<td>0.002202075</td>
<td></td>
</tr>
<tr>
<td><strong>Win Server 2008</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPv6to4</td>
<td>0.00668345</td>
<td>0.00426825</td>
<td>0.001959175</td>
<td>0.004221375</td>
<td>0.0019963</td>
<td></td>
</tr>
<tr>
<td>IPv6in4</td>
<td>0.007698725</td>
<td>0.0043265</td>
<td>0.0019488</td>
<td>0.0042454</td>
<td>0.00221385</td>
<td></td>
</tr>
<tr>
<td>Dual-Stack</td>
<td>0.00634605</td>
<td>0.004486375</td>
<td>0.002080875</td>
<td>0.0045628</td>
<td>0.002208025</td>
<td></td>
</tr>
<tr>
<td><strong>Windows 7</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPv6to4</td>
<td>0.00162455</td>
<td>0.00113315</td>
<td>0.000763225</td>
<td>0.000794525</td>
<td>0.000829425</td>
<td></td>
</tr>
<tr>
<td>IPv6in4</td>
<td>0.00159025</td>
<td>0.00117975</td>
<td>0.000753825</td>
<td>0.0008109</td>
<td>0.00081175</td>
<td></td>
</tr>
<tr>
<td>Dual-Stack</td>
<td>0.001512925</td>
<td>0.001118075</td>
<td>0.000695575</td>
<td>0.000775775</td>
<td>0.00072955</td>
<td></td>
</tr>
</tbody>
</table>
Appendix B: Jitter results for part 1 of data analysis

Table below shows the results for part 1: VoIP performance on three IP transition mechanisms using five platforms.

<table>
<thead>
<tr>
<th></th>
<th>IPv6to4</th>
<th>IPv6in4</th>
<th>Dual-Stack</th>
<th>IPv6to4</th>
<th>IPv6in4</th>
<th>Dual-Stack</th>
<th>IPv6to4</th>
<th>IPv6in4</th>
<th>Dual-Stack</th>
<th>IPv6to4</th>
<th>IPv6in4</th>
<th>Dual-Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Federa 14</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.711.1</td>
<td>0.0005392</td>
<td>0.00043485</td>
<td>0.000392525</td>
<td>0.0004505</td>
<td>0.000387275</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.711.2</td>
<td>0.0005277</td>
<td>0.0004483</td>
<td>0.000398975</td>
<td>0.000451375</td>
<td>0.000371575</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.723.1</td>
<td>0.002005125</td>
<td>0.00207615</td>
<td>0.001043925</td>
<td>0.001916625</td>
<td>0.000641625</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Ubuntu 10.10</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.711.1</td>
<td>0.00190825</td>
<td>0.0018915</td>
<td>0.0010585</td>
<td>0.0020711</td>
<td>0.00059685</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.711.2</td>
<td>0.0020231</td>
<td>0.0021869</td>
<td>0.00104255</td>
<td>0.00184715</td>
<td>0.000676375</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.723.1</td>
<td>0.0021629</td>
<td>0.002289625</td>
<td>0.001037325</td>
<td>0.0020399</td>
<td>0.000691375</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OpenSUSE 11.3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.711.1</td>
<td>0.001925425</td>
<td>0.002204425</td>
<td>0.000975775</td>
<td>0.002311925</td>
<td>0.00061765</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.711.2</td>
<td>0.00201195</td>
<td>0.002055425</td>
<td>0.00105715</td>
<td>0.002002525</td>
<td>0.0006616</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.723.1</td>
<td>0.0020462</td>
<td>0.00221895</td>
<td>0.00104925</td>
<td>0.002138125</td>
<td>0.000665225</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Win Server 2008</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.711.1</td>
<td>0.001832075</td>
<td>0.0019747</td>
<td>0.000967</td>
<td>0.0020178</td>
<td>0.000617825</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.711.2</td>
<td>0.002373125</td>
<td>0.0020666</td>
<td>0.00093735</td>
<td>0.00196775</td>
<td>0.000675125</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.723.1</td>
<td>0.00198325</td>
<td>0.0021428</td>
<td>0.001087325</td>
<td>0.00223445</td>
<td>0.000699425</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Windows 7</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.711.1</td>
<td>0.00056985</td>
<td>0.000489875</td>
<td>0.00040205</td>
<td>0.000431975</td>
<td>0.00037425</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.711.2</td>
<td>0.000558075</td>
<td>0.000530525</td>
<td>0.000404425</td>
<td>0.0004414</td>
<td>0.000361925</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.723.1</td>
<td>0.000599</td>
<td>0.000514225</td>
<td>0.000398925</td>
<td>0.000468325</td>
<td>0.000342075</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix C: Packet loss results for part 1 of data analysis

Table below shows the results for part 1: VoIP performance on three IP transition mechanisms using five platforms.

<table>
<thead>
<tr>
<th>Platform</th>
<th>G.711.1</th>
<th>G.711.2</th>
<th>G.723.1</th>
<th>G.729.2</th>
<th>G.729.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv6to4</td>
<td>0.00275</td>
<td>0.0025</td>
<td>0.001</td>
<td>0.0025</td>
<td>0.00225</td>
</tr>
<tr>
<td>IPv6in4</td>
<td>0.00275</td>
<td>0.00125</td>
<td>0.00175</td>
<td>0.0015</td>
<td>0.0025</td>
</tr>
<tr>
<td>Dual-Stack</td>
<td>0.00075</td>
<td>0.00075</td>
<td>0.00075</td>
<td>0.00175</td>
<td>0.00125</td>
</tr>
<tr>
<td>IPv6to4</td>
<td>0.003</td>
<td>0.00225</td>
<td>0.0015</td>
<td>0.002</td>
<td>0.00175</td>
</tr>
<tr>
<td>IPv6in4</td>
<td>0.00175</td>
<td>0.00125</td>
<td>0.00225</td>
<td>0.0025</td>
<td>0.0025</td>
</tr>
<tr>
<td>Dual-Stack</td>
<td>0.00175</td>
<td>0.001</td>
<td>0.0015</td>
<td>0.001</td>
<td>0.00075</td>
</tr>
<tr>
<td>IPv6to4</td>
<td>0.00225</td>
<td>0.0015</td>
<td>0.0015</td>
<td>0.00175</td>
<td>0.00275</td>
</tr>
<tr>
<td>IPv6in4</td>
<td>0.002</td>
<td>0.002</td>
<td>0.00225</td>
<td>0.00175</td>
<td>0.003</td>
</tr>
<tr>
<td>Dual-Stack</td>
<td>0.00125</td>
<td>0.00075</td>
<td>0.00175</td>
<td>0.00075</td>
<td>0.00225</td>
</tr>
<tr>
<td>IPv6to4</td>
<td>0.00225</td>
<td>0.003</td>
<td>0.00225</td>
<td>0.002</td>
<td>0.00275</td>
</tr>
<tr>
<td>IPv6in4</td>
<td>0.01</td>
<td>0.006</td>
<td>0.00125</td>
<td>0.00225</td>
<td>0.00425</td>
</tr>
<tr>
<td>Dual-Stack</td>
<td>0.001</td>
<td>0.0015</td>
<td>0.0001</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>IPv6to4</td>
<td>0.008</td>
<td>0.00128</td>
<td>0.00225</td>
<td>0.00125</td>
<td>0.0025</td>
</tr>
<tr>
<td>IPv6in4</td>
<td>0.0075</td>
<td>0.00125</td>
<td>0.00375</td>
<td>0.00225</td>
<td>0.00275</td>
</tr>
<tr>
<td>Dual-Stack</td>
<td>0.001</td>
<td>0.00075</td>
<td>0.00075</td>
<td>0.00075</td>
<td>0.00125</td>
</tr>
</tbody>
</table>
Appendix D: Delay results for part 2 of data analysis

Table below shows the results for part 2: VoIP performance on both IP versions & IP transition mechanisms.

<table>
<thead>
<tr>
<th></th>
<th>G.711.1</th>
<th>G.711.2</th>
<th>G.723.1</th>
<th>G.729.2</th>
<th>G.729.3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IPv4</strong></td>
<td>0.006059675</td>
<td>0.004225725</td>
<td>0.0018976</td>
<td>0.0043277</td>
<td>0.002105575</td>
</tr>
<tr>
<td><strong>IPv6</strong></td>
<td>0.00616395</td>
<td>0.004562425</td>
<td>0.001921475</td>
<td>0.0043137</td>
<td>0.00225385</td>
</tr>
<tr>
<td><strong>IPv6to4</strong></td>
<td>0.00668345</td>
<td>0.00426825</td>
<td>0.001959175</td>
<td>0.004221375</td>
<td>0.0019963</td>
</tr>
<tr>
<td><strong>IPv6in4</strong></td>
<td>0.007698725</td>
<td>0.0043265</td>
<td>0.0019488</td>
<td>0.0042454</td>
<td>0.00221385</td>
</tr>
<tr>
<td><strong>Dual-Stack</strong></td>
<td>0.00634605</td>
<td>0.004486375</td>
<td>0.002080875</td>
<td>0.0045628</td>
<td>0.002208025</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>G.711.1</th>
<th>G.711.2</th>
<th>G.723.1</th>
<th>G.729.2</th>
<th>G.729.3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IPv4</strong></td>
<td>0.007047075</td>
<td>0.004732025</td>
<td>0.002029175</td>
<td>0.0041796</td>
<td>0.00214145</td>
</tr>
<tr>
<td><strong>IPv6</strong></td>
<td>0.00741815</td>
<td>0.00465075</td>
<td>0.00196405</td>
<td>0.0044357</td>
<td>0.0022154</td>
</tr>
<tr>
<td><strong>IPv6to4</strong></td>
<td>0.005903975</td>
<td>0.00402595</td>
<td>0.002071725</td>
<td>0.0043016</td>
<td>0.002021925</td>
</tr>
<tr>
<td><strong>IPv6in4</strong></td>
<td>0.00568965</td>
<td>0.0045722</td>
<td>0.002007325</td>
<td>0.0039917</td>
<td>0.002100625</td>
</tr>
<tr>
<td><strong>Dual-Stack</strong></td>
<td>0.00738645</td>
<td>0.0047286</td>
<td>0.00200825</td>
<td>0.0042951</td>
<td>0.002195925</td>
</tr>
</tbody>
</table>
Appendix E: Jitter results for part 2 of data analysis

Table below shows the results for part 2: VoIP performance on both IP versions & IP transition mechanisms.

<table>
<thead>
<tr>
<th></th>
<th>Windows</th>
<th></th>
<th>Linux</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IPv4</td>
<td>IPv6</td>
<td>IPv4</td>
<td>IPv6</td>
</tr>
<tr>
<td>G.711.1</td>
<td>0.002005925</td>
<td>0.00203115</td>
<td>0.00205495</td>
<td>0.00205315</td>
</tr>
<tr>
<td>G.711.2</td>
<td>0.002033225</td>
<td>0.002154175</td>
<td>0.00231595</td>
<td>0.002241225</td>
</tr>
<tr>
<td>G.723.1</td>
<td>0.000967025</td>
<td>0.000940625</td>
<td>0.0009762</td>
<td>0.00093735</td>
</tr>
<tr>
<td>G.729.2</td>
<td>0.00206535</td>
<td>0.00204255</td>
<td>0.001937375</td>
<td>0.002157325</td>
</tr>
<tr>
<td>G.729.3</td>
<td>0.000700675</td>
<td>0.0006911</td>
<td>0.000682225</td>
<td>0.00070185</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>IPv6to4</th>
<th>IPv6in4</th>
<th>IPv6to4</th>
<th>IPv6in4</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4</td>
<td>0.001832075</td>
<td>0.00198325</td>
<td>0.00205495</td>
<td>0.00205315</td>
</tr>
<tr>
<td>IPv6</td>
<td>0.0019747</td>
<td>0.0021428</td>
<td>0.00231595</td>
<td>0.002241225</td>
</tr>
<tr>
<td>IPv6to4</td>
<td>0.000967</td>
<td>0.001087325</td>
<td>0.0009762</td>
<td>0.00093735</td>
</tr>
<tr>
<td>IPv6in4</td>
<td>0.0020178</td>
<td>0.00223445</td>
<td>0.001937375</td>
<td>0.002157325</td>
</tr>
<tr>
<td>Dual-Stack</td>
<td>0.000617825</td>
<td>0.000699425</td>
<td>0.000682225</td>
<td>0.00070185</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>IPv6to4</th>
<th>IPv6in4</th>
<th>IPv6to4</th>
<th>IPv6in4</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4</td>
<td>0.00190825</td>
<td>0.00198325</td>
<td>0.00205495</td>
<td>0.00205315</td>
</tr>
<tr>
<td>IPv6</td>
<td>0.0018915</td>
<td>0.0021428</td>
<td>0.00231595</td>
<td>0.002241225</td>
</tr>
<tr>
<td>IPv6to4</td>
<td>0.0010585</td>
<td>0.001087325</td>
<td>0.0009762</td>
<td>0.00093735</td>
</tr>
<tr>
<td>IPv6in4</td>
<td>0.0021869</td>
<td>0.0021428</td>
<td>0.001937375</td>
<td>0.002157325</td>
</tr>
<tr>
<td>Dual-Stack</td>
<td>0.00059685</td>
<td>0.000699425</td>
<td>0.000682225</td>
<td>0.00070185</td>
</tr>
</tbody>
</table>
Appendix F: Packet loss results for part 2 of data analysis

Table below shows the results for part 2: VoIP performance on both IP versions & IP transition mechanisms.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>G.711.1</th>
<th>G.711.2</th>
<th>G.723.1</th>
<th>G.729.2</th>
<th>G.729.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>IPv4</td>
<td>0</td>
<td>0.00825</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>IPv6</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.00175</td>
<td>0.00025</td>
<td>0.00225</td>
</tr>
<tr>
<td></td>
<td>IPv6to4</td>
<td>0.00225</td>
<td>0.003</td>
<td>0.00225</td>
<td>0.002</td>
<td>0.00275</td>
</tr>
<tr>
<td></td>
<td>IPv6in4</td>
<td>0.004</td>
<td>0.02</td>
<td>0.00125</td>
<td>0.00225</td>
<td>0.00425</td>
</tr>
<tr>
<td></td>
<td>Dual-Stack</td>
<td>0.001</td>
<td>0.0015</td>
<td>0</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>Linux</td>
<td>IPv4</td>
<td>0.00025</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>IPv6</td>
<td>0.00075</td>
<td>0.001</td>
<td>0.00125</td>
<td>0.00025</td>
<td>0.00325</td>
</tr>
<tr>
<td></td>
<td>IPv6to4</td>
<td>0.003</td>
<td>0.00225</td>
<td>0.0015</td>
<td>0.002</td>
<td>0.00175</td>
</tr>
<tr>
<td></td>
<td>IPv6in4</td>
<td>0.00175</td>
<td>0.00125</td>
<td>0.00225</td>
<td>0.0025</td>
<td>0.0025</td>
</tr>
<tr>
<td></td>
<td>Dual-Stack</td>
<td>0.00175</td>
<td>0.001</td>
<td>0.0015</td>
<td>0.001</td>
<td>0.00075</td>
</tr>
</tbody>
</table>
Appendix G: Windows 7 OS routing enabled

Diagram below illustrates the commands which were used to enable routing on Windows 7 operating system.

I enabled IP Routing by updating the registry key below on Vista Desktop #1
HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet\Services\Tcipc \Parameters
subkey : "IPEnableRouter" entry to 1
and rebooted

C:\Users\Benoy George>ipconfig /all

Windows IP Configuration

Host Name .............. : VistaDesk1
Primary Dns Suffix ......... :
Node Type .............. : Hybrid
IP Routing Enabled....... : Yes
WINS Proxy Enabled....... : No

Appendix H: VoIP CODECS specification

Table below describes the features of various voice CODECS.

<table>
<thead>
<tr>
<th>CODECS</th>
<th>Samples</th>
<th>Frame size</th>
<th>Packets (Per Second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.711.1</td>
<td>1</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>G.723.1</td>
<td>1</td>
<td>30</td>
<td>26</td>
</tr>
<tr>
<td>G.729.2</td>
<td>2</td>
<td>10</td>
<td>50</td>
</tr>
</tbody>
</table>
References


