Performance Analysis of IPv4 vs. IPv6 on various Operating Systems using Jumbo Frames

Name: Paula Raymond Lutui

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

UNITEC New Zealand, 2011
Abstract

The services offered on the Internet, the requirements and demands for use of multimedia applications, such as sharing video files, photos, video conferencing, and the increased usage of social networking sites, have all contributed to the exponential growth of the Internet usage over the last ten years. In order to meet these demands, hardware developers have increased the speed of hardware such as processors and switches, and routers, also have increased capacity of the communication medium. However, the maximum data unit that can be passed onwards by any layer via this communication medium remains untouched. The current Maximum Transfer Unit (MTU) is 1500 Bytes, to address this issue, a new MTU has been emerged which is known as Jumbo frames. The proposed MTU for Jumbo frames in now 9000 Bytes.

The purpose of this study is to evaluate the performance of Jumbo frames on a network environment employing six operating systems from two different distributions. These operating systems are; Microsoft Windows Server 2008, Microsoft Windows Server 2003 and Microsoft Windows 7 Professional and from the Linux distributions, Linux Fedora, Ubuntu and OpenSUSE. In this study, two transmission protocols were employed namely, Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). Two Internet protocols were also engaged in these performance experiments, Internet Protocol version six (IPv6) and Internet Protocol version four (IPv4). There were five main performance metrics extract from the data collected in this experimental study namely the throughput, delay, jitter, the CPU utilisations on the software routers and the packets dropped rate. The experiments were conducted on the network level and at the application level with the following applications; DNS, Games and VoIP traffics on five different VoIP CODECs. The Jumbo frame sizes involved ranging from 1518 Bytes to 9014 Bytes.

The findings of this study concluded that for traffic employing TCP as transport protocol, Microsoft Windows Server 2008 and Microsoft Windows 7 yielded the highest throughput on both IPv6 and IPv4 and also Linux OpenSUSE on IPv4 only. When UDP was employed as transmission protocol, all of the operating systems yielded similar throughput values.

With regards to the applications’ results, the findings of this study concluded that for DNS, Microsoft Windows Server 2008 and Microsoft Windows Server 2003 yielded the highest throughput on IPv6 and Linux Fedora on IPv4. For the games on IPv6, Linux Ubuntu yielded the highest throughput on the CSa game while all of the operating systems yielded similar values on the CSI game. On the Quake3 game, Microsoft Windows Server 2008 produced the highest throughput values.
With regards to the VoIP on IPv6, Microsoft Windows Server 2003 and Linux Ubuntu yielded the highest throughput on the G.711.1 CODEC while OpenSUSE yielded the highest values on the G.711.2 CODEC hence; Microsoft Windows Server 2008 generated the highest amount of delay on all of the CODECs. With regards to VoIP on IPv4, Microsoft Windows 7 yielded the highest throughput on the G.711.1 CODEC while Microsoft Windows 7, Linux Fedora and OpenSUSE yielded the highest values on the G.711.2 CODEC however, on both IPv6 and IPv4 for the G.723.1, G.729.2 and the G.729.3 CODECs, all of the operating systems yielded similar throughput values. The good combination for IPv6 on VoIP would be Microsoft Windows Server 2003 or Linux Ubuntu on the G.711.1 CODEC, they both yielded high throughput but low delay and low jitter and for IPv4, Microsoft Windows 7 on the G.711.1 or G.711.2 CODECs; it yielded high throughput but low delay and low jitter.
Acknowledgement

It is such an amazing feeling to complete this study for my thesis however; I have a sense of duty to acknowledge the guidance and support I received from all the experts who assisted me in the compilation and successful completion of this paper.

Firstly, I would like to give thanks to the King of kings and the Lord of lords for giving me the most precious gift of all, which is the gift of life and for allowing me to effectively complete this work.

Secondly, I would like to thank my principal supervisor Shaneel Narayan who has been with me from the very beginning right through to the completion of this study. He was always there when I needed him; he listened and provided advice and ideas on how to tackle the technical problems I encountered. I would also like to thank my associate supervisor Aaron Chen for his assistance during the compilation of this document.

Thirdly, I would also like to acknowledge the privilege I had during this research to work together with Professor Dr. Kay Fielden. Her guidance and expert advice during the compilation of this document was priceless.

Last but not least, I want to acknowledge the help and support I receive from my family. They have been my silent partner during this work. The prayers, the encouragement and the support not only during this work but throughout my Master of Computing study, were the motivating force I needed during this journey and to them I am forever grateful.
# Table of Contents

Abstract .................................................................................................................. ii
Acknowledgement ................................................................................................. iv
List of Tables .......................................................................................................... ix
List of Figures ......................................................................................................... x
List of Abbreviations ............................................................................................. xiii

Publications ........................................................................................................... 1

Chapter 1: Introduction ......................................................................................... 2
  1.1 Structure of the Report ................................................................................ 4

Chapter 2: Literature Review .............................................................................. 5
  2.1 Literature Background ................................................................................ 5
    2.1.1 Networking and Network Performance. ............................................. 5
    2.1.2 Performance Analysis ...................................................................... 7
  2.2 Internet Protocol .......................................................................................... 11
  2.3 The Transmission Protocols ....................................................................... 17
  2.4 The Network Performance Measurement Tools ....................................... 19
    2.4.1 IP Traffic ......................................................................................... 20
    2.4.2 Iperf .............................................................................................. 21
    2.4.3 Netperf .......................................................................................... 22
    2.4.4 Multi-Generator (MGEN) ................................................................. 22
    2.4.5 Distributed Internet Traffic Generator (D-ITG) .............................. 23
  2.5 Performance Measurement Tool for this Study ........................................... 23
  2.6 Jumbo Frames ............................................................................................. 25
  2.7 Literature Analysis ....................................................................................... 28
  2.9 Chapter Summary ......................................................................................... 29

3.0 Methodology .................................................................................................... 30
  3.1 Hypotheses .................................................................................................. 30
  3.2 Research Methods ....................................................................................... 31
  3.3 Methodology Employed for this Study ....................................................... 34
  3.4 Data Collection Method ............................................................................ 34
  3.5 Chapter Summary ......................................................................................... 37
4.0 Experimental Setup ............................................................................................................. 38
4.1 Hardware Specifications ...................................................................................................... 38
4.2 Software Specifications ...................................................................................................... 39
4.3 Network Design ................................................................................................................ 39
4.4 Determining the Optimal Value ......................................................................................... 40
4.5 Optimal Values for this Study ........................................................................................... 45
4.6 Chapter Summary .............................................................................................................. 46

Chapter 5: Data Analysis ............................................................................................................. 47
5.1 TCP Analysis ...................................................................................................................... 47
   5.1.1 TCP Throughputs on IPv6 & IPv4 Jumbo Frames ......................................................... 47
   5.1.2 Compare TCP Throughputs on IPv4 & IPv6 Jumbo Frames ....................................... 49
   5.1.3 TCP One Way Delay on IPv6& IPv4 Jumbo Frame .................................................... 50
   5.1.4 Compare TCP Delays on IPv4 & IPv6 Jumbo Frames .................................................. 52
   5.1.5 TCP Jitter on IPv6 & IPv4 Jumbo Frame .................................................................... 53
   5.1.6 Compare TCP Jitter on IPv4 & IPv6 Jumbo Frames .................................................... 54
   5.1.7 TCP CPU Utilisations for Router 1 on IPv6 & IPv4 Jumbo Frame ............................. 55
   5.1.8 Comparing TCP CPU Utilisations for Router 1 on IPv4 & IPv6 Jumbo Frames ......... 57
   5.1.9 TCP CPU Utilisations for Router 2 on IPv6 & IPv4 Jumbo Frame ................................ 58
   5.1.10 Comparing TCP CPU Utilisations for Router 2 on IPv4 & IPv6 Jumbo Frames .......... 60

5.2 UDP Analysis ..................................................................................................................... 61
   5.2.1 UDP Throughput on IPv6 & IPv4 Jumbo Frames ......................................................... 61
   5.2.2 Compare UDP Throughputs on IPv4 & IPv6 Jumbo Frames ....................................... 63
   5.2.3 UDP One Way Delay on IPv6 & IPv4 Jumbo Frame .................................................... 64
   5.2.4 Compare UDP One Way Delay on IPv4 & IPv6 Jumbo Frames .................................... 66
   5.2.5 UDP Jitter Results on IPv6 & IPv4 Jumbo Frame ........................................................ 68
   5.2.6 Compare UDP Jitter Results on IPv4 & IPv6 Jumbo Frames ......................................... 70
   5.2.7 UDP Packets Dropped on IPv6 & IPv4 Jumbo Frame ................................................... 71
   5.2.8 Compare Packets Dropped Results on IPv4 & IPv6 Jumbo Frames .............................. 73
   5.2.9 UDP CPU Utilisations on IPv6 & IPv4 Jumbo Frame on Router 1 ............................... 75
   5.2.10 Compare UDP CPU Usage for Router 1 on IPv4 & IPv6 Jumbo Frames ....................... 77
   5.2.11 UDP CPU Utilisations on IPv6 & IPv4 Jumbo Frame on Router 2 ............................... 79
   5.2.12 Compare UDP CPU Usage for Router 2 on IPv4 & IPv6 Jumbo Frames ...................... 81
5.2.13 Compare TCP & UDP Throughput on IPv6 Jumbo Frames ............................................. 82
5.2.14 Compare TCP & UDP Throughput on IPv4 Jumbo Frames ............................................. 85
5.2.15 Compare Jumbo Frames IPv6 & IPv4 Throughput on TCP ............................................. 87

5.3 Applications Analysis ............................................................................................................. 90
5.3.1 DNS Throughputs on IPv6 & IPv4 Jumbo Frames ......................................................... 90
5.3.2 Compare DNS IPv6 & IPv4 Throughput results on Jumbo Frames .......................... 91
5.3.3 DNS Delay in IPv6 & IPv4 Jumbo Frames .................................................................. 92
5.3.4 Compare DNS IPv6 & IPv4 Delay results on Jumbo Frames ....................................... 93
5.3.5 DNS Jitter on IPv6 & IPv4 Jumbo Frames ....................................................................... 94
5.3.6 Compare DNS IPv6 & IPv4 Jitter results on Jumbo Frames ........................................ 95
5.3.7 DNS CPU Usage on Router 1 on IPv6 & IPv4 Jumbo Frames .................................... 96
5.3.8 Compare DNS IPv6 & IPv4 Usage results on Router 1 on Jumbo Frames .................. 97
5.3.9 DNS CPU Usage on Router 2 on IPv6 & IPv4 Jumbo Frames .................................... 98
5.3.10 Compare DNS IPv6 & IPv4 CPU Usages on Router 2 on Jumbo Frames ................. 99
5.3.11 Games Throughput Results on IPv6 & IPv4 Jumbo Frames ...................................... 100
5.3.12 Compare Games Throughput Results on IPv6 & IPv4 Jumbo Frames ...................... 101
5.3.13 Games Delay Results on IPv6 & IPv4 Jumbo Frames ................................................ 103
5.3.14 Compare Games Delay Results on IPv6 & IPv4 Jumbo Frames ................................. 105
5.3.15 Games Jitter Results on IPv6 & IPv4 Jumbo Frames .................................................. 106
5.3.16 Compare Games Jitter Results on IPv6 & IPv4 Jumbo Frames .................................. 108
5.3.17 Games CPU utilisations Results on Router 1 IPv6 & IPv4 Jumbo Frames ............... 110
5.3.18 Compare Games CPU Usage Results on IPv6 & IPv4 Jumbo Frames on Router 1 .. 112
5.3.19 Games CPU utilisations Results on Router 2 IPv6 & IPv4 Jumbo Frames ............... 115
5.3.20 Compare Games CPU Usage Results on IPv6 & IPv4 Jumbo Frames on Router 2 .. 117
5.3.21 Games Packets Dropped Results for IPv6 & IPv4 Jumbo Frames .............................. 119
5.3.22 Compare Games Packets Dropped Results on IPv6 & IPv4 Jumbo Frames ............. 121
5.3.23 VoIP Throughput Results for IPv6 & IPv4 Jumbo Frames ........................................ 123
5.3.24 Compare VoIP Throughput Results on IPv6 & IPv4 Jumbo Frames ......................... 126
5.3.25 VoIP Delay Results for IPv6 & IPv4 Jumbo Frames ................................................ 128
5.3.26 Compare VoIP Delay Results on IPv6 & IPv4 Jumbo Frames ..................................... 130
5.3.27 VoIP Jitter Results for IPv6 & IPv4 Jumbo Frames .................................................... 132
5.3.27 Compare VoIP Jitter Results on IPv6 & IPv4 Jumbo Frames ..................................... 134
5.3.29 VoIP CPU Usage Results for IPv6 & IPv4 Jumbo Frames on Router 1 .................. 136
5.3.30 Compare VoIP IPv6 & IPv4 Jumbo Frames CPU Usage Results on Router1 ......... 138
5.3.31 VoIP CPU Usage Results for IPv6 & IPv4 Jumbo Frames on Router 2 ................. 139
5.3.32 Compare VoIP IPv6 & IPv4 Jumbo Frames CPU Usage Results on Router2 .......... 142
5.3.33 VoIP Packets Dropped Results for IPv6 & IPv4 Jumbo Frames on Router2 ........... 143
5.3.34 Compare VoIP Packets Dropped Results on IPv6 & IPv4 Jumbo Frames ................ 145
5.4 Summary ............................................................................................................... 147
Chapter 6: Discussion ..................................................................................................... 148
6.1 TCP Performance on Jumbo Frames with IPv6 and IPv4 ...................................... 148
6.2 UDP Performance on Jumbo Frames IPv6 and IPv4 .............................................. 154
6.3 Comparison of TCP and UDP Performance on Jumbo Frames IPv6 and IPv4 ........ 160
6.4 Applications Performance on Jumbo Frames IPv6 and IPv4 ................................. 162
6.4.1 DNS Performance on Jumbo Frames IPv6 and IPv4 ........................................ 162
6.4.2 Games Performance on Jumbo Frames IPv6 and IPv4 .................................... 163
6.4.3 VoIP Performance on Jumbo Frames IPv6 and IPv4 ...................................... 165
6.5 Summary of Discussions ....................................................................................... 167
6.5.1 Summary of Applications’ Discussions ............................................................ 170
7.0 Conclusions ............................................................................................................ 174
7.1 Further Study ......................................................................................................... 180
References: .................................................................................................................... 181
List of Tables

Table 1.1: World Internet Users and Population Statistics .................................................................2
Table 2.1: Throughput measures and units used. .................................................................................8
Table 2.2: IPv4 three main classes .......................................................................................................11
Table 2-3: IPv4 header field and its corresponding IPv6 header field ..................................................15
Table 2-4: Comparing the main differences of IPv4 and IPv6..............................................................16
Table 2-5: Comparing TCP and UDP .................................................................................................18
Table 2-6: Computer specifications and operating systems employed. ...............................................27
Table 3-1: Features of Qualitative & Quantitative Research .................................................................33
Table 4-1: Hardware specifications ......................................................................................................38
Table 4-2: Optimal value combinations employed for this study .......................................................45
List of Figures

Figure 1: Diagram of network environment for this study. ................................. 1
Figure 2: Type of Research, Data Collection & Techniques. .................................. 2
Figure 3: The TCP/IP model. ........................................................................... 4
Figure 4: TCP throughput on Jumbo Frames on various operating systems. .......... 4
Figure 5: TCP IPv4 throughput on Jumbo frames on various operating systems. .... 5
Figure 6: TCP IPv6 throughput on Jumbo frames on various operating systems. .... 6
Figure 7: IPv6 Jitter on Jumbo frames on various operating systems. .................. 7
Figure 8: IPv4 Jitter on Jumbo frames on various operating systems. .................. 8
Figure 9: Differences in throughput between packet sizes 12896-16120 Bytes. ...... 9
Figure 10: Differences in throughput between packet sizes 9672-12896 Bytes. ...... 10
Figure 11: Throughputs on 90000 packets per seconds. ................................. 11
Figure 12: Throughputs on 180000 packets per seconds................................. 12
Figure 13: Throughputs on 270000 packets per seconds.................................. 13
Figure 14: UDP Throughputs on 3224 Bytes Packet size................................ 14
Figure 15: UDP Throughputs on 6448 Bytes Packet size................................. 15
Figure 16: UDP Throughputs on 9672 Bytes Packet size................................. 16
Figure 17: UDP Throughputs on 12896 Bytes Packet size................................. 17
Figure 18: UDP Throughputs on 16120 Bytes Packet size................................. 18
Figure 19: TCP IPv4 delay on Jumbo frames on various operating systems. ....... 19
Figure 20: TCP IPv6 delay on Jumbo frames on various operating systems. ....... 20
Figure 21: Comparison of IPv4 & IPv6 delay results on Jumbo frames............... 21
Figure 22: Comparison of IPv4 & IPv6 Jitter results on Jumbo frames................ 22
Figure 23: TCP IPv4 CPU utilisations on Router 1 for all operating systems....... 23
Figure 24: TCP IPv6 CPU utilisations on Router 1 for all operating systems....... 24
Figure 25: Comparison of TCP IPv6 & IPv4 CPU utilisations on Router 1 Jumbo frames... 25
Figure 26: TCP IPv6 CPU utilisations on Router 2 for all operating systems....... 26
Figure 27: TCP IPv4 CPU utilisations on Router 2 for all of the operating systems. 27
Figure 28: Comparison of TCP IPv6 & IPv4 Router 2 CPU utilisations on Jumbo frames. 28
Figure 29: UDP IPv6 throughput on Jumbo frames on various operating systems. 29
Figure 30: UDP IPv4 throughput on Jumbo frames on various operating systems. 30
Figure 31: Comparison of the Jumbo frames UDP Throughput results on IPv6 & IPv4. 31
Figure 32: UDP IPv6 delay on Jumbo frames on various operating systems. ....... 32
Figure 33: UDP IPv4 delay on Jumbo frames on various operating systems. ....... 33

X
Figure 5-3-36: VoIP throughput results on Jumbo frames IPv4 ................................................................. 124
Figure 5-3-37: Compare VoIP throughput on Jumbo frames IPv6 & IPv4 ....................................................... 126
Figure 5-3-38: VoIP delay results on IPv6 Jumbo frames ............................................................................. 128
Figure 5-3-39: VoIP delay results on Jumbo frames IPv4 ............................................................................. 129
Figure 5-3-40: Compare VoIP delay results on Jumbo frames IPv6 & IPv4 ................................................... 130
Figure 5-3-41: VoIP jitter results on Jumbo frames IPv6 ............................................................................... 132
Figure 5-3-42: VoIP jitter results on Jumbo frames IPv4 ............................................................................... 133
Figure 5-3-43: Compare VoIP Jitter results on Jumbo frames IPv6 & IPv4 .................................................... 134
Figure 5-3-44: VoIP IPv6 Jumbo frames CPU utilisations on Router 1 ............................................................ 136
Figure 5-3-45: VoIP IPv6 Jumbo frames CPU utilisations on Router 1 ............................................................ 137
Figure 5-3-46: Compare VoIP Jumbo frames IPv6 & IPv4 CPU usage on Router 1 .......................................... 138
Figure 5-3-47: VoIP IPv6 Jumbo frames CPU usage on Router 2 ................................................................ 140
Figure 5-3-48: VoIP Jumbo frames CPU usage on Router 2 IPv4 ............................................................... 141
Figure 5-3-49: Compare VoIP Jumbo frames IPv6 & IPv4 CPU utilisations on Router 2 ............................ 142
Figure 5-3-50: VoIP packets dropped on Jumbo frames IPv6 ...................................................................... 143
Figure 5-3-51: VoIP packets dropped on Jumbo frames IPv4 ...................................................................... 144
Figure 5-3-52: Compare VoIP packets dropped on Jumbo frames IPv6 & IPv4 .......................................... 146
Figure 6-1: IPv6 - TCP extra experiments on Microsoft Windows Server 2008 ............................................. 149
Figure 6-2: TCP throughput results on frame size 4095 .............................................................................. 150
Figure 6-3: TCP throughput results on frame size 4096 ............................................................................. 150
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARPANET</td>
<td>Advanced Research Project Agency Network</td>
</tr>
<tr>
<td>ATM</td>
<td>Asynchronous Transfer Mode</td>
</tr>
<tr>
<td>CAN</td>
<td>Campus Area Network</td>
</tr>
<tr>
<td>CIDR</td>
<td>Classless Inter-Domain Routing</td>
</tr>
<tr>
<td>CODEC</td>
<td><strong>Com</strong>ressor – <strong>Dec</strong>ompressor or <strong>Co</strong>der - <strong>De</strong>coder</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CTS</td>
<td>Clear to send</td>
</tr>
<tr>
<td>DARPA</td>
<td>Defence Advanced Research Project Agency</td>
</tr>
<tr>
<td>D-ITG</td>
<td>Distributed Internet Traffic Generator</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>FRJ</td>
<td>Fast Resilient Jumbo frame</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronic Engineer</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPSec</td>
<td>Internet Protocol Security</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>ISO</td>
<td>International Organization for Standardisation</td>
</tr>
<tr>
<td>ITD</td>
<td>Inter Departure Time</td>
</tr>
<tr>
<td>Kbps</td>
<td>Kilobits per second</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>MAN</td>
<td>Metropolitan Area Network</td>
</tr>
<tr>
<td>Mbps</td>
<td>Megabits per second</td>
</tr>
<tr>
<td>MGEN</td>
<td>Multi-Generator</td>
</tr>
<tr>
<td>ms</td>
<td>Milliseconds</td>
</tr>
<tr>
<td>MTU</td>
<td>Maximum Transmission Unit</td>
</tr>
<tr>
<td>MGEN</td>
<td>Multi-Generator</td>
</tr>
<tr>
<td>NAT</td>
<td>Network Address Translation</td>
</tr>
<tr>
<td>NGIP</td>
<td>Next Generation Internet Protocol</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection</td>
</tr>
<tr>
<td>PAN</td>
<td>Personal Area Network</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Explanation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistance</td>
</tr>
<tr>
<td>QPR</td>
<td>Quantitative Positivist Research</td>
</tr>
<tr>
<td>RMA</td>
<td>Reliability, Maintainability and Availability</td>
</tr>
<tr>
<td>RTS</td>
<td>Request to send</td>
</tr>
<tr>
<td>SCTP</td>
<td>Stream Control Transmission Protocol</td>
</tr>
<tr>
<td>SONET</td>
<td>Synchronous Optical Networking</td>
</tr>
<tr>
<td>SPEC</td>
<td>Standard Performance Evaluation Corporation</td>
</tr>
<tr>
<td>SR</td>
<td>Sample Rate</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>VoIP</td>
<td>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>
</tbody>
</table>
Publications

Two publications resulted out of this work. They have been presented as full papers at Institute of Electrical and Electronics Engineers (IEEE) conferences and can be accessed through IEEE Xplore. Their citations are:


Additional Publication

Two publications resulted out of a research project conducted for my Bachelor of Computing Systems degree. This was the beginning of the work that I continue for my Master of Computing thesis. They have been presented as full papers at Institute of Electrical and Electronics Engineers (IEEE) conferences and can be accessed through IEEE Xplore. Their citations are:


Chapter 1: Introduction

According to Table 1.1 presented below, Internet usage has increased by 480.4% during the period 2000-2011. The table below was taken from the World internet usage and population statistics website, illustrating the population of internet users and the growth from major world regions.

WORLD INTERNET USAGE AND POPULATION STATISTICS

March 31, 2011

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Africa</td>
<td>1,037,524,058</td>
<td>4,514,400</td>
<td>118,609,620</td>
<td>11.4 %</td>
<td>2,527.4 %</td>
<td>5.7 %</td>
</tr>
<tr>
<td>Asia</td>
<td>3,879,740,877</td>
<td>114,304,000</td>
<td>922,329,554</td>
<td>23.8 %</td>
<td>706.9 %</td>
<td>44.0 %</td>
</tr>
<tr>
<td>Europe</td>
<td>816,426,346</td>
<td>105,096,093</td>
<td>476,213,935</td>
<td>58.3 %</td>
<td>353.1 %</td>
<td>22.7 %</td>
</tr>
<tr>
<td>Middle East</td>
<td>216,258,843</td>
<td>3,284,800</td>
<td>68,553,666</td>
<td>31.7 %</td>
<td>1,987.0 %</td>
<td>3.3 %</td>
</tr>
<tr>
<td>North America</td>
<td>347,394,870</td>
<td>108,096,800</td>
<td>272,066,000</td>
<td>78.3 %</td>
<td>151.7 %</td>
<td>13.0 %</td>
</tr>
<tr>
<td>Latin America/Caribbean</td>
<td>597,283,165</td>
<td>18,068,919</td>
<td>215,939,400</td>
<td>36.2 %</td>
<td>1,037.4 %</td>
<td>10.3 %</td>
</tr>
<tr>
<td>Oceania / Australia</td>
<td>35,426,995</td>
<td>7,620,480</td>
<td>21,293,830</td>
<td>60.1 %</td>
<td>179.4 %</td>
<td>1.0 %</td>
</tr>
<tr>
<td>WORLD TOTAL</td>
<td>6,930,055,154</td>
<td>360,985,492</td>
<td>2,095,006,005</td>
<td>30.2 %</td>
<td>480.4 %</td>
<td>100.0 %</td>
</tr>
</tbody>
</table>

Table 1.1: World Internet Users and Population Statistics

The increased use of social networking sites on the Internet, and the growing services of the internet, including the demands and requirements for the use of multimedia applications, needs higher communication speed. To address this issue, hardware developers have increased the speed of hardware such as processors, switches and routers. Developers have also increased the speed of infrastructure backbones such as the capacity of the cables used, however the maximum amount of data that can be transferred via these media remain untouched and unchanged. Hsiao, Chen, Huang, Chu and Yeh (2009) stated that, “the CPU workload is heavy and the processing of network protocol task is the bottleneck”. There is an issue with the existing Internet Protocol version, IPv4, which is running out of IP addresses; this issue has been addressed by the development and the introduction of the Next Generation Internet Protocol (NGIP) also known as Internet Protocol version 6. In the computer networking arena, the maximum number of data that can be transferred over the network can be defined, this is known as MTU. This can be done via the network interface
card (NIC). However, the MTU that can be transferred from source to destination remains unchanged.

The current MTU is 1500 Bytes. A new MTU has been introduced that aims to resolve this issue and is known as “Jumbo Frames”. Jumbo Frames has an MTU of 9000 Bytes. According to Hewlett-Packard (2000), “the conventional Ethernet frame with its MTU of 1500 bytes was developed within the limitations of mid-twentieth century computers.” Networking hardware technologies such as servers, Ethernet cables, switches and routers have been improved with regards to their speed and capacity, however the maximum number of data units that can be passed onwards via the communication medium such as the network interface card remains the same. Hewlett-Packard (2000) rightly explained that “the net result is that a substantial portion of the speed offered by today’s servers is disproportionately taxed by this legacy 1500 byte MTU.” Currently, the demand and the need for applications such as multimedia over a network or the internet are increasing rapidly. Typical uses are video conferencing, and the need to transfer images such as an x-ray over the network. The number of video clips that are shared over the internet has also increased.

Silvestre, Sempere and Albero (2005) stated that MTU limitations had been one of the “characteristics that limit throughput in these networks” and also “restricts the range of multimedia applications viable over these infrastructures.” Silvestre, Sempere and Albero (2005) also stated that “in Ethernet, the limitation of 1500 bytes is currently being surpassed by the use of jumbo-frames, which offer a maximum frame size of 9 Kb.” Jumbo frame is a new method that can be used to solve this issue. Therefore, it is necessary to study the performance of this new solution within a real network environment in order to find out how jumbo frames perform.

Jumbo Frames is still an emerging technology as Chelsio Communications (2010) stated that “it has been a decade since jumbo Ethernet frames were first proposed, and an IEEE standard or significant deployment are yet to be seen.” As a result, this study is conducted to evaluate and analyse the performance of a network utilising this new technology for transferring data from source to destination. There are four main metrics that this study will focus on in order to measure the precise performance of a network and they are, the throughput, delay, jitter and the CPU usage. As Galati and Greenhalgh (2009) explain the, “performance of networks is generally estimated through characteristic indices such as delivery ratio, latency, delay, jitter and loss.” These metrics will be measured on six operating systems platforms; Microsoft Windows Server 2008, Microsoft Windows 7 Professional, Microsoft Windows Server 2003, Linux Ubuntu, Fedora and OpenSUSE with Jumbo Frames using TCP and UDP for transmission on a network environment implementing IPv4 and IPv6.
1.1 Structure of the Report

This section will provide an overall outline of the structure of this document. There are a total of seven chapters in this report; following is the structure in details.

1. Chapter one provides an introduction and an overview of what initiates this study, followed by an outline of the structure of this document.

2. Chapter two features the review of all the literatures related to this study. This includes the two Internet protocols and the two transport protocols. This chapter also contains the review of literature regarding the tools used in this study and also the large Ethernet frame sizes which is the centre of this research.

3. Chapter three presents the hypotheses of this study, also the methodology employed for this research. This chapter also will present the method used in the data collection phase of this study.

4. In chapter four, a detailed explanation of the specification of all hardware employed in the data collection phase of this research. The hardware devices were setup and used for the experiments in the test lab. This chapter will also provide the configurations used during this phase and also present the diagram of the network setup used.

5. Chapter five of this document will present in line charts the representations of the data gathered during the experiments and test phase. This chapter also contains the discussions and analysis of this data.

6. Chapter six presents the findings from this study and an in depth discussions of the findings.

7. The final chapter will be the conclusion of this study.
Chapter 2: Literature Review

In this literature review, prior research and studies had been explored regarding the domain. This chapter consists of a brief description of the two main internet protocols, which are IPv4 and IPv6. This is important as it provides a logical understanding of why these two protocols are included in this study. The problems and opportunities offered by these protocols are also presented. Discussion and explanation of why the performance of a network is important to be monitored at all times is also given.

In this chapter a discussion of the use of the two main transmission protocols (i.e. TCP and UDP) their features and the differences between these will two will also be presented. This study has been carried out using the standard Ethernet packet sizes and as well as the emerging technology known as Jumbo frames. This chapter will also discuss the features, differences and importance of Jumbo frames.

2.1 Literature Background

In this section, background literature related to the research topic is presented.

2.1.1 Networking and Network Performance.

There are different types and sizes of networks in the world today but, size does not matter as the performance of the network is important regardless of size. It is therefore important to obtain an understanding of what a computer network is and why network performance is monitored and measured. A computer network is a set of computers using common protocols to communicate over connecting transmission media (Quarterman and Hoskins, 1986). Quarterman and Hoskins explicate that in a network, computers must be interconnected. Further, for communication to happen, computers need to be able to speak the same ‘language’. The need for a computer network is all about sharing resources and this has led to the development of very complex and complicated networks.

In today’s business world, networking is further complicated because there is a need to communicate with devices such as PDAs, laptops, telephones and faxes. Ismail and Zin (2008) stated that, “today, retrieving and sending information can be done using a variety of technologies such as PC, PDA, fix and mobile phones via the wireless, high speed network.” Devices such as PDAs and phones are usually isolated but nowadays the demands to share company resources with them is high and this has been enabled by computer networks. Nonetheless, those are not the only reasons but also as following:

- Companies save money by sharing resources such as printers, scanners and other necessary hardware.
• Costs are reduced for software applications by centralising resources and sharing rather than running on individual machines.

• Networking is a key to effective and efficient communication between staff members regardless of their location.

Stallings (2007) stated that, "the opportunities for using networks as an aggressive competitive tool and as a means of enhancing productivity and slashing costs are great."

There are five main types of networks. These are; Personal Area Network (PAN), Campus Area Network (CAN), Local Area Network (LAN), Metropolitan Area Network (MAN) and Wide Area Network (WAN). Mitchell (2010) stated that “the different types of computer network designs are by their scope or scale. For historical reasons, the networking industry refers to nearly every type of design as some kind of area network”. As Mitchell explains, the type of network can be identified by its size; the name can be self-explanatory. For instance, Campus Area Network refers to a network within a campus. However, regardless of the networks types and sizes, it is still important to optimise their performance.

A critical task for network administration is network performance where the flow of all network traffics is monitored to identify and rectify bottlenecks. Lu, Zhang, Fu, Qian, Gumaste and Zheng (2007) also pointed out that the fundamental design philosophy of a network is also important, “we argue that the fundamental issue for a communication network is to provide connectivity, which means that one node in the network shall be able to send data to another node within a certain amount of time.” Ismail and Zin (2008) stated that, “the main factors of network congestion are related to network design and bandwidth capacity.” In order to do away with any of these factors, network administrators must monitor and measure the networks performance based on certain metrics.

Within network connectivity as discussed above, there are performance levels and metrics that must be monitored. The main principles are to enhance the effectiveness and performance of the network. McCabe (2003) stated that, “performance is the set of levels for capacity, delay, and RMA in a network.” To obtain the maximum performance for a network McCabe (2003a) also stated that “it is usually desirable to optimize these levels, either for all (user, application, and device) traffic flows in the network or for one or more sets of traffic flows, based on groups of users, applications, and/or devices.” The user, device and application requirements are all important requirements that need to be met in order to eliminate bottlenecks.
McCabe (2003b) also stated that networks need high performance or “guaranteed behaviour to support user requirements for timeliness, interactivity, reliability, quality, adaptability, and security.” It is important to mention here that timeliness and interactivity are also required. Network users require a system to perform within a certain time period. Users do not like to wait, they want to access the network resources and transfer their files in certain amount of time. McCabe (2003b) stated that, “end-to-end or round-trip delay can be a useful measurement.” On the other hand, interactivity is in the same category with timeliness but the centre of attention is on the response time of the network.

2.1.2 Performance Analysis

Analysing and evaluating the performance of a network has been studied heavily by researchers globally. Much of the focus has been on structuring and measuring the performance of network services and technologies. Ismail and Zin (2008) stated that, “accurate measurements and analysis of network characteristics (remote data transfers) are essential for robust network performance and management.” In view of this fact networking and its services are important for supporting critical day-to-day operations of organisations and businesses around the world. It is critical that the performance of the network should be at an optimal level at all times. To meet such an optimum level, it is necessary to monitor and measure network performance to identify any bottlenecks. To achieve this, there are known network parameters that can be measured. These parameters are called, network performance metrics. According to Yunos, Noor and Ahmad (2010), “the performance of the network and parts of networks can be affected by errors such as jitter, datagram or packet loss, latency, poor transfer rates and bandwidth quality.” These four metrics will be extracted from the data collected during this study when the analysis is conducted.

Network throughput, also referred to as transfer rate or bandwidth, is one of the important factors in network performance; throughput is one performance metric with which network administrators around the world are concerned. Isaksson, Chevul and Fiedler (2005) stated that, “throughput is thus one of the most essential enablers for networked applications, if not the most important one”. This is important because it is concerned with the amount of data a network can transmit within a certain time period. Dean (2009) also suggests that throughput is the “measure of how much data is transmitted during a given period of time.” There are other known factors in the network that can affect throughput such as, speed of the processor, capability of the network adapter used, and the medium capacity. Other processes that are running on the network can also affect performance. Table 2.1 on the next page shows the unit, its abbreviation and quantity.
In Table 2.1 above, the units that will be used in this study to measure the throughput of a network are detailed. This study will measure throughput in the amount of bits, and because in today’s network, the capacity or bandwidth can go up to a maximum of Gigabits. Therefore, this study will measure throughput in Megabits per second, for instance 1Mbps, in order to eliminate displaying large numbers such as 1,000,000 bits per second.

McCabe (2003c) stated that delay is a “measure of the time differences in the transfer and processing of information. There are many sources of delay, including propagation, transmission, queuing, processing, and routing”. This study will measure delay based on one way (OWD) only, from source to destination. McCabe also stated that delay is the time difference between the data being transferred from source to destination. When data is sent from its source, there is process that is required before the data reaches its destination. The data may go through routers and switches to reach its destination, and the processing time taken by these devices may cause delay. The less delay there is the better. However, other factors in the network that can contribute to this metric are the distance from source to destination, the amount of packets send and the quality of the medium used (cable).

Ying, Jun, Li-ke and Hong-da (2011) defined jitter as, “a variation in the delay of received packets.” There are factors within a network that causes jitter. When a user sends data to another user in a network, data splits into packets and those packets are sent out continuously and evenly distributed via the medium. Ying et al (2011) stated that, “due to network congestion, improper queuing, or configuration errors, this steady stream and become lumpy, or the delay between each packet can vary instead of remaining constant.”

CPU usage is a measure of how hard a processor works to process information that passes through the system. The CPU usage is measured in percentage, which can be a sign of how much of the CPU’s processing power is in use. However, Brandt, Nutt, Berk and Mankovich (2002) stated that, “the maximum CPU usage number depends on the system on which the application is being executed” indicating that, the CPU usage of an old computer with low
specification will use higher a percentage of its CPU to that of a computer with new specifications when processing a large number of data.

Packet loss is a sensitive subject in the field of communication. In fact, no one wants sensitive data to be lost. Yet, this happens in networking and can be as a result of any queuing delay. Packet loss happens when packets encounter a router and the queue is too long. Depending on the protocol being used, for instance UDP, the router will drop these packets. Conversely, Kurose and Ross (2010) stated that, “a packet can arrive to find a full queue: With no place to store such a packet, a router will drop that packet; that is, the packet will be lost.” In case that packet loss takes place, then throughput also drops accordingly.

As discussed, throughputs, delay (latency), jitter and packet loss are all important factors to be measured in order to optimize network performance (Yunos et al, 2010). However, in order to see how much of the processors processing power and time is used, it is also necessary to include the CPU utilisation in this study. There are two approaches that are usually used by researchers for measuring network performance. According to Gan, Zhang and Qian (2009), “there are two network measurement methods. One is active measurement, the other one is passive measurement.” To measure the performance of a network, the researcher needs to know what to look for and which method to use in order to collect the right data for analysis.

According to Gan, Zhang and Qian (2009), the passive method refers to the, “process of measuring a network, without creating or modifying any traffic on the network. It collects the traffic and monitors the related packets though the link by laying agent or probe on the network.” This means that passive measurement methods such as, a traffic monitor tool can be placed anywhere in the network, and this monitoring tool observes the traffic when data passes through such a link. However, there is a problem with this method. According to Jaiswal, Iannaccone, Kurose and Towsley (2007) they stated that, “the nature of the measurements also presents some challenging problems. Since the monitor can be located anywhere on the end-end path, it has limited knowledge of all the events that occur along the path.” With this method, the true performance reading of the network may not be disclosed.

Active measurements are another monitoring tool. According to Ma, Yan and Huang (2003), “active measurement methods are typically used to obtain end-to-end statistics such as latency, loss, and route availability.” This method, such as experimenting in a test lab is widely used by researchers and it usually involves setting up a test bed using necessary network devices in a test laboratory. The traffic will be generated within laboratory conditions.
in order to measure and collect the data. Those results are considered to be the closest to a measuring in a real network environment.

However, there are two other methods that are also used currently for measuring network performance, namely emulation and simulation. Emulation is a method that is well known and used widely. There have been a number of studies that have used this method in the past such as Kim, Hwang and Yoo (2008), Gu and Fujimoto (2008) and Yan-hui and Tao (2009). The emulation method is a mixture of both emulation and simulation. Zheng and Ni (2004) stated that, “network emulation is real-time simulation that enables test and evaluation of real network systems, protocols, and applications in a reconfigurable and controllable hardware and software environment.” For instance, researchers may have only two computers and a few cables but instead of a using a real router, they may replace this with emulation software. Researchers also use an emulation method to model real network environments.

Network emulation enables researchers to test and evaluate a real network system such as protocols, links, hardware and software in a controlled environment. If the target network is too big then it will be very expensive to conduct the evaluation. In this case, according to Zheng and Ni (2004), “the target network has to be abstracted to a single router modelled with some performance metrics.” Thus, emulators are simply abstracted in a real network environment and as a result, they do not reflect the true elements of a real network. For instance, network operating system and routing protocols because emulators employ one-to-one node evaluations.

Network simulation on the other hand is also widely used by researchers. This tool is used to provide an imitation of the real network. Mehta, Ullah, Kabir, Sultana and Kwak (2009) stated that, “simulations are a good compromise between cost and complexity, on the one hand, and accuracy of the results, on the other hand.” There have been a number of studies that used this method in the past such as Oliveira and Silva (2008), Ziolkowski and Brauer (2010) and Zheng, Tang and Zhang (2008). In a survey conducted on simulator tools by Mehta et al (2009), some of the widely used simulator tools used by researchers were Ns-2, GloMoSim, J-Sim, OMNet++, OPNet, and QualNet. Zheng and Ni (2004) stated that, “for large network simulation, simulators will consume a large amount of time and memory, and its result is largely based on some modelling assumptions that may not hold in the real world.” Simulation methods are cost effective as they do not have a heavy demand on equipment. However, this may compromise the accuracy of the results gathered from such a study. This is due to the fact that simulation implementation depends on software (usually developed by the researcher). This also means that network operating systems are not included in a simulation environment or performance metrics such as delays, which are found in real network environments.
2.2 Internet Protocol

Internet Protocol was established about 30 years ago as a result of a project conducted by Advanced Research Project Agency Network (ARPANET). According to Vasseur and Dunkels (2010), “ARPANET was the first operational network using the concept of packet switching, which was at that time a revolutionary approach for inter-host communication”. The Internet Protocol is fundamentally the primary protocol used by Information Technology across the global communication infrastructure. This is used for the transmitting of data from source to destination by using the Internet Protocol suite also known as the Internet Protocol stack. IP suite or IP stack is made up of four concept layers namely the Link layer, the Internet layer, and the Transport layer with the top layer being the Application layer. Each of these layers is characterized by its functionality. For instance, the Link layer focuses on providing connectivity for the network in which all the hosts are connected. The Internet layer provides ways in which communication should happen and in this layer, and the Internet protocol is located. The version of the Internet protocol that is widely used in the world today is known as Internet protocol version 4 (IPv4). Meng, Xu, Zhang, Huston, Lu and Zhang (2005) also stated that, “the original IP design divided address space into three different address classes {Class A, B, and C}. Each class had a fixed boundary between the network-pre-x and the host-number within the 32-bit IP address.” The following Table 2.2 below outlines IPv4 three main classes that are currently used in today’s network.

<table>
<thead>
<tr>
<th>IPv4 Classes</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>0</td>
<td>126</td>
</tr>
<tr>
<td>Class B</td>
<td>128</td>
<td>191</td>
</tr>
<tr>
<td>Class C</td>
<td>192</td>
<td>224</td>
</tr>
</tbody>
</table>

Table 2.2: IPv4 three main classes

Notice that between class A and class B, 127 is missing Bryant (2011) stated that this is because, “127 is the reserved first octet for loopback addresses, such as the 127.0.0.1 address assigned to a PC.” Bryant (2011) also stated that, “Class D networks are reserved for multicasting. Class D addresses begin with an octet in the 224 - 239 range. Class E networks are reserved for “experimental use”, and the first octet of these addresses is 240 - 255”. Table 2.2 in the previous page shows three different main classes of IPv4 addresses. In IPv4 addressing scheme, there are four octets, with each octet containing eight bits in binary, as a result the complete address space of IPv4 comes to 32 bits in binary. In another illustration, IPv4 has $2^{32}$ addresses which equals to 4,294,967,296 in decimal numbers.

IPv4 was designed to be used when sending data across the network. Not only does this happen but IP also provides, according to Postel (1981), “fragmentation and reassembly of
long datagrams, if necessary, for transmission through "small packet" networks.”

Fragmentation and reassembly is a very important feature of IP. Network device such as a network interface card can define the maximum unit that can be transferred. When this device receives an IP packet where the payload is more than the defined MTU, the device will fragment the data. Each segment must be equal to or less than the defined MTU. As a result, each segment of the fragmented data will be flagged, and this way the receiving side can tell that this data has more than one segment and is able to identify the segments. When the first packet arrives at the receiving side, the device will see the flag and it can then tell that this packet is a part of a fragment. The receiver will then store the packet until it receives them all.

Yunos, Noor and Ahmad (2010) stated that IPv4 is, “one of the many protocols that serve the end users is the Internet Protocol version 4 (IPv4) addressing scheme.” This extensively used IPv4 has now run out of addresses. Yet again, Yunos, Noor and Ahmad (2010) appropriately stated that, “with the tremendous growth of the Internet, the current address provided by IPv4 has proven to be insufficient and inadequate.” This is the problem that is being faced at present that, IPv4 can no longer meet the demand for an IP address.

As discussed in section 2.2 above, as a result of the incredible growth of the internet, IPv4 addresses are running out. In fact, according to Fletcher (2011), “the Internet Address and Naming Agency allocated the last blocks of internet addresses on Tuesday, with three out of the eight remaining blocks going to the Asia Pacific region.” Another report from the Xinhua News Agency (2011) stated that, “the pool of available unallocated addresses for the existing Internet protocol has now been completely emptied, the organization that oversees the allocation of Internet addresses announced on Thursday.” In Chapter 1, Table 1.1 outlined the statistics of the world Internet usage and its corresponding populations. It is evident that from the year 2000 to June 2010, the growth of Internet users increased by 480.4%. Now it is understandable why IPv4 addresses ran out this year. As a result, the Internet Engineering Task Force (IETF) developed a solution, to rectify this dilemma caused by the collapse of IPv4. Chengqing, Yinglong and Jizhi (2007) stated that, “the Internet Engineering Task Force (IETF) has introduced IPv6 with a mission to meet the growing demands of the future Internet.” In the following section is a discussion of the IETF solution that is, IPv6 also known as the Next Generation Internet Protocol.

During the development period of IPv6, there were temporary solutions that were used to help in alleviating the predicament of IPv4 running out of addresses. As a result of IPv4 running out of IP addresses, one of the temporary solutions that were introduced was the Classless Inter-domain Routing (CIDR). Frankel and Green (2008) stated that, “CIDR enables a variable-sized network prefix, allowing more flexibility in the number of addresses
allocated address blocks.” With CIDR, large organisations with multiple networks can now allocate multiple class C network IDs instead of, one class B network ID. However, CIDR managed to solve that problem. On the other hand, with multiple class C network IDs, every network ID needs an entry into the routing table. Nevertheless, Stevens (2004) stated that CIDR, “is a way to prevent this explosion in the size of the Internet routing tables. It is called supernetting.” CIDR was a success since it met the purpose of its existence with regards to minimizing the need for multiple entries into the routing table. This is due to the fact that, CIDR allowed blocks of addresses to be grouped, so they can have only one entry in the routing table of the router.

CIDR had a mission and it was successful because, CIDR can group block of addresses. This means that for those block of addresses, they only need one entry in to router’s routing table. On the other hand is the Network Address Translation (NAT). NAT is one of the temporary solutions that were introduced to alleviate the pressure. This also helped at the same time to slow down IPv4 from running out. NAT as explained by Frankel & Green (2008) is, “NAT lets multiple devices use a single IP address” NAT also helps to minimise the cost of having to buy public IP addresses for all the hosts in the network. As mentioned in the previous paragraph by Mr. Frankel and Mr. Green, NAT allows all the hosts in a private LAN to connect to the outside world through NAT using one public IP address. This way, it not only helps to hide the private LAN from the outside world but also minimised the use of public IP addresses.

Frankel and Green (2008) stated how NAT works, “a NAT box, situated between the private network and the Internet, manipulates portions of the header (most frequently, the port number) so that it can distinguish which incoming traffic is intended for each device on the network.” A very simple and good example of NAT is an Internet connection from home. At home, there may be two or three computers all connecting to the Internet at the same time. This is made possible by using a device known as, an ADSL modem router. This device has NAT functionality within it. All the home computers each of which has their own IPv4 address, but when connecting to the Internet they use one public IP address. As mentioned earlier, CIDR and NAT were temporary solutions to slow down IPv4 exhaustion while they worked on developing IPv6.

IPv6 was developed by the IETF to overcome the inadequacy of IPv4. The 128 bit address space of IPv6 is beyond anyone’s imagination. According to Beijnum (2006) it is, “340,282,366,920,938,463,463,374,607,431,768,211,456” for IPv6 while there is only “4,294,967,296” possible addresses for IPv4. With the 128 bit address space, in U.S. measurements, Beijnum (2006) stated that, “it’s enough to supply every square inch of the Earth’s surface with the equivalent of a hundred trillion IPv4 Internets.” Yet, meeting the
The ever-increasing demand for IP addresses of the Internet, is not the only advantage of the Next Generation Internet Protocol but, Chengqing, Yinglong and Jizhi (2007) stated that, “IPv6 was designed not only to increase the address space, but also includes unique benefits such as scalability, security, simple routing capability, easier configuration “plug and play”, support for real-time data and improved mobility support.” As explained earlier, IPv6 has more addresses than anyone can imagine but, it also comes with other features such as simple routing capability, scalability and more importantly, security.

IPv6 has full support for IPSec, and IPv6 is more secure when compared to IPv4. However, IPSec also works with IPv4 but the problem is, Network Address Translation. IPSec was designed to encrypt and authenticate packets at IP level. As a result, when employing IPSec in a network that uses NAT, IPSec will drop a packet once it detects the change made by NAT. Beijnum (2006) stated that, “IPv6 does have one security advantage over IPv4, though: in IPv6, an Ethernet usually gets 64 bits to number hosts. With IPv4, this is never more than 16 bits and is often much less.” In a simplified version, an intruder will find it more challenging to hack into an IPv6 subnet than breaking into an entire IPv4 Internet. Not only because of the 128 bit address space of IPv6 but also because of its full support for the IP security feature.

The address space of IPv6 is 128 bits long while the IPv4 address space is only 32 bits long. IPv6 header is 40-Bytes, which is double the 20 Bytes header of IPv4. Zeadally and Raicu (2003) stated that, “the overall impact of this increase is 1.32 percent higher overhead with IPv6 for an Ethernet maximum transmission unit (MTU) size of 1,514 bytes.” Although that the increase is only 1.32 percent. Zeadally and Raicu (2003) also claimed in their study that, “that the real difference between IPv4 and IPv6 is much larger than predicted.” However, Frankel and Green (2008) pointed out that, “even with the larger source and destination addresses, the working group designed the IPv6 header for better transmission performance by aligning the header frame for 64-bit processing in routers and switches and eliminating the redundant header checksum field.” That is quite true, although the address space of IPv6 is 4 times larger when compared to that of IPv4, the header is only two times larger, the processing of an IPv6 packet will be more efficient than an IPv4 packet. However, that is not the only enhancement that comes with IPv6. Following is an outline of some efficiency enhancements that IPv6 brings:

- The IPv6 header has a fixed length
- The IPv6 header is optimized for processing up to 64 bits at a time (32 in IPv4)
- The IPv4 header checksum that is calculated every time a packet passes a router was removed from IPv6
- Routers are no longer required to fragment oversized packets; they can simply signal the source to send smaller packets.
- All broadcasts for discovery functions were replaced by multicasts. Only hosts that are actively listening for multicasts are interrupted, rather than all hosts, as with broadcasts Beijnum (2006).

With all the enhancements that IPv6 brings, ideally IPv6 should be more effective and efficient to use. IPv6 has developed a lot of interest within the network research community including studies conducted by Murugesan, Ramadass and Budiarto (2009), Chengqing Yinglong and Jizhi (2007) and Narayan, Shang and Fan (2009). Jointly, these researchers focussed on evaluating and analysing the performance of a network that implements IPv6. The major concern is the size of IPv6’s overhead, it doubles the IPv4’s overhead size, and it may degrade a network’s performance. However, these researchers have a totally different set of hardware and as a result; it is very hard to compare them. Table 2-4 shows the IPv4 Header fields and their corresponding IPv6 Header fields.

<table>
<thead>
<tr>
<th>IPv4 Header Field</th>
<th>IPv6 Header Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Same field but with a different version number.</td>
</tr>
<tr>
<td>Internet Header Length</td>
<td>Removed in IPv6. IPv6 does not include a Header Length field because the IPv6 header is always a fixed length of 40 bytes. Each extension header is either a fixed length or indicates its own length.</td>
</tr>
<tr>
<td>Type of Service</td>
<td>Replaced by the IPv6 Traffic Class field.</td>
</tr>
<tr>
<td>Total Length</td>
<td>Replaced by the IPv6 Payload Length field, which indicates only the size of the payload.</td>
</tr>
<tr>
<td>Identification Flags</td>
<td>Removed in IPv6. Fragmentation information is not included in the IPv6 header. It is contained in a Fragment extension header.</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td></td>
</tr>
<tr>
<td>Time-to-Live</td>
<td>Replaced by the IPv6 Hop Limit field.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Replaced by the IPv6 Next Header field.</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>Removed in IPv6. The link layer has a checksum that performs bit-level error detection for the entire IPv6 packet.</td>
</tr>
<tr>
<td>Source Address</td>
<td>The field is the same except that IPv6 addresses are 128 bits in length.</td>
</tr>
<tr>
<td>Destination Address</td>
<td>The field is the same except that IPv6 addresses are 128 bits in length.</td>
</tr>
<tr>
<td>Options</td>
<td>Removed in IPv6. IPv6 extension headers replace IPv4 options.</td>
</tr>
</tbody>
</table>

Table 2-3: IPv4 header field and its corresponding IPv6 header field. (Davies, 2008)

As shown in Table 2-3, four of the IPv4 Header fields are no longer in use in the IPv6 header, they are as follows: The Internet Header Length has been removed in IPv6 due to
the fact that the header length in IPv6 is fixed to 40 bytes. The other three that have been removed are Identification Flags Fragment Offset, the Header Checksum field and the Option field. Table 2-4 below outlines some of the key differences between IPv4 and IPv6.

<table>
<thead>
<tr>
<th>Internet Protocol version 4</th>
<th>Internet Protocol version 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source and destination addresses are 32 bits (4 bytes) in length.</td>
<td>Source and destination addresses are 128 bits (16 bytes) in length.</td>
</tr>
<tr>
<td>IPSec header support is optional.</td>
<td>IPSec header support is required.</td>
</tr>
<tr>
<td>No identification of packet flow for prioritized delivery handling by routers is present within the IPv4 header.</td>
<td>Packet flow identification for prioritized delivery handling by routers is present within the IPv6 header using the Flow Label field.</td>
</tr>
<tr>
<td>Fragmentation is performed by the sending host and at routers, slowing router performance.</td>
<td>Fragmentation is performed only by the sending host.</td>
</tr>
<tr>
<td>Has no link-layer packet-size requirements, and must be able to reassemble a 576-byte packet.</td>
<td>Link layer must support a 1280-byte packet and be able to reassemble a 1500-byte packet.</td>
</tr>
<tr>
<td>Header includes a checksum.</td>
<td>Header does not include a checksum.</td>
</tr>
<tr>
<td>Header includes options.</td>
<td>All optional data is moved to IPv6 extension headers.</td>
</tr>
<tr>
<td>ARP uses broadcast ARP Request frames to resolve an IPv4 address to a link-layer address.</td>
<td>ARP Request frames are replaced with multicast Neighbor Solicitation messages.</td>
</tr>
<tr>
<td>Internet Group Management Protocol (IGMP) is used to manage local subnet group membership.</td>
<td>IGMP is replaced with Multicast Listener Discovery (MLD) messages.</td>
</tr>
<tr>
<td>ICMP Router Discovery is used to determine the IPv4 address of the best default gateway and is optional.</td>
<td>ICMPv4 Router Discovery is replaced with ICMPv6 Router Solicitation and Router Advertisement messages, and it is required.</td>
</tr>
<tr>
<td>Broadcast addresses are used to send traffic to all nodes on a subnet.</td>
<td>There are no IPv6 broadcast addresses. Instead, a link-local scope all-nodes multicast address is used.</td>
</tr>
<tr>
<td>Must be configured either manually or through DHCP for IPv4.</td>
<td>Does not require manual configuration or DHCP for IPv6.</td>
</tr>
<tr>
<td>Uses host address (A) resource records in the Domain Name System (DNS) to map host names to IPv4 addresses.</td>
<td>Uses AAAA records in the DNS to map host names to IPv6 addresses.</td>
</tr>
<tr>
<td>Uses pointer (PTR) resource records in the IN-ADDR.ARPA DNS domain to map IPv4 addresses to host names.</td>
<td>Uses pointer (PTR) resource records in the IP6.ARPA DNS domain to map IPv6 addresses to host names.</td>
</tr>
</tbody>
</table>

Table 2-4: Comparing the main differences of IPv4 and IPv6. (Davies, 2008)

Table 2-4 in the previous page shows some of the significant differences between IPv4 and IPv6. When looking closely at these differences, it can be seen that IPv6 was not designed
only to meet the demands for IP addresses. IPv6 was also designed for better performance, efficiency and security as well.

### 2.3 The Transmission Protocols

When travelling from home to work or home to school day by day, there are certain sets of rules to abide with as a road user. The vehicle that we use for transportation has its own set of rules to maintain. In this section of this chapter, this study will discuss and compare the two vehicles that will be used in this study for transmissions and they are, TCP and UDP. Within a computer network, protocol is a set of rules that oversee how data is delivered from sender to the receiver. The Open Systems Interconnection reference model was introduced by the International Standardisation Organisation (ISO) in the late 1970s. This reference model is widely used as a structure to relate all the functions that a computer network entails for data communication. Following is a brief explanation of the seven layers of the OSI model.

- The application layer is responsible for the creation of the users’ data, from word document to a simple e-mail to send via the network.
- The presentation layer is the layer that is responsible for making communication between sender and receiver possible. For instance, encrypting and decrypting, protocol conversion and compression.
- The session layer is responsible for opening and closing of the communication process between source and destination.
- The transport layer is the one that is responsible for everything regarding the transportation of data from source to destination. For instance, the transport layer is required to divide the data into small pieces or packets before sending. When a packet arrives at destination host, the transport layer is responsible for putting these packets back together.
- The network layer is where all the translation of addresses happens. The network layer is required to look at the address or name on the packet and translate it into a physical address (MAC address). In this layer also is where all the switching and routing is determined.
- The data link layer is responsible for error correction as there may be errors created by the physical layer. These errors are corrected in the data link layer and determine other routes to send the packets through to its destination.
- The physical layer is the lowest layer of the OSI model. This is the connection point of the other layers of the OSI model with the physical medium in the network such as, cables, and network interface cards.

Since the Internet came in to its existence, and due to its ever increasing rate, the Defence Advanced Research Project Agency (DARPA) decided to develop a simplified model and
they introduced the TCP/IP model. The TCP/IP model has only four layers, which correspond with the OSI reference model. Figure 2-1 on the next page illustrates the layers of the TCP/IP model.

Figure 2-1: The TCP/IP model. (TechNet, 2005)

Figure 2-1 above shows the four layers of the TCP/IP model and the protocols that each layer uses. The Application layer combines the top four layers of the OSI reference model and this is called the Application layer. The Transports layer of the OSI model is still called the Transport layer. The Data link layer becomes the Internet layer of the TCP/IP model while, the Physical layer becomes the Network Interface layer of the TCP/IP model. Following in Table 2-6 below is a comparison of the two transmissions protocol that will be used in this study.

<table>
<thead>
<tr>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reliability:</strong> TCP is connection-oriented protocol. When a file or message send it will get delivered unless connections fails. If connection lost, the server will request the lost part. There is no corruption while transferring a message.</td>
<td><strong>Reliability:</strong> UDP is connectionless protocol. When you a send a data or message, you don't know if it'll get there, it could get lost on the way. There may be corruption while transferring a message.</td>
</tr>
<tr>
<td><strong>Ordered:</strong> If you send two messages along a connection, one after the other, you know the first message will get there first. You don't have to worry about data arriving in the wrong order.</td>
<td><strong>Ordered:</strong> If you send two messages out, you don't know what order they'll arrive in i.e. no ordered</td>
</tr>
<tr>
<td><strong>Heavyweight:</strong> when the low level parts of the TCP &quot;stream&quot; arrive in the wrong order, resend requests have to be sent, and all the out of sequence parts have to be put back together, so requires a bit of work to piece together.</td>
<td><strong>Lightweight:</strong> No ordering of messages, no tracking connections, etc. It's just fire and forgets! This means it's a lot quicker, and the network card / OS have to do very little work to translate the data back from the packets.</td>
</tr>
<tr>
<td><strong>Streaming:</strong> Data is read as a &quot;stream,&quot; with nothing distinguishing where one packet ends and another begins. There may be multiple packets per read call.</td>
<td><strong>Datagrams:</strong> Packets are sent individually and are guaranteed to be whole if they arrive. One packet per one read call.</td>
</tr>
<tr>
<td><strong>Examples:</strong> World Wide Web (Apache TCP port 80), e-mail (SMTP TCP port 25 Postfix MTA), File Transfer Protocol (FTP port 21) and Secure Shell (OpenSSH port 22) etc.</td>
<td><strong>Examples:</strong> Domain Name System (DNS UDP port 53), streaming media applications such as IPTV or movies, Voice over IP (VoIP), Trivial File Transfer Protocol (TFTP) and online multiplayer games etc</td>
</tr>
</tbody>
</table>

Table 2-5: Comparing TCP and UDP (Gite, 2007).

Table 2-5 provides a very comprehensive comparison of the two main transmissions protocol (TCP/UDP) that will be used in this study. As Petrovic and Aboelaze (2003) stated, “TCP is the De facto standard for connection oriented transport layer protocol, while UDP is
the De facto standard for transport layer protocol.” Yeh, Lin, Chung and Chih (2010) stated that, “network reliability is one of the useful decision support measures in management science. It has been applied in many real-world systems such as oil/gas production systems, computer and communication systems, power transmission and distribution systems, and transportation systems.” Network reliability is an important aspect in today’s world. As shown in Table 2-5 above, under TCP and UDP reliability, TCP is a connection-oriented protocol while UDP is a connectionless protocol. This means that TCP is more reliable than UDP because TCP will endeavour to establish the connection first from the sender to the receiver’s host before commencing the transmission. Not only that, Table 2-6 shows that TCP is a heavyweight protocol. When talking about the organisation of packets during transmission, Table 2-5 shows that UDP has no order. If 10 packets are sent using UDP, then it is not know which one will arrive first.

2.4 The Network Performance Measurement Tools

The rapid growth of the Internet, which is the largest network in the world, has led to many researchers developing an interest in monitoring, analysing and measuring traffic and performance of networks. Consequently, software developers have also developed network traffic and monitoring tools, network traffic generating tools and network performance measurement tools.

A requirement for this study was to select the most appropriate network performance measurement tool. As a result, a small study was required in order to determine the most suitable tool for this work. This small study found that there were hundreds of tools available that ranged from those that cost thousands of dollars to open source tools and free to use tools. Some tools have graphical user interface (GUI) mode while others needed to use command line mode. Since the different tools came with diverse functionalities, a selection criterion was developed to facilitate the selection process. Following in the next page is an outline of the selection criteria:

- The tool must support different distributions of Linux and Windows based platforms
- The tool must support both the currently used packet sizes and the Jumbo frame sizes
- The tool must be able to generate traffic, measure and display the results including all the required performance metrics i.e.
  - Throughput, Delay, Jitter and Packet loss
- The tool must support both of the Internet protocol’s versions i.e. IPv4 and IPv6.
- The tool must be able to generate traffic using TCP and UDP as transmission protocol.
- The tool must be able to generate traffic in the application level such as DNS, Games and Voice over IP traffics.
There were five possible candidates for this job; IP Traffic, IPPerf, Netperf, Multi-Generator (MGEN) and Distributed Internet Traffic Generator (D-ITG). In this section analysis, discussion and review of all of the above mentioned possible candidates is presented. Also, the reason why that tool was selected to use in this study is discussed.

2.4.1 IP Traffic

IP Traffic is a test and measure tool developed by a French Telecommunication company known as ZTI. This tool is distributed by a company known as Packet Data Systems Ltd from the United Kingdom. The IP Traffic tool can be run using GUI mode, below in Figure 2-2 illustrates type of network that IP Traffic test and measure.

![Figure 2.2: IP Traffic test and measure sample network](ZTI, 2009)

Figure 2-2 above shows the type of network setup that IP Traffic can be used in. ZTI (2009) explained that, “IP Traffic Test & Measure can generate, receive, capture and replay IP traffic (TCP, UDP or ICMP), and measure end-to-end performance and Quality of Service over any fixed or mobile IP network (IPv4 & IPv6): LAN, MAN, WAN, WWAN, satellite, PLC, mobile, FTTx”. However, IP Traffic is a tool for commercial use that requires a license to use. IP Traffic generates traffics for TCP, UDP and ICMP also. IP Traffic can also generate traffics both in the packet level and application level. IP Traffic is an advanced traffic generation tool that can have up to 16 connections simultaneously. IP Traffic tool can also synchronise clock using satellites. IP Traffic can test and measure any network regardless of its size and type. According to Packet Data Systems (2009), “IP Traffic Test & Measure requires at least two PC’s running on Microsoft Windows 2000, XP, Server 2003 or Vista. Various testing configurations may be implemented using more than two PC’s.” IP Traffic can test for all the performance metrics that this study required, however, IP Traffic did not meet the selection criteria in terms of operating systems requirements, for instance, Microsoft Windows Server 2008, Microsoft Windows 7 and the Linux distributions.
2.4.2 Iperf

Rutgers (2006) stated that IPerf, “is a tool used to measure maximum TCP bandwidth between a client and server. LSS maintains an IPerf server that can be used to report information about bandwidth, delay jitter, and datagram loss.” IPerf was developed by a team known as Distributed Applications Support Team (DAST). Jduganand Mitchkutzko (2011) also stated that, “IPerf was developed by NLANR/DAST as a modern alternative for measuring maximum TCP and UDP bandwidth performance IPerf allows the tuning of various parameters and UDP characteristics. IPerf reports bandwidth, delay jitter, datagram loss.” IPerf is an open source tool meaning it is free to use, and no license is needed. IPerf can run on any platform such as Microsoft Windows, any Linux distribution or Mac OS. IPerf is a client/server network environment measurement tool, which means that it can have multiple simultaneous connections. IPerf can be used in LAN or over the Internet, or a WAN environment.

Although IPerf runs in a client/server environment it comes only in one package. The client/server functionality can only be executed by using a command. IPerf is a command line tool for instance; TCP port 5001 is the default port number that client can use to connect to the Iperf server. To run IPerf on server mode, use the following command line.

```
# iperf –s
```

IPerf will respond as follows,

```
--------------------------------------------------
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
--------------------------------------------------
(Pietroforte, 2007).
```

To run IPerf on client mode on the client machine use the following command.

```
# iperf –c
```

is then followed by the IP address of the server or the server name and accordingly, this is what will appear,

```
--------------------------------------------------
Client connecting to 10.0.0.5, TCP port 5001
TCP window size: 8.00 KByte (default)
--------------------------------------------------
[1912] local 10.0.0.211 port 1793 connected with 10.0.0.5 port 5001
[ ID] Interval Transfer Bandwidth
[1912] 0.0-30.0 sec 680 MBytes 190 Mbits/sec 
(Pietroforte, 2007).
```

In the above example, Pietroforte (2007) uses this command, “iperf -c 10.0.0.5 -t 30” where the server IP address is 10.0.0.5 and the (-t) option used to define how long IPerf will run that test for. The (30) means IPerf will that particular test for 30secs (Pietroforte, 2007).
2.4.3 **Netperf**

Jones (2007) stated that, “Netperf is a benchmark that can be used to measure various aspect of networking performance.” Netperf supports both TCP and UDP for sending tests data on two tests mode. According to Jones (2007), “As of this writing, the tests available either unconditionally or conditionally include:” The test environments currently considered by Netperf are as following:

- For unidirectional using TCP and UDP over both IPv4 and IPv6 using socket interface
- For unidirectional runs using TCP and UDP over both IPv4 using XTI interface
- For Link-level unidirectional run using the DLPI interface.
- For a Unix domain sockets
- Using Stream Control Transmission Protocol (SCTP) unidirectional run over both IPv4 and IPv6 using the sockets interface.

Netperf supports both the required platforms, which are Microsoft Windows and Linux. Netperf also supported UNIX and OpensVMS. Netperf is a command line tool only and its design is similar to Iperf client/server environment. According to Jones (2007), “netperf is designed around a basic client-server model. There are two executables - netperf and netserver. Generally you will only execute the netperf program, with the netserver program being invoked by the remote system's inetd or equivalent.” When Netperf runs, a control connection will be established with the remote machine. This connection can be either IPv4 or IPv6. This control connection will be using BSD socket and it will start a TCP connection. This connection is not only for the experimental test but it will also be used for sending information data between the client’s machine and the Netperf server. However, according to Jones (2007), “Netperf places no traffic on the control connection while a test is in progress.” This statement confirms that the traffic used for controlling the connection, will not interfere with the traffic generated for the tests.

2.4.4 **Multi-Generator (MGEN)**

Naval Research Laboratory (n/d) stated that the, “Multi-Generator (MGEN) is open source performance measurement tool that was developed by the Naval Research Laboratory (NRL) PROTocol Engineering Advanced Networking (PROTEAN) Research Group.” MGEN uses scripts to generate traffics for testing and these traffics are logged for analysis later when the run is finished. The model of the generated traffic on MGEN can be of either unicast or multicast or both using TCP or UDP. The Naval Research Laboratory (n/d) stated that the, “MGEN log data can be used to calculate performance statistics on throughput, packet loss rates, communication delay, and more.” MGEN only support UNIX based including MacOS and WIN32 platforms.


2.4.5 Distributed Internet Traffic Generator (D-ITG)

As the section header indicates, Distributed Internet Traffic Generator is the name of this tool, however, it is widely known as D-ITG. D-ITG was developed by the department of Information Systems at the Universita' degli Studi di Napoli "Federico II" Italy. D-ITG as according to Botta, Dainotti and Pescapè (2007), “is a platform capable to produce traffic at packet level accurately replicating appropriate stochastic processes for both IDT (Inter Departure Time) and PS (Packet Size) random variables (exponential, uniform, cauchy, normal, pareto ...).” D-ITG is a command line tool only and is currently available on Linux, Microsoft Windows and other Linux familiar platform. According to Botta, Dainotti and Pescapè (2007), D-ITG supports the following protocols, “TCP, UDP, ICMP, DNS, Telnet, VoIP (G.711, G.723, G.729, Voice Activity Detection, Compressed RTP).” With regards to delay, D-ITG can measure both one way delay (OWD) and round trip time (RTT). Not only that but D-ITG can also measure packet-loss, jitter and of course throughput. D-ITG is able to store the data file on both the sender and the receiver for decoding later. However, if you have to analyse the data on the fly, a dedicated log server can be set up to log the data and the data can be decoded and analyse right there and then.

![Figure 2-3: D-ITG software architecture](Botta, Dainotti & Pescapè, 2007).

As can be seen from Figure 2-3 above, D-ITG has four components namely, ITGSend server, which is used for sending the script that generates necessary traffics. ITGRecv server is use on the receiving side to capture the data file generated during the test. The ITGLog server is used for logging both the sending data and/or the receiving data. The ITGDec server is used for decoding the log data for analysis. ITGManager server is also there and it is used to start and close ITGSend, ITGRecv and ITGLog remotely if needed.

2.5 Performance Measurement Tool for this Study

Of all the possible candidates for this research, the decision had been made to use D-ITG. The decision was based on the selection criteria that were developed and the information that was available at the time. Following is the selection criteria:

- The tool must support different distributions of Linux and Widows based platforms
- The tool must support both the currently used packet sizes and the Jumbo frame sizes
• The tool must be able to generate traffic, measure and display the results including all the required performance metrics i.e.
  o Throughput, Delay, Jitter and Packet loss
• The tool must support both of the Internet protocol’s versions i.e. IPv4 and IPv6
• The tool must be able to generate traffic using TCP and UDP as transmission protocol
• The tool must be able to generate traffic in the application level such as DNS, Games and Voice over IP traffics.

Not only did D-ITG fulfill all the requirements shown above but there was also a comprehensive comparison analysis study was conducted on various performance measurement tools. The following Figure 2-4 shows the results of this study.

![Figure 2-4: Comparative study of traffic generators](Avallone et al, 2004).

Avallone, Guadagno, Emma, Pescape, and Ventre (2004) stated that D-ITG, “shows the best performance, as its received data rate is the closest to the expected one.” However, IPerf did very well also yet according to Avallone et al (2004), “Iperf does not produce a log file, it only provides an estimation of the received and transmitted date rate at the end of the experiment.” Another result provided by independent researchers which is quite interesting also. According to Narayan, Kolahi, Sunarto, Nguyen, and Mani (2009) stated, “in overall, we can conclude Netperf and D-ITG have a normal pattern have a similar behaviour when packet size changed. Both two performance software do not show any unsuspected point like IP Traffic and Iperf.” Figure 2-5, below illustrates the result of the study mentioned above.
As mentioned earlier in this section, the decision had been made to employ D-ITG as the tool to use in the data collection phase of this study. However, another important factor to be considered in the decision making regarding this subject is Jumbo Frames. Jumbo Frames is still a new and emerging technology. Jumbo Frames happens to be the main focus of this study consequently, as it is important to find a tool that supports Jumbo Frames. An e-mail was sent out to other tool developers and only D-ITG developer replied with the good news that D-ITG supports Jumbo Frames.

2.6 Jumbo Frames

Since Jumbo Frames is still a new and emerging technology, it is important to understand what Jumbo Frames are. As Chelsio Communications (2010) explained, “Jumbo Ethernet frames are ones that are larger than the maximum standard frame size of 1,522 bytes (with VLAN tag), typically up to 9,180 bytes.” Normal Ethernet frame sizes are between 46 bytes to 1500 bytes, as Chelsio Communications explained; anything up to 9180 bytes are Jumbo Ethernet frames. There can be no limit for Ethernet Frame sizes but there are reasons for limiting Jumbo frame size to 9000 Bytes at least for now. There are also other aspects to consider in terms of maintaining better networking performance. First of all, the traditional Ethernet frame uses 32 bit Cyclic Redundancy Check (CRC). CRC was introduced by Petersen in 1961 and it is still in use today. Chelsio Communications (2010) stated that, “Ethernet uses a 32 bit CRC that loses its effectiveness above about 12000 bytes.” Therefore, 9000 Bytes seems to be the most logical number to have however, until the 32 bit CRC is improved, there may not be much that can be done about this. Yet, the interest in using Jumbo frames is still developing and further research is required on Jumbo frames. NETGEAR (2010) stated that, “if most of your local network devices and applications can be
configured for the same Jumbo Frame size, your network may benefit from large packets, little fragmentation, and little overhead.” The use of multimedia applications over the Internet is rapidly increasing. Jalobeanu, Nistor, English and Wheeler (2003) stated that, “currently, there are over 10 million students online globally.” This is due to advances in technology, which makes distance education available to most parts of the world. This makes multimedia applications an important part of this process.

Garcia, Freire and Monteiro (2008) stated that, “the IPv4 traffic characteristics are deeply related to the 1500 byte limit, a consequence of the payload capacity of the Ethernet frame.” This meant that with the four times increase in the address space of IPv6, this may give better performance with regards to throughput while reducing the load on the CPU. This study intends to investigate these phenomena. Even though Jumbo Frames is still a new and an emerging technology, Jumbo Frames can be traced back to 1980. There are factors that drive network professionals away from implementing this technology such as, compatibility issues. To implement Jumbo Frames on a network, there is a requirement to change the traditional MTU from 1500 bytes to 9000 Bytes. To make this possible, the Network Interface Card (NIC), all the switches in the network, the routers, and the operating systems all need to support Jumbo Frames. As a result, hardware manufacturers have no choice but to keep increasing the hardware’s speed in order to meet the network users’ and applications’ demands. Chelsio Communications (2010) appropriately explained that,

“This was the case when Gigabit Ethernet was rolled out in the mid-1990s and, today, with the recent deployment of 10 Gbps Ethernet networks... the CPU performance gap today looks particularly severe: even the best performing CPU on the market cannot fill half a 10 Gbps link when using standard frames.”

The need for Jumbo Frames becomes more and more evident, especially with the deployment of a 10 Gbps link and its wide deployment. However, hardware manufacturers have developed more powerful network and computing devices to meet these demands. With Jumbo frames, the defined MTU can be increased to 9000 Bytes which means that more data can be sent through the Inter-network communication channels. There has been research done on this topic, for instance Mindur (2006); Iyer et al (2009); and Yasu et al (2002). This research explored different issues related to network performance when Jumbo frames are used such as is outlined below in Table 2-7 for specifications of computers and the operating systems involved by Yasu et al (2002) in their study.
Table 2-6: Computer specifications and operating systems employed. (Yasu et al, 2002)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Pentium-Xeon 450 MHz with 512KB cache x 2</td>
<td>Pentium2-333MHz with 512KB cache</td>
</tr>
<tr>
<td>Memory</td>
<td>256 MB</td>
<td>160 MB</td>
</tr>
<tr>
<td>Gigabit Ethernet</td>
<td>AceNIC with 1 MB memory</td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>RedHat 5.2 with kernel 2.2.6</td>
<td>RedHat 6.0 with kernel 2.2.5</td>
</tr>
<tr>
<td>C compiler</td>
<td>gcc version 2.7.2.3</td>
<td>gcc version egcs-2.91.66</td>
</tr>
</tbody>
</table>

Following in Figure 2-6 is a module developed to show the relationships between related factors that work together with larger frame sizes to improve the performance of a network.

1. Increasing of the maximum transfer unit will help to minimise data fragmentations which also minimise the processing time of the packets on a network node and as a result, it will improve the network throughput.
2. Regular monitoring and testing of the network will identify the bottlenecks in the network.
3. When the bottleneck is identified, this will allow network professionals to make well informed decisions on how to optimise the overall performance network.
4. Identifying of the right network performance tool to use is a critical task with regards to collecting of the relevant data. In this study, literatures were collected regarding network performance tools and as a result, the proper tool was selected for this study.
5. Network topology referred to the layout of the network and how all the nodes in the network are connected. The performance of the network also depends on the layout or design of the network.

6. With Jumbo frames, selecting the type of network to use will have to be compatible and support Jumbo frames such as the Network Interface card, operating systems and Internet protocol version to employ etc.

2.7 Literature Analysis

Network performance has been an area under discussion by international researchers since the computing domain started. Computer networking has advanced rapidly as developers keep up, with the demands of applications and network users. These demands led hardware manufacturers to develop and manufacture more advanced and faster hardware. Researchers continue to study the performance of networks. This includes evaluating operating systems from different distributions, analysing protocols such as, IPv4 and IPv6, and the transport protocols such as, TCP and UDP. There are a number of articles available regarding this topic, covering this important subject matter.

In all of this literature, researchers in the network performance arena seem to have different views about network performance metrics. However performance metrics are not the only point of disagreement. Also, the method of collecting and measuring the data for analysis is contentious. This is due to the fact that technology is advancing so rapidly. This issue clearly demonstrated in the literature study collected for this study that researchers measure different metrics on various network setups, while running on various operating systems. So far, among the literature collected, two of the performance metrics were tested in most of the articles reviewed. These metrics are network throughput and latency. Network throughput is considered to be the most important metric and the main concern matter for all network professionals.

IP is the backbone within any type of networks. From the literature reviewed it was seen that IP was another issue that occupied the mind of network researchers. This is a result of the fact that the current IP version (IPv4) is running out of addresses. IPv6 is the new version of IP; and its architecture is completely different from its predecessor. There is a variety of performance issues involved with IPv6. Hence, international researchers are implementing the new version of IP in a test environment to study network performance on various operating systems that support the new version of IP.

As a result of the rapid growth of the Internet and networking popularity, the demands for speed are great. As mentioned earlier, hardware manufacturers have no choice but to develop new hardware to meet these demands. Yet, the maximum number that can be transferred over the network is still the same since the birth of networking. A new technology
is emerging known as Jumbo frames. Jumbo frames raised the MTU from 1500 Bytes to 9000 Bytes. This technology will allow more data to send over the network without increasing the workload of the processor. However, since Jumbo frame is still a new technology, there is very limited number of literatures found regarding Jumbo frame. Various areas have been researched, ranging from quality of service to network performance.

There appears to be little research on how a network will perform when Jumbo frame is used. According to the literature reviewed, Microsoft Windows Server 2003 and Ubuntu were the only two operating systems that have been involved. As a result, there are many more network operating systems to study, such as the different versions of Microsoft Windows server operating systems for instance, Server 2008, Server 2008 R2. Linux distributions are released almost every six months. Therefore, this study involves six operating systems namely, Microsoft Windows Server 2008 standard edition, Microsoft Windows Server 2003 Enterprise edition, Microsoft Windows 7 and three Linux based. Linux Fedora, Linux Ubuntu and OpenSUSE.

No literature was found that compares the performance of a network using Jumbo frames under IPv4 and IPv6. The results obtained from this study will help network professionals to make better informed decisions regarding the utilisation of Jumbo frames. According to the literature found, no research compared the performance of a network implementing Jumbo frames to that of a network uses the normal Ethernet frame sizes. Therefore, this study will compare the performance of a network implementing Jumbo frames and normal Ethernet frames. This will be on the operating systems mentioned earlier and using both IPv4 and IPv6.

2.9 Chapter Summary

This chapter reviewed research related to this study that ranged from the Internet protocols (IPv4 and IPv6) to network performance measurement tools. Also included are the necessary network performance metrics and the Jumbo frames discussion required for this research project. After a comprehensive study of the literature found, three major gaps were identified. The first gap was that there were only two operating systems used rather than the number of network operating systems used in today’s networks. The second gap was research about the new version of the Internet protocol (IPv6). The third gap found was the dearth of literature that compared the performance of a network using Jumbo frames under IPv4 and IPv6. Finally, no literature was found that compared the performance of a network using Jumbo frames and a network using normal Ethernet frames. In the next chapter, the research methodology employed for this study is discussed.
3.0 Methodology

This chapter will cover the methodology employed in this study, the data collection method and the hypotheses that this study will answer in the conclusion of this document. Initially when questions arise, there are different ways of finding answers. In this case, research is conducted in order to answer the questions that triggered this study. The word research as defined by Kumar (1998) as, “research is a way of thinking: examining critically the various aspects of your profession; understanding and formulating guiding principles that govern a particular procedure.” Because research is a way of thinking, it needs a method. Method is a logical and orderly course of action for accomplishing the goal. Merriam-Webster (2011) defined method as “a body of methods, rules, and postulates employed by a discipline: a particular procedure or set of procedures.” Although a methodology does not define precise methods, Pattron (2009) explained methodology as, “a highly intellectual human activity used in the investigation of nature and matter and deals specifically with the manner in which data is collected, analysed and interpreted.” These human activities that Pattron referred to are used in an investigation and are comprised of a framework. This framework is used to distinguish one type of research from another.

3.1 Hypotheses

There are a number of aspects of network environment that toil together in order to send packets successfully from source to destination. These will all be involved in this study such as operating systems that currently used in a network environments, protocols used for transporting packets from source to destination and different packet sizes that used in real network environment on both the two internet protocols (IPv4 & IPv6). The main hypothesis of this study is:

“Performance of networks implemented with Jumbo frames gives different performance metrics values depending on transmission protocol, Internet protocol version utilised and the operating system platform employed”

There are also a number of hypotheses that will be tested in this study and these are as follows:

1. IPv6 network using Jumbo frames give different network performance values depending on the distribution of Linux operating system implemented.
2. IPv6 network using Jumbo frames give different network performance values depending if the operating system is that from Microsoft or a Linux distribution.;
3. IPv4 network using Jumbo frames give different network performance values depending on the distribution of Linux operating system implemented on. Linux Ubuntu degrades at a different rates to that of a network using Jumbo frames on Linux Fedora;
4. IPv4 network using Jumbo frames give different network performance values depending if the operating system is that from Microsoft or a Linux distribution.;
5. IPv6 network using Jumbo frames on Microsoft Windows Server operating systems give different network performance to that when implemented on Linux Fedora; and
6. IPv4 network using Jumbo frames on Microsoft Windows Server operating systems give different network performance to that when implemented on Linux Fedora.

As a result of an inductive reasoning which leads to drawing a logical generalisation, the hypotheses listed above were produced based upon observations. This study’s aim is to test the above listed hypotheses via experiments conducted in a controlled computer laboratory. The experimental setup will be discussed in more details in chapter 4.

3.2 Research Methods

An Information Technology researcher will have to decide on the type of research to be conducted. Confirmatory and exploratory researches are the two most common research types. Brown (2006) explained that confirmatory, “deals specifically with measurement models, that is, the relationship between observed measures or indicators.” However, according to Jaeger and Halliday (1998), “confirmatory research proceeds from a series of alternatives, a prior hypothesis concerning some topic of interest, followed by the development of a research design (often experimental) to test those hypotheses.” The previous chapter outlined in detail the pre-defined hypotheses concerning the topic of interest that influence this study.

The second type of research is known as exploratory. DJS Research Ltd (2011) explained the usage of exploratory research as, “exploratory research is used principally to gain a deeper understanding of something.” For instance, every time we learn of a release of a new technology such as the iPad from Apple, it does not mean that we know how the iPad works. This is when exploratory research can be very helpful. As the DJS Research Ltd explained earlier, exploratory research allows researchers to go deeper into an issue or a problem. Figure 3-1 below illustrates how it all connected.

![Figure 3-1: Type of Research, Data Collection & Techniques.](Detmar, Gefen and Boudreau, 2004)
Figure 3-1 shows the relationship between them all, from the type of research used to the general research approach taken, the techniques used for collecting the data and analysing them. Relating the two types of research shown above to this work has shown that there was no dilemma in choosing between confirmatory and exploratory research.

Since the hypotheses for this study has been pre-defined in the previous section, it can be said that this study is a confirmatory research. Two research methodologies that this study employed were Quantitative and Positivist. Detmar, Gefen and Boudreau (2004) stated that, “QPR is a set of methods and techniques that allow IS researchers to answer research questions about the interaction of humans and computers.” This study will be experimenting in order to collect data. That data will be analysed to answer predefined hypotheses regarding the performance of a computer network. Detmar et al agree that Quantitative, Positivist is the most suitable method to apply for such research.

A quantitative approach was employed for this study because it is comprised predominantly of experiments in a controlled laboratory environment. A quantitative research method is numerically based, in the main, which characterises and signifies the values that can be interpreted as scientific evidence to prove hypotheses or phenomena. On the other hand, Qualitative method is comprised of techniques such as action research, case study research and ethnography. According to Detmar et al (2004), “Qualitative research methods were developed in the social sciences to enable researchers to study social and cultural phenomena.” The data collection method employed by Qualitative research stated by Detmar et al (2004) as “observation and participant observation (fieldwork), interviews and questionnaires, documents and texts, and the researcher’s impressions and reactions”. This is why this study did not employ the Qualitative research method because this work’s data collection requires experimenting in a test laboratory.

With regards to the positivist approach, choosing a method to use in any research environment is important. Employing an inappropriate methodology can still yield empirical data but, that data may not have significant value, and could mislead the findings and conclusion for the researcher. Qualitative and quantitative research methods are the two main research methodologies. An interpretivist approach can also have an influence on the outcome of the study. According to Dawson, Bones, Oates, Brereton, Azuma & Jackson (2004), “the outcome of a study depends very much on whether the researcher is a Positivist or an Interpretivist (sometimes called anti-positivist) in their approach.” This means that in positivist philosophy, the researcher will need to take a systematic way of thinking, since researching in Information System and Information Technology arena is not purely scientific. On the other hand, Dawson et al (2004) stated that, “Interpretivists believe all research must be interpreted within the context in which it takes place where even the researcher must be
considered part of the context." However, interpretive philosophy can yield rich data however it depends on the researchers’ own interpretations and values which could impact on understanding any phenomena under study. In this study the Positivist approach has been adopted since this allows the study to adopt a systematic way of proving hypotheses that depend mainly on data collected from experiments conducted in a controlled test environment. The diagram in figure 3-2 on the next page shows the relationship between Qualitative research and Quantitative research with Positivist approach.

Figure 3-2: Epistemological Assumptions for Qualitative & Quantitative Research. (Detmar, Gefen and Boudreau, 2004)

Figure 3-2 above points to illustrate that from the underlying epistemology the threefold approach can be employed, together with either one of the main two research methodology and that is; Positivist, interpretive and critical. As demonstrated in Figure 3-2 above, the positivist approach can be employed by either one of the two research methodology however, the Interpretive and Critical approach can only work together with the qualitative research method. In Table 3-1 below, it shows in details the main differences between the two main research methodologies.

<table>
<thead>
<tr>
<th>Qualitative</th>
<th>Quantitative</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;All research ultimately has a qualitative grounding&quot;</td>
<td>&quot;There's no such thing as qualitative data. Everything is either 1 or 0&quot;</td>
</tr>
<tr>
<td>The aim is a complete, detailed description.</td>
<td>The aim is to classify features, count them, and construct statistical</td>
</tr>
<tr>
<td></td>
<td>models in an attempt to explain what is observed.</td>
</tr>
<tr>
<td>Researcher may only know roughly in advance what he/she is looking for.</td>
<td>Researcher knows clearly in advance what he/she is looking for.</td>
</tr>
<tr>
<td>Recommended during earlier phases of research projects.</td>
<td>Recommended during latter phases of research projects.</td>
</tr>
<tr>
<td>The design emerges as the study unfolds.</td>
<td>All aspects of the study are carefully designed before data is collected.</td>
</tr>
<tr>
<td>Researcher is the data gathering instrument.</td>
<td>Researcher uses tools, such as questionnaires or equipment to collect</td>
</tr>
<tr>
<td></td>
<td>numerical data.</td>
</tr>
<tr>
<td>Data is in the form of words, pictures or objects.</td>
<td>Data is in the form of numbers and statistics.</td>
</tr>
<tr>
<td>Subjective - individuals interpretation of events is important, e.g.,</td>
<td>Objective seeks precise measurement &amp; analysis of target concepts, e.g.,</td>
</tr>
<tr>
<td>uses participant observation, in-depth interviews etc.</td>
<td>uses surveys, questionnaires etc.</td>
</tr>
<tr>
<td>Qualitative data is more ‘rich’, time consuming, and less able to be</td>
<td>Quantitative data is more efficient, able to test hypotheses, but may</td>
</tr>
<tr>
<td>generalized.</td>
<td>miss contextual detail.</td>
</tr>
<tr>
<td>Researcher tends to become subjectively immersed in the subject matter.</td>
<td>Researcher tends to remain objectively separated from the subject matter.</td>
</tr>
</tbody>
</table>

Table 3-1: Features of Qualitative & Quantitative Research (Neill, 2007).
Table 3-1 above outlines in detail the main differences between the two main research methodologies. For instance, Neill (2007) stated that, “qualitative data is more ‘rich’, time consuming, and less able to be generalized. Quantitative data is more efficient, able to test hypotheses, but may miss contextual detail.” The table above provide us with an excellent and clear understanding of the two main research methodologies.

3.3 Methodology Employed for this Study

This study aims to measure the performance of a network that employs and utilises Ethernet packet sizes larger than the normal Ethernet packet sizes that is, larger than 1518 Bytes in size. In this study, the large frame sizes tested were up to 9000 Bytes. This study compared the performance of a network using the normal frame sizes with the performance of the network using the larger frame sizes also known as Jumbo frames.

Due to the nature of this study, the research instrument employed was experimental in a controlled computer laboratory environment. Therefore, as mentioned in the previous section, the chosen type of research for this study was exploratory; the methodology employed for this study was the Quantitative Research Method and alongside that was the Positivist approach.

Jenkins (2009) rightly explained that, “in quantitative research design the researcher will count and classify, and build statistical models to then explain what is observed. Data collected using this research approach is in the form of numbers and statistics.” Results extracted from the data collected during the data collection phase of this study, it will be pure Quantitative data. The product of this study is to prove or to disprove the pre-defined hypotheses shown in the previous section of this chapter. The next section will provide an in depth explanation of the method employed for this study.

3.4 Data Collection Method

Research method and data collection technique are significant parts of a research project. As mentioned earlier, the Qualitative method is very systematic and the data collection instrument adopted for this study requires following certain processes that define the principle of this study. According to Guthrie (2010), “quantitative research primarily focuses on the measurement of objective variables that affect individuals or groups.” As a result, the measurement of variables that affects the performance of a network was done by setting up a test bed in a computer laboratory. In a test bed, dependent variables were introduced to measure the performance of the network. Vanderstoep and Johnston (2008) stated that, “important component of dependent variable measurement is consistency in data gathering.” Throughout the experimental tests, the variables introduced were consistent and constant.
This means that the variables introduced to test the normal Ethernet frame sizes, were introduced also to test larger frame sizes. This was consistent in all aspects of the experiments and for all operating systems.

The information gathering method that was used in this study was principally dependent on two approaches, reviewing of all the literature gathered and the experimental data. All of the literature that was collected were from books, IEEE conference proceedings, Journals and reputable Internet websites. In the first part of the data collection phase, relevant literature was found to support this work. This literature was reviewed in order to gain further understanding about the topic studied. Emerging from this review was an enhanced knowledge of what had been studied in this arena before. Gaps in the literatures were also identified, which then guided the research conducted.

The primary data collection method used in this study was experimental. The main objective was to study the performance of the network that employed Jumbo frames. This experiment was conducted on six different operating systems namely; Microsoft Windows Server 2003 standard edition, Microsoft Windows Server 2008 standard edition, Microsoft Windows 7 professional, Linux Fedora Core, Linux Ubuntu and OpenSUSE. In order to gain better understanding, the normal Ethernet frame sizes were tested with the same dependent variables. However, included in this test were the two transmission protocols TCP and UDP, and the two main Internet Protocols IPv4 and IPv6. The results extracted from the data collected in these experiments helped in drawing conclusions for this study to prove whether the pre-defined hypotheses were true or false.

The second part of the data collection phase was conducted in a controlled computer laboratory environment. The data in this stage will be collected by using the performance measurement tool that was selected in the beginning (D-iTG). ITGSend was used to generate the required traffic. Also at the same time ITGRecv was running on the receiving side, ITGRecv was used to collect and store the data. At the end of the experiment, ITGDec was used to decode the data and its output was in text format. After decoding the data, the data was then entered into Microsoft Excel and plotted into graphs. These graphs were then used as the main source for the data analysis phase. Figure 3-3 below is a sample of the scripts used to generate the traffics required for the tests.

![Figure 3-3: Script used by ITGSend to generate traffics during the experiments.](image-url)
The above sample in Figure 3-3 above is a sample of script of commands used by ITGSend to generate the necessary traffic for the required test. Figure 3-4 below is a sample of total results after decrypted by ITGDec.

<table>
<thead>
<tr>
<th>TOTAL RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Flows = 10</td>
</tr>
<tr>
<td>Total time = 10.01500 s</td>
</tr>
<tr>
<td>Total packets = 1071568</td>
</tr>
<tr>
<td>Minimum delay = 1.34600 s</td>
</tr>
<tr>
<td>Maximum delay = 2.709900 s</td>
</tr>
<tr>
<td>Average delay = 1.813848 s</td>
</tr>
<tr>
<td>Standard deviation = 0.088717 s</td>
</tr>
<tr>
<td>Bytes received = 97934032</td>
</tr>
<tr>
<td>Average bitrate = 43423.513462 kbit/s</td>
</tr>
<tr>
<td>Average packet rate = 35773.573442 pkt/s</td>
</tr>
<tr>
<td>Packets dropped = 0 (0.00 %)</td>
</tr>
<tr>
<td>Error Times = 0</td>
</tr>
</tbody>
</table>

In Figure 3-4 above, a result of a data decoded by the ITGDec is shown. Notice in the sample above, the total results of the decoded data are shown. The number of flows, the minimum delay, the maximum delay, and the average delay are all calculated and shown. Figure 3-4 also shows the number of Bytes received during the test. The Average bitrate highlighted the throughput in Kilobits per second. This amount is then converted into Megabits before being carried over to the Microsoft Excel to be graphed. The reason for this is that the amount of data to be shown in the graph is then minimised.

Following in Figure 3-5 below is a sample of line graph created in Microsoft Excel.

The line graph in Figure 3-5 above demonstrates the result of what can be achieved when the data is entered into Microsoft Excel and plotted into line graphs. From the line graph showing in Figure 3-5 above, the Y – axis shows the normal Ethernet frame sizes that were tested. On the X – axis it shows the throughput in Megabits per seconds as a result of this sample tests.
3.5 Chapter Summary

In this chapter, the elements of research methods including hypotheses, research methodology employed and data gathering methods adopted by this study have been presented. The pre-defined hypotheses depict the boundary or the scope of this research, while the methodologies employed defined the type of vehicle used to carry this study from beginning to end. It was previously decided that a quantitative research method would be used for this study. A quantitative method is also a positivist approach, since positivism will add and help this study in taking a systematic way of thinking. Therefore, primary data for this research was gathered experimentally. When data gathering was completed, the data was entered into Microsoft Excel for the analysis phase.
4.0 Experimental Setup

The data collection instrument employed by this study, as explained in section 3.4 earlier, is experimenting in a controlled computer laboratory. The primary focus of this chapter is to explain in detail the equipment employed and the entire experimental setup. As it was mentioned in section 3.3 earlier, the predominantly focus of this study is to measure and compare the performance of a network that employed the Jumbo frames to that of a network that used the normal Ethernet frames. Six operating systems involved in this study namely Microsoft Windows Server 2008, Microsoft Windows Server 2003, Microsoft Windows 7 Professional, Linux Fedora Core, Linux Ubuntu and OpenSUSE.

4.1 Hardware Specifications

In order to be consistent and produce accurate data from this study, all of the hardware used in all of the experiments was kept identical. Following in Table 4-1 below outlines the type and specifications of the hardware involved.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Intel® Core 2 Duo CPU E6300 @ 1.86GHz</td>
</tr>
<tr>
<td></td>
<td>Core Speed – 1596.0 MHz</td>
</tr>
<tr>
<td></td>
<td>Bus Speed – 266.0 MHz</td>
</tr>
<tr>
<td></td>
<td>Rated FSB – 1064.0 MHz</td>
</tr>
<tr>
<td>Motherboard</td>
<td>Lenovo</td>
</tr>
<tr>
<td>Motherboard Chipset</td>
<td>Intel Q965 Rev. 1</td>
</tr>
<tr>
<td></td>
<td>Southbridge – Intel 82801HB/HR (ICH8/R)</td>
</tr>
<tr>
<td>BIOS</td>
<td>Lenovo 2JKT40AUS</td>
</tr>
<tr>
<td>Cache</td>
<td>L1 Data 2 x 32 KB 8 - way</td>
</tr>
<tr>
<td></td>
<td>L1 Inst 2 x 32 KB 8 - way</td>
</tr>
<tr>
<td></td>
<td>L2 2048 KB 8 – way</td>
</tr>
<tr>
<td>Memory</td>
<td>1GB x 2 Samsung M3 DDR2 PC2-5300 (333MHZ)</td>
</tr>
<tr>
<td>2 x PCI Network Interface Card</td>
<td>Intel® PRO/1000 GT Desktop Adapter</td>
</tr>
</tbody>
</table>

Table 4-1: Hardware specifications

In addition to the hardware shown in Table 4-1 above, Cat5e crossover cable was used to connect all the computers together. The two Network Interface Adapters were chosen because they support Jumbo frames.
4.2 Software Specifications

As mentioned in section 3.4, six operating systems were involved in this study. Three operating systems were Microsoft products and the other three were Linux distributions. Apart from the operating systems, the tool used to generate the traffics for the experiments, is a software tool known as Distributed Internet Traffic Generator (D-ITG). Following is a list of all software involved in this study.

Operating Systems:
- Microsoft Windows 7 Professional
- Microsoft Windows Server 2003 Enterprise Edition

Linux Distributions:
- Linux Fedora Core 12
- Linux Ubuntu 10.10
- OpenSUSE 11.3

Software Tool
- Distributed Internet Traffic Generator (D-ITG) v2.6.1

The D-ITG platform came with five components, three of these components were used in this study and these components are as follows:
- ITGSend
- ITGRecv
- ITGDec

The ITGSend component of the software tool was used for generating the required traffic and for data sent from source to destination. The ITGRecv was used to receive and store the data collected during the experiments. The last component was the ITGDec that was used to decode the receiving data. This tool also analysed and calculated the average bit rate for the relevant metrics.

4.3 Network Design

The design of the test-bed for this study involved four computers. One computer was the sender, the two computers in the middle were configured as routers and the last computer was the receiver. The infrastructure was designed to simulate a wide area network. The simulation consisted of two private networks connected by two routers representing public network. Figure 4-1 on the next page exemplifies the infrastructure design in more detail.
As shown on Figure 4-1 above, the four computers were connected by three Cat5e cross-over cables, and this link sent a Gigabit of data from sender to receiver. The Figure 4-1 above also shows the three different network setups. The two internal networks were connected via an external network. Also, this diagram shows the IPv4 addresses used and the IPv6 addresses. The D--ITG sender and receiver were configured with the Microsoft Windows 7 Professional Edition. This configuration was kept constant throughout the whole experimental phase. The experiments were performed on the six operating systems by configuring one platform on the two routers at a time.

4.4 Determining the Optimal Value

At the initial stage of data collection, prior to commencing the actual experiment, it was necessary to determine the optimal value that would be used throughout the entire data gathering phase. This was considered to be a significant ingredient in terms of maintaining the consistency of the results of this study. As a result, in order to establish the integrity of this study’s findings and experimental results, several experiments were conducted in order to determine the optimal value for variables that could be fixed for all the experiments. Variables involved in these initial experiments were: determining the duration of one run; the maximum of packets to send every second; and also determining the maximum number of flows that could be generated in one run. In figure 4-2 on the next page, the line graphs show the results of these test runs.
The initial test runs were conducted on Microsoft Windows Server 2008. There were no particular reasons for this selection. There have been a number of research articles found on Normal Ethernet frame sizes therefore, the researcher decided to conduct these test runs on Jumbo frame sizes. In the initial stage of these experiments, the MTU allowed by the chosen Network Interface Card (NIC) was 16120 Bytes.

All of the chosen frame sizes were not tested; instead several frame sizes were selected randomly. Figure 4-2 above shows the result of test conducted on Microsoft Windows Server 2008. TCP was the transmission protocol used on IPv6. In Figure 4-3 above, are the results for the test conducted on Jumbo frame size 6448 Bytes. Figure 4-4 above shows that throughput was still going up, as the number of packets per second increased and the duration of the experiment increased as well.
According to the line graphs shown in Figure 4-2, Figure 4-3 and Figure 4-4 on the previous page, and in Figure 4-5 and Figure 4-6 above, it can be said that the amount of throughput is still growing. This is showing for all of the variables that were tested such as, the duration of the tests and the packets that were sent per second. For instance, between Figure 4-2 and Figure 4-3, in particular the test ran for 60 seconds, 180,000 Bytes of packets were sent every second, and an increase of 14.55% in terms of throughput was seen. Also, between packet sizes 6448 Bytes and 9672 Bytes, there was an increase of 6.35%. From packet size 12896 to packet size 16120 Bytes there was an increase of 1.83%. Even though a drop with regards to the number of throughputs was shown, there was still an increase. Figures below illustrate the differences in throughput as the number of packets per seconds’ increased and also the duration of the tests.

Figure 4-5: 12896 Bytes results.

Figure 4-6: 16120 Bytes results.

Figure 4-7 and Figure 4-8 above show the differences between selected packet sizes for these test runs. For instance; there was an increase of 14.5% in terms of throughput between packet size 3224 and 6448 Bytes, 180000 packets per second was sent and the test was run for the duration of 60 seconds. For the same duration and same amount of packets per seconds, throughput was dropped from 14.5% to only 6% increase. On the next page Figures 4.9 and Figure 4.10 show the difference for the rest of the packet sizes selected for these test runs.
When analysing the same variables mentioned earlier for Figure 4-7 and Figure 4-8, when applying them to Figure 4-9 and Figure 4-10, there was a significant change, which was that there was a drop of 6% between packet sizes 6448 – 9672 Bytes to only 1.5% between 9672 – 12896 Bytes. However, the most obvious drop with regards to the amount of throughput shows between packet sizes ranging from 12896 to 16120 Bytes. There was a drop of 1.8% lower than the throughput amount of packet size 12896 Bytes and this is shown as negative. Following in the next page, Figure 4-11 presents a line graph illustrating the figures that were the basis for optimal value in this study.
According to the entire graph illustrations provided in this section, it can be said that there was no major difference between the variables tested. According to the line graphs showing in Figure 4-11, Figure 4-12 and Figure 4-13, for all of the packets that were sent per seconds, most of the time, throughputs stay above 400 Mbps for all of the packet sizes. Following in Figure 4-14 and Figure 4-15 are the results extracted from experiments conducted on the same variables but where UDP was the transport protocol employed.

**Figure 4-14: UDP Throughputs on 3224 Bytes Packet size.**

**Figure 4-15: UDP Throughputs on 6448 Bytes Packet size.**

**Figure 4-16: UDP Throughputs on 9672 Bytes packet size.**

**Figure 4-17: UDP Throughputs on 12896 Bytes Packet size.**

**Figure 4-18: UDP throughput on 16120 Bytes Packet size.**
According to the figures above, it can be said that when employing UDP as the transmission protocol, throughput is not as high as TCP. The reason for this will be discussed in the next chapter (data analysis).

### 4.5 Optimal Values for this Study

After running the initial experiments and analysing the results, it can be said that the best combinations to employ for this study would be the following:

<table>
<thead>
<tr>
<th>Duration for one run</th>
<th>60 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packets per second</td>
<td>180000</td>
</tr>
<tr>
<td>Number of flows per run</td>
<td>60 flows</td>
</tr>
</tbody>
</table>

These combinations were chosen based on the results of the initial experiments conducted. The 60 seconds duration on 180000 packets per second demonstrated a very stable behaviour on figure 4-12 in section 4.4 above. The initial experiments were conducted with only 10 flows per run as is shown in Figure 3-3 in Chapter 3. However, more flows were introduced to the runs especially to the chosen combination as is showed in Figure 3-4 in chapter 3; it is evident that there was a difference with regards to throughput results.

Literature reviewed for this study on Jumbo frames, included research on up to 9000 MTU such as Hsiao, Chen, Huang and Yeh (2009); Maccabe, Zhu, Otto and Riesen (2002) and Trisnawan, Jiang, Hooi, and Budiarto (2010). Further research was conducted to find out why the MTU was limited to only 9000 Bytes instead of 16120 Bytes. As explained earlier in section 2.6, MTU of 16120 is allowed by the Intel® PRO/ 1000 GT Desktop Adapter. However, there are some issues to consider before defining such an MTU. For instance; according to Chelsio Communications (2010), “Ethernet uses a 32 bit CRC that loses its effectiveness above about 12000 bytes.” This can affect the results of this study; CRC was introduced back in 1961 and is still used in today’s computing and networking technologies. However, it is expected that network performance in a network using Jumbo frames should be better, due to the fact that with higher MTU, packet fragmentation will be less and also there is less overhead. As a result, the decision was made that the most ideal MTU size to be used in this study would be 9000 Bytes.
4.6 Chapter Summary

This chapter covered the premise of the experimental setup for this study, starting with discussing the specifications of the hardware employed. As is mentioned in this chapter, four computers in total were involved, with identical hardware. Also included in this chapter, was a discussion of the specifications of all software involved. All of the operating systems used are also described, as well as the software tool employed to generate the necessary traffic. A discussion of the network setup was also included in this chapter; a detailed diagram of the design has been included also.

This chapter ends with a discussion of the process taken by this study to determine the optimal value to use. This optimal value refers to the value of the variables that were fixed during the experiments. In the next chapter, an analysis of the results extracted from the data gathered during the experiments is presented.
Chapter 5: Data Analysis

In this chapter the analysis of the data obtained as a result of the experiments is presented. This chapter is necessary to understand the network performance of the two Ethernet frame sizes discussed in this study that is, the normal Ethernet frame sizes and the Jumbo frame sizes. The results have been plotted into line graphs.

5.1 TCP Analysis

This section is concerned with the analysis of all TCP results from the experiments conducted on Jumbo frames. Results in discussions are also based on the chosen metric discussed in chapter 2, section 2.1.1, which are throughput, delay and jitter. The CPU utilisations were also measured in the experiments, so these will also be included in the analysis.

5.1.1 TCP Throughputs on IPv6 & IPv4 Jumbo Frames

This section presents the line graphs of the throughput results for the experiments conducted on a network employing Jumbo frames on IPv6 and using TCP as the transmission protocol. Figure 5-1 below presents the Jumbo frames throughput results of experiments conducted on various operating systems implemented on a network employing IPv6 and use TCP as the transmission protocol.

![IPv6 - TCP Throughput on Various OS](image)

Figure 5-1-1: TCP IPv6 throughput on Jumbo frames on various operating systems.

The following list is the analysis of the throughput results for all of the operating systems on TCP IPv6 presented in Figure 5-1-1 above:
- On the smallest Jumbo frame size, 1518 Bytes, throughput value is very similar
- On Jumbo Frame size 2455 Bytes, Microsoft Windows 7 outperformed the rest of the operating systems involved by 4%.
- The performance of all operating systems on Jumbo frame sizes ranging from 3992 Bytes to 6203 Bytes produces similar throughput values.
- On Jumbo frame size 4329 Bytes, throughput value for all of the operating systems dropped by around 85%.
- On Jumbo frame size 7140 Bytes Microsoft Windows Server 2008 produces the highest in terms of throughput values outperforming Linux Fedora by 5% while Fedora tops Ubuntu by 4% and OpenSUSE produces the lowest.
- On the two largest frame sizes, all of the operating systems produce very similar throughput values.
- According to Figure 5-1-1, all of the operating systems produced similar performance and degraded in a similar way.

Figure 5-1-2 below presents the Jumbo frames throughput results of experiments conducted on various operating systems implemented on a network employing IPv4 and using TCP as the transmission protocol.

Figure 5-1-2: TCP IPv4 throughput on Jumbo Frames on various operating systems.

The following list is the analysis of the throughput results for all of the operating systems on TCP IPv4 presented in Figure 5-1-2 above:
- For all of the six operating systems involved, on the three smallest Jumbo frame sizes, ranging from frame size 1518 Bytes to 3392 Bytes, throughput values are very similar except for Microsoft Windows Server 2003 which produced 9% lower than the other operating systems involved.
- On Jumbo frame size 4329 Bytes, throughput value for all of the operating systems dropped by around 84%.
- On Jumbo Frame size 5266 Bytes, all of the operating systems involved in this study produce the same output in terms of throughput values.
- For the last four frame sizes ranging from frame size 6203 Bytes to 9014 Bytes, all of the operating systems produce very different throughput values. On frames size 6203 Bytes OpenSUSE outperformed the rest of the operating systems, 8% more than Microsoft Windows Server 2003 which produced the lowest throughput value.
- For the frame size 7140 Bytes, Fedora produced the highest in terms of throughput values, 10% more than Microsoft Windows Server 2008, which came second and 24% more than Ubuntu which produced the lowest throughput values.
- According to Figure 5-1-2, excluding frame size 4329 Bytes, frame sizes ranging from 1518 to 6203 Bytes, Microsoft Windows Server 2003 produced 8% lower throughput values than the rest of the operating systems.

### 5.1.2 Compare TCP Throughputs on IPv4 & IPv6 Jumbo Frames

This section presents in line graphs a comparison of IPv4 and IPv6 throughput results on all of the six operating systems involved using Jumbo frames while TCP was the transmission protocol.

![Compare Jumbo Frames TCP Throughput on Various OS on IPv6 & IPv4](image)

The following list compares and analyses the throughput results for all of the operating systems on TCP IPv6 and IPv4 presented in Figure 5-1-3 above:

- Jumbo frame size 4329 produced the lowest in terms of throughput values on all of the operating systems and on both of the Internet protocol versions.
- Microsoft Windows Server 2008, IPv4 outperformed IPv6 by 2% on all of the frame sizes except for frame size 6203 Bytes where IPv6 produced almost 3% more than IPv4 in terms of throughput values.
- Microsoft Windows 7 on some particular frame sizes such as 1518, 7140 and 9014 Bytes, IPv4 outperformed IPv6 by up to 5% in terms of throughput values. For the rest of the Jumbo frame sizes, IPv6 produced more throughputs of up to 6% on frame size 6203 Bytes.
- Microsoft Windows Server 2003, IPv6 outperformed IPv4 on all of the Jumbo frame sizes except for frame sizes 7140 and 8037 Bytes where IPv4 produced more throughput values than IPv6.
- All of the Linux distributions showed a very similar performance for all of the Jumbo frame sizes and the two Internet protocols for frame size ranging from 1518 Bytes to 6203 Bytes.
- For frame size 7140 Bytes in particular, Fedora produced 16% more throughputs on IPv4 than IPv6.
- Jumbo frame size 8037 Bytes in particular, on all of the Linux distribution operating systems involved, IPv6 outperformed IPv4.
- For the largest frame size tested in this study, IPv4 outperformed IPv6 on Ubuntu and OpenSUSE where IPv4 on Fedora outperformed IPv6.
- Overall, Microsoft Windows Server 2008, Microsoft Windows 7 and OpenSUSE produced very similar results and the highest results with regards to throughput values on most of the frame sizes. Therefore, it can be said that a network employing Internet protocol version 4 outperformed a network that employed Internet protocol version 6 while using TCP as transmission protocol. Moreover, Figure 5-1-3 concluded that excluding frame size 4329 Bytes, frame sizes ranging from 1518 to 6203 Bytes, Microsoft Windows Server 2003 on IPv4 produced 8% lower throughput values than the rest of the operating systems.

5.1.3 TCP One Way Delay on IPv6& IPv4 Jumbo Frame

Figure 5-1-4 below presents the Jumbo frames throughput results of experiments conducted on various operating systems implemented on a network employing IPv6 and using TCP as the transmission protocol.

![Figure 5-1-4: TCP IPv6 delay on Jumbo frames on various operating systems.](image-url)
The following list is the analysis of the OWD results for all of the operating systems on TCP IPv6 presented in Figure 5-1-4 above:

- Microsoft Windows Server 2008 and 2003 produced the same values in terms of delay on frame sizes ranging from 1518 to 3392 Bytes.
- Microsoft Windows Server 2008 produced the highest values in terms of delay on the larger frame sizes except on frame size 5266 Bytes where Microsoft Windows 7 produces 9% more delays than Microsoft Windows Server 2008.
- Microsoft Windows 7 and Linux Fedora produced the same amounts of delay on small frame sizes ranging from 1518 Bytes to 4329 Bytes, also on frame sizes 6203 and 7140 Bytes.
- Figure 5-1-4 concluded that OpenSUSE produced 41.5% which was the lowest amount of delays when compared to Microsoft Windows Server 2008 which produced the highest delay values.

Figure 5-1-5 below presents the Jumbo frames throughput results of experiments conducted on various operating systems implemented on a network employing IPv4 and using TCP as the transmission protocol.

The following list is the analysis of the OWD results for all of the operating systems on TCP IPv4 presented in Figure 5-1-5 above:

- Microsoft Windows Server 2008 produced the highest amounts in terms of delay on all of the frame sizes except of frame sizes 2455 Bytes and 5266 Bytes where it produced the same amount of delays with Microsoft Windows 7.
- Microsoft Windows Server 2008 produced .290ms more delay than Microsoft Windows Server 2003 which produces the second highest quantity of delays.
- Linux Fedora and Ubuntu produced the same amount of delays on the two smallest frame sizes ranging from 1518 Bytes to 2455 Bytes.
Figure 5-1-6 concluded that Ubuntu produced 80.3% which was the lowest amount of delays on all of the larger frame sizes ranging from 3392 Bytes to 9014 Bytes when compared to Microsoft Windows Server 2008 which produced the highest delay values.

The next section presents a line graph representations of a comparison of the amount of delays of TCP Jumbo frames on IPv6 and IPv4 which presented in Figure 5-1-4 and Figure 5-1-5 above.

**5.1.4 Compare TCP Delays on IPv4 & IPv6 Jumbo Frames**

Figure 5-1-6 below presents a comparison of the Jumbo frames one way delay results on experiments conducted on various operating systems implemented on a network employing IPv4 and IPv6 and also using TCP as the transmission protocol.

The following list compares and analyses the OWD results for all of the operating systems on TCP IPv6 and IPv4 presented in Figure 5-1-6 above:

- For all of the Microsoft Windows operating systems, IPv4 produced more amounts of delays than IPv6 for all of the Jumbo frame sizes.
- With regards to the Linux distributions, IPv6 on Fedora and Ubuntu generated more delays than IPv4.
- IPv6 on OpenSUSE created less delay than all of the other operating systems involved while Ubuntu on IPv4 generates the lowest amounts of delays.
- IPv4 on Microsoft Windows Server 2008 yielded the highest delay while Microsoft Windows Server 2008 and 2003 yielded similar amount of delays.
- Therefore, it can be concluded that IPv6 created less delay than IPv4 on Jumbo frames.
5.1.5 TCP Jitter on IPv6 & IPv4 Jumbo Frame

Figure 5-1-7 below presents the Jumbo frames jitter results of experiments conducted on various operating systems implemented on a network employing IPv6 and using TCP as the transmission protocol.

The following list is the analysis of the results for jitter for all of the operating systems on TCP IPv6 presented in Figure 5-1-7 above:

- For most of the Jumbo frame sizes tested, very similar jitter value on all of the operating systems except for frame sizes 4329, 6205 and 7140 Bytes was produced
- On frame size 4329 Bytes in particular, all of the operating systems showed similar behaviour in terms of jitter, which was not expected
- On the highest frame size, Microsoft Windows Server 2003 produced a higher amount of jitter while the rest of the operating systems produced similar amount of jitter.

The following list is the analysis of the results for jitter for all of the operating systems on TCP IPv4 presented in Figure 5-1-8 above:

- The three lowest frame sizes ranging from 1518 to 3392 Bytes produced very similar jitter values on all of the operating systems.
On frame size 4329 Bytes in particular, all of the operating systems showed similar behaviour in terms of jitter which was not expected.

On frame sizes ranging from 6203 to 9014 Bytes, Microsoft Windows Server 2003 produced the lowest amount of jitter.

On frame size 4329 Bytes and 6203 to 7140 Bytes, Linux Ubuntu generated the highest amount of jitter.

On the largest frame size 9014 Bytes, Microsoft Windows 7 generated the highest jitter values while the rest of the operating systems yielded similar amounts of jitter.

The next section presents a line graph representations of a comparison of the amount of jitter on TCP Jumbo frames on IPv6 and IPv4 which was presented in Figure 5-1-7 and Figure 5-1-8 above.

5.1.6 Compare TCP Jitter on IPv4 & IPv6 Jumbo Frames

Figure 5-1-9: Comparison of IPv4 & IPv6 Jitter results on Jumbo frames.

The following list Compares and analyses the results for jitter for all of the operating systems on both IPv6 and IPv4 presented in Figure 5-1-9:

- For frame sizes ranging from 1518 to 3392 Bytes and 5266 Bytes, all of the operating systems involved in this study generated very similar jitter values.
- On frame size 4329 Bytes in particular, all of the operating systems on both IPv6 and IPv4 behaved similarly and produce an out of the ordinary large amount of jitter.
- Frame sizes ranging from 6203 Bytes to 8037 Bytes, all of the operating systems gradually produced more jitter on both IPv6 and IPv4 as the frame size increases.
- According to Figure 5-1-9 above, it can be said that IPv6 on all of the operating systems generated more jitter than IPv4.
5.1.7 TCP CPU Utilisations for Router 1 on IPv6 & IPv4 Jumbo Frame

Figure 5-1-10 below presents the Jumbo frames IPv6 CPU utilisation results of Router 1 from the experiments conducted on various operating systems using TCP as the transmission protocol.

The following list is the analysis of the CPU utilisations results for IPv6 on Router 1 for all of the operating systems presented in Figure 5-1-10 above:

- OpenSUSE used more of the CPU resources than any of the operating systems involved in this study on all of the frame sizes except for 7140 and 8037 Bytes where Ubuntu used the most.

- On Jumbo frame sizes 3392 and 9014 Bytes, OpenSUSE showed unanticipated high usages of the CPU resources. For instance, OpenSUSE used 8% more of the CPU resources than Ubuntu which was the second highest in those particular frame sizes.

- All of the Microsoft operating systems behaved and showed very similar pattern throughout for all of the Jumbo frame sizes except, on frame size 2455 Bytes where Microsoft Windows 7 used the least CPU resources.

- Linux Ubuntu showed a very constant performance for all of the frame sizes except for frame sizes 7140 and 8037 Bytes where Ubuntu used up to 2% more of the CPU time.

Figure 5-1-11 presents the Jumbo frames IPv4 CPU utilisation results of Router 1 from the experiments conducted on various operating systems using TCP as the transmission protocol.
The following list is the analysis of the CPU utilisation results for IPv4 on Router 1 for all of the operating systems presented in Figure 5-1-11:

- For IPv4, OpenSUSE performed unsteadily and used high percentage of the CPU resources on most of the frame sizes except for frame sizes 4329, 5266 and 6203 Bytes where Linux Ubuntu was the highest in terms of CPU utilisation.
- All of the Microsoft operating systems demonstrated a very similar pattern throughout for most of the frame sizes apart from frame sizes ranging from 7140 to 9014 Bytes where Microsoft Windows Server 2003 used more CPU resources than Microsoft Windows Server 2008 and Microsoft Windows 7.
- Frame sizes ranging from 5266 to 9014 Bytes, Microsoft Windows Server 2008 used the least of the CPU resources.

The next section presents a line graph representations of a comparison of the amount of the CPU utilisation measured on Router 1 which was presented in Figure 5-1-10 and Figure 5-1-11 above.
5.1.8 Comparing TCP CPU Utilisations for Router 1 on IPv4 & IPv6 Jumbo Frames.

The following list compares and analyses the CPU utilisation results for all of the operating systems on both IPv6 and IPv4 on Router 1 presented in Figure 5-1-12:

- For all of the Linux distributions on the large frame sizes ranging from 5266 to 9014 Bytes, IPv6 employed more CPU resources than IPv4 except on frame size 6203 Bytes where OpenSUSE IPv4 used more CPU resources than IPv6.
- Microsoft Windows Server 2008 and 2003 performed similarly for the two small frame sizes.
- On frame sizes 3392, 5266 and 6203 Bytes, IPv6 used more CPU resources than IPv4 on Server 2003 and on the larger frame sizes ranging from 7140 to 9014 Bytes IPv4 used more CPU resources than IPv6.
- IPv6 on Microsoft Windows Server 2008 IPv6 used more of the CPU resources than IPv4 on all of the Jumbo frame sizes.
- For all of the larger frame sizes ranging from 4329 to 9014 Bytes, IPv6 used more of the CPU resources than IPv4.
- On frame sizes 2455, 4329, 5266 and 7140 Bytes, IPv6 and IPv4 on OpenSUSE demonstrated very similar percentages in terms of CPU utilisation.
- The three Linux distributions on both IPv6 and IPv4 utilised more of the CPU resources than all of the Microsoft operating systems.
- OpenSUSE on both IPv6 and IPv4 utilised the CPU resources the most on most of the frame sizes except for frame sizes 7140 and 8037 Bytes where Ubuntu used the most.
- Based on the graph in Figure 5-1-12, it can be said that IPv6 on most of the operating systems for most of the frame sizes required more time of the CPU than IPv4.
- Based on the graph in Figure 5-1-12, it can be said that Fedora IPv4 on frame size 3392 Bytes in particular, where it required more of the CPU time than IPv6.
Based on the graph in Figure 5-1-12, it can be said that Microsoft Windows 7 IPv4 on frame sizes ranging from 2455 to 6203 Bytes and also 8037 Bytes needed more of the CPU time than IPv6.

According to Figure 5-1-12, it can be concluded that IPv6 utilised more of the CPU time that IPv4 on Router 1.

5.1.9 TCP CPU Utilisations for Router 2 on IPv6 & IPv4 Jumbo Frame

Figure 5-1-13 below presents the Jumbo frames IPv6 CPU utilisation results measured on Router 2 during the experiments conducted on various operating systems using TCP as the transmission protocol.

The following list is the analysis of the CPU utilisation results for IPv6 Jumbo frames on Router 2 presented in Figure 5-1-13 line graph above:

- For IPv6, OpenSUSE performed unsteadily and used a high percentage of the CPU resources on most of the frame sizes except for frame sizes 2455 and 7140 Bytes where Linux Ubuntu was the highest in terms of CPU utilisations by 0.5%.
- All of the Microsoft operating systems demonstrated a very similar pattern throughout for most of the frame sizes apart from frame size 2455 Bytes where Microsoft Windows 7 used less CPU resources than Microsoft Windows Server 2008 and Microsoft Windows 2003.
- The three Linux distributions used more of the CPU resources for all of the Jumbo frame sizes involved in this study.
- Among the three Linux distributions, Fedora Core used less CPU resources while OpenSUSE being the highest in terms of CPU utilisations on most of the Jumbo frame sizes.
Figure 5-1-14 below presents the Jumbo frames IPv4 CPU utilisation results measured on Router 2 during the experiments conducted on various operating systems using TCP as the transmission protocol.

The following list is the analysis of the CPU utilisation results for IPv4 Jumbo frames on Router 2 presented in Figure 5-1-14 line graph above:

- For IPv4, OpenSUSE performed unsteadily and used a very high percentage of the CPU resources on two particular frame sizes 3392 and 8037 Bytes. There was 4% more usage on frame size 3392 Bytes and 1.5% more on frame size 8037 Bytes than Ubuntu, which is the second highest in terms of CPU utilisation.

- All of the Microsoft operating systems demonstrated a very similar pattern throughout for most of the frame sizes apart from frame sizes ranging from 7140 to 8037 Bytes where Microsoft Windows Server 2003 used more CPU resources than Linux Fedora which was next.

- Linux Fedora demonstrated a very steady pattern in its CPU utilisation activities where it dwelled around 1.4% and 1.7% for all of the frame sizes.

- According to Figure 5-1-14, Linux Ubuntu utilised 31.5% more of the CPU time on Router 2 on most of the frame sizes.

The next section presents a line graph representation of a comparison of the amount of the CPU utilisation measured on Router 2 which was presented in Figure 5-1-13 and Figure 5-1-14 above.
5.1.10 Comparing TCP CPU Utilisations for Router 2 on IPv4 & IPv6 Jumbo Frames.

The following list compares and analyses the CPU utilisations results for all of the operating systems on both IPv6 and IPv4 on Router 2 presented in Figure 5-1-15:

- All of the Linux distributions on both IPv6 and IPv4 used more CPU resources than all of the Microsoft Windows operating systems on most of the frame sizes except for frame sizes 7140 and 8037 Bytes where Microsoft Windows server 2003 IPv4 employed more CPU resources.
- IPv6 on all of the Microsoft Windows operating systems employed more CPU time on most of the frame sizes apart from the large frame sizes ranging from 7140 to 9014 Bytes where IPv4 on Microsoft Windows Server 2003 required more of the CPU time.
- IPv6 on OpenSUSE required more of the CPU time on all of the frame sizes excluding frame sizes 3392 and 9037 Bytes where IPv4 required 3% more of the CPU time than IPv6.
- IPv6 on Microsoft Windows Server 2008 used more of the CPU resources on all of the frame sizes apart from frame size 8037 Bytes where IPv4 used 0.5% more than IPv6.
- IPv6 on Fedora Core used more of the CPU time on all of frame sizes excluding the two smallest frame sizes 1518 and 2455 Bytes where IPv4 required more of the CPU time than IPv6.
- IPv6 on Microsoft Windows 7 required more of the CPU time on all of the frame sizes excluding frame sizes 2455 and 4329 Bytes where IPv4 required 1% more of the CPU time.
- Six small frame sizes ranging from 1518 to 6203 Bytes on Linux Ubuntu apart from 3392 and 5266 Bytes, IPv6 used more CPU resources while IPv4 required more of the CPU time on the large frame sizes ranging from 7140 to 9014 Bytes.
Based on the graph showing in Figure 5-1-15, it can be said that IPv6 on Microsoft Windows Server 2008 for most of frame sizes required more of the CPU time than IPv4 apart from frame size 8037 Bytes where IPv4 utilised the CPU resources more.

Based on the graph showing in Figure 5-1-15, it can be said that IPv6 on frame sizes 1518 and 3392 Bytes and also larger frame sizes ranging from 5266 to 9014 Bytes on Microsoft Windows 7 utilised the CPU resources than IPv4. On frame sizes 2455 and 4329 Bytes, IPv4 on Microsoft Windows 7 required more of the CPU resources the IPv6.

Based on the graph showing in Figure 5-1-15, it can be said that IPv4 on Microsoft Windows Server 2003 on the smallest frame size and also on the larger frame sizes ranging from 7140 to 9014 Bytes utilised more of the CPU resources than IPv6. On frame sizes 2455, 3392, 5266 and 6203 Bytes, IPv6 on Microsoft Windows Server 2003 required more of the CPU time than IPv4.

Based on the graph showing in Figure 5-1-15, it can be said that IPv6 on Fedora Core on all of the frame sizes consumed CPU resources more than IPv4.

Based on the graph showing in Figure 5-1-15, it can be said that IPv4 on Linux Ubuntu on frame sizes 3392, 5266 Bytes and larger frame sizes ranging from 7140 to 9014 Bytes consumed more of the CPU time than IPv4. On smaller frame sizes 1518, 2455 and 4329, and 6203 Bytes where IPv6 consumed more of the CPU resources than IPv4.

Based on the graph showing in Figure 5-1-15, it can be said that IPv6 on OpenSUSE for most of the frame sizes utilised more of the CPU time than IPv4 apart from frame sizes 3392 and 8037 Bytes where IPv4 consumed more of the CPU resources.

5.2 UDP Analysis

The aim of this section is to present the analysis of all UDP results extracted from the experiments conducted on Jumbo frames. Results in discussions were based on the chosen metric discussed in chapter 2, section 2.1.1, which are throughput, delay and jitter. The CPU utilisations were also measured, and for UDP, the packets dropped were also measured in the experiments so, this is also included in the analysis.

5.2.1 UDP Throughput on IPv6 & IPv4 Jumbo Frames

This section presents the line graphs of the throughput results from the experiments conducted on a network employing Jumbo frames on IPv6 and using TCP as the transmission protocol. Figure 5-2-1 below presents the Jumbo frames throughput results of experiments conducted on various operating systems implemented on a network employing IPv6 and used UDP as the transmission protocol.
The following list is the analysis of the throughput results for all of the operating systems on UDP IPv6 presented in Figure 5-2-1 above:

- Fedora on the small frame sizes ranging from 1518 to 3392 Bytes outperformed the rest of the operating systems involved.
- Frame sizes ranging from 4329 to 9014 Bytes, OpenSUSE outperformed all the Linux operating systems involved in this study.
- On the smallest frame sizes ranging from 1518 to 3392 Bytes, Microsoft Windows 7 produced the least in terms of throughput values.
- On the largest frame size 9014 Bytes, OpenSUSE outperformed the remaining operating systems while Microsoft Windows Server 2003 yielded the least in terms of throughput values.
- The three Linux distributions outperformed all the Microsoft Windows operating systems entailed in this study on the largest frame size.

The following list is the analysis of the throughput results for all of the operating systems on UDP IPv4 presented in Figure 5-2-2 above:

- Fedora on the small frame sizes ranging from 1518 to 3392 Bytes outperformed the rest of the operating systems involved.
- Frame sizes ranging from 4329 to 9014 Bytes, OpenSUSE outperformed all the Linux operating systems involved in this study.
- On the smallest frame sizes ranging from 1518 to 3392 Bytes, Microsoft Windows 7 produced the least in terms of throughput values.
- On the largest frame size 9014 Bytes, OpenSUSE outperformed the remaining operating systems while Microsoft Windows Server 2003 yielded the least in terms of throughput values.
- The three Linux distributions outperformed all the Microsoft Windows operating systems entailed in this study on the largest frame size.
- On the three smallest frame sizes ranging from 1518 to 3392 Bytes, Fedora Core outperformed the rest of the operating systems involved.
- Frame sizes ranging from 4329 to 8037 Bytes, all of the operating systems generated very similar performance with regards to throughput values excluding Microsoft Windows Server 2003.
- Microsoft Windows Server 2003 produced the lowest throughput values on all of the frame sizes apart from frame size 1518 Bytes where Microsoft Windows Server 2008 yields 6% lower than Microsoft Windows Server 2003.
- On the three smallest frame sizes ranging from 1518 to 3392 Bytes, all the operating systems performed very differently. There was a 7% difference between Fedora, which yielded the most throughput values and OpenSUSE which came next.
- On the largest frame size, OpenSUSE and Ubuntu yielded very similar throughput values and 8% more than Fedora which yielded the second highest. With regards to the Microsoft operating systems, Microsoft Windows 7 performed better than Microsoft Windows Server 2008 by 4% and Microsoft Windows Server 2003 was the lowest in terms of throughput by 8% lower than Microsoft Windows Server 2008.
- According to Figure 5-2-2, Microsoft Windows Server 2003 yielded 25.2% lower throughput than the rest of the operating systems involved.

### 5.2.2 Compare UDP Throughputs on IPv4 & IPv6 Jumbo Frames

This section presents in line graphs a comparison of IPv4 and IPv6 throughput results on all of the six operating systems involved using Jumbo frames while UDP was the transmission protocol employed.

![Compare UDP Jumbo Frames Throughput on Various Operating Systems on IPv6 & IPv4](image)

The following list compares and analyses the throughput results for all of the operating systems on UDP IPv6 and IPv4 presented in Figure 5-2-3 above:
IPv6 and IPv4 Jumbo frames on all of the Linux distributions produced very similar values in terms of throughput on frame sizes ranging from 4329 to 8037 Bytes excluding Fedora IPv6 on frame size 8037 in particular, it dropped by 26%.

IPv6 and IPv4 on Fedora yielded very similar throughput values apart from the smallest frame size where IPv4 dropped by 7%.

IPv4 on Microsoft Windows Server 2003 produced the lowest throughput values on most of the frame sizes apart from IPv4 on Microsoft Windows Server 2008 on the smallest frame size where throughput values dropped by 11%.

Based on the graph showing in Figure 5-2-3, it can be said that IPv6 on Microsoft Windows Server 2008 on most of the frame sizes required more of the CPU time than IPv4. However, frame sizes 3392 and 9014 Bytes, IPv4 utilises 2% more of the CPU time than IPv6.

Based on the graph showing in Figure 5-2-3, it can be said that IPv4 on Microsoft Windows 7 on most of the frame sizes utilised the CPU time more than IPv6. However, in some particular frame sizes, 2455, 6203 and 9014 Bytes, IPv6 required more of the CPU time by 5% more than IPv4.

Based on the graph showing in Figure 5-2-3, it can be said that IPv6 on Microsoft Windows Server 2003 on all of the frame sizes required more of the CPU resources than IPv4 except for the largest frame size 9014 Bytes where IPv4 utilised more of the CPU time by 3%.

Based on the graph showing in Figure 5-2-3, it can be said that most of the frame sizes on Linux Fedora, IPv4 required more of the CPU time than IPv6 except for the smallest frame size where IPv6 uses 7% more than IPv4.

Based on the graph showing in Figure 5-2-3, it can be said that most of the frame sizes on IPv4 on Linux Ubuntu required more of the CPU time apart from some particular frame sizes 1518, 4329 and 5266 Bytes where IPv6 required 1% more of the CPU time.

Based on the graph showing in Figure 5-2-3, it can be said that IPv4 on OpenSUSE for all of the frame sizes utilised the CPU time more than IPv6 except for frame size 1518 Bytes in particular where IPv6 required 3% more of the CPU time.

According to Figure 5-2-3, Microsoft Windows Server 2003 IPv4 yielded 25.2% lower throughput than the rest of the operating systems involved while the rest on IPv6 and IPv4 yielded similar performance.

5.2.3 UDP One Way Delay on IPv6 & IPv4 Jumbo Frame

Figure 5-2-4 below presents the Jumbo frames one way delay IPv6 results of experiments conducted on various operating systems using UDP as the transmission protocol.
The following list is the analysis of the OWD results for all of the operating systems on UDP IPv6 presented in Figure 5-2-4 above:

- Among the three Microsoft operating systems, Microsoft Windows Server 2003 produced less delay values than the other two operating systems by maximum of 20% on frame size 9014 Bytes.
- Microsoft Windows Server 2008 and Microsoft Windows 7 produced the same values in terms of delay on frame sizes 4329, 5266 and 9014 Bytes.
- The smallest frame size produced the highest amount of delays than any of the operating systems involved in this study. However, there were 1.3% more delays than Microsoft Windows 7.
- On frame size 2455 Bytes, Microsoft Windows 7 produced 8% more delays when compared to Microsoft Windows Server 2008.
- On frame size 3392 Bytes, Microsoft Windows Server 2008 created 2.6% more delays than Microsoft Windows Server 2008.
- On frame size 6203 Bytes, Microsoft Windows 7 produced 1% more delays when compared to Microsoft Windows Server 2008.
- On frame size 7140 Bytes, Microsoft Windows Server 2008 created 3% more delays than Microsoft Windows Server 2008.
- On frame size 8037 Bytes, Microsoft Windows Server 2008 created 1% more delays than Microsoft Windows Server 2008.
- According to Figure 5-2-4, all of the Linux distributions involved in this study produced similar values in terms of delay on all frame sizes which was lower than all of the Microsoft Windows operating systems involved.

Figure 5-2-5 next presents the Jumbo frames one way delay IPv4 results of experiments conducted on various operating systems using UDP as the transmission protocol.
The following list is the analysis of the OWD results for all of the operating systems on UDP IPv4 presented in Figure 5-2-5 above:

- For all of the Microsoft operating systems involved in this study, they created more delays than the Linux based operating systems except for Microsoft Windows Server 2003 on the largest frame size 9014 Bytes in which Microsoft Windows Server 2003 creates same amount of delays as all of the Linux operating systems.
- Among the three Microsoft operating systems, Microsoft Windows Server 2003 produced less delay values than the other two operating systems by maximum of 22.8% less than Microsoft Windows 7 on the smallest frame size 1518 Bytes.
- On the smallest frame size, Microsoft Windows Server 2008 produced the highest amount of delays than any of the operating systems involved in this study. However, 3.6% more delays than Microsoft Windows 7.
- Frame sizes ranging from 255 to 4329 Bytes, and on frame sizes 6203 and 8037 Bytes, Microsoft Windows 7 created more delay than any of the operating systems involved by up to 4.4% more than Microsoft Windows Server 2008 which the next one up.
- Apart from the smallest frame size 1518 Bytes, frame sizes 5266, 7140 and the largest frame size 9014 Bytes, Microsoft Windows Server 2008 produced the highest delay values than any of the operating systems involved in this study by up to .4% on frame size 5266 Bytes more than Microsoft Windows 7 in particular.
- According to Figure 5-2-5, all of the Linux distributions involved in this study on all frame sizes produced similar values in terms of delay which was lower than the delay values produced by all of the Microsoft Windows operating systems involved.

5.2.4 Compare UDP One Way Delay on IPv4 & IPv6 Jumbo Frames

This section presents in line graphs a comparison of IPv4 and IPv6 delay values on all of the six operating systems involved in this study using Jumbo frames while UDP was the transmission protocol employed.
The following list compares and analyses the Jumbo frames delay results for all of the operating systems on UDP IPv6 and IPv4 presented in Figure 5-2-6 above:

- Microsoft Windows Server 2003 produced the lowest delay values on all of the frame sizes on both IPv6 and IPv4 when compared to other Microsoft operating systems involved in this study.
- Based on the graph showing in Figure 5-2-6, it can be said that IPv6 on Microsoft Windows Server 2003 on frame sizes ranging from 1518 to 3392 Bytes and on the largest frame size 9014 Bytes, created more delays than IPv4. However, on frame sizes ranging from 4329 to 8037 Bytes, IPv6 and IPv4 on Microsoft Windows Server 2003 produced similar amount in terms of delay values.
- Based on the graph showing in Figure 5-2-6, it can be said that IPv6 on frame sizes ranging from 2455 to the largest frame size 9014 Bytes created up to 0.6% more delays than IPv4 on frame size 9014 Bytes in particular on Microsoft Windows Server 2008.
- Based on the graph showing in Figure 5-2-6, it can be said that on Microsoft Windows Server 2008, IPv4 on the smallest frame size 1518 Bytes created 0.6% more delays than IPv6.
- Based on the graph showing in Figure 5-2-6, it can be said that for Microsoft Windows 7, IPv6 and IPv4 produced similar amount of delays on most of the frame sizes apart from frame size 5266 and 9014 Bytes where IPv6 created up to 0.6% more delays on the largest frame size 9014 Bytes in particular.
- Based on the graph showing in Figure 5-2-6, it can be said that IPv6 on Microsoft Windows Server 2003 created more delays than IPv4 on most of the frame sizes apart from frame sizes 7140 and 8037 Bytes where IPv4 created only 0.3% on frame size 8037 Bytes more delays than IPv6.
Based on the graph showing in Figure 5-2-6, it can be said that on Microsoft Windows operating systems, IPv6 created more delays on most of the Jumbo frame sizes involved in this study than IPv4.

According to Figure 5-2-6, IPv6 and IPv4 on all of the Linux distributions involved in this study on all frame sizes produced similar delay values which were lower than the delay values produced by all of the Microsoft Windows operating systems involved.

Section 5-2-5 next presents the Jumbo frames IPv6 and IPv4 jitter results of experiments conducted on various operating systems using UDP as the transmission protocol.

5.2.5 UDP Jitter Results on IPv6 & IPv4 Jumbo Frame

Figure 5-2-7 below presents the Jumbo frames jitter IPv6 results of experiments conducted on various operating systems using UDP as the transmission protocol.

The following list is the analysis of the jitter values for all of the operating systems on UDP IPv6 presented in Figure 5-2-7 above:
- All of the Linux distributions involved in this study produced the very similar values in terms of jitter on all frame sizes.
- For all of the Microsoft operating systems involved in this study, they created more jitters than the Linux based operating systems.
- Among the three Microsoft operating systems, Microsoft Windows Server 2003 produced less jitter than the other two Microsoft operating systems to a maximum of 0.2% on frame size 9014 Bytes.
- On most of the frame sizes, Microsoft Windows 7 created more jitter than Microsoft Windows Server 2008 apart from frame sizes 6203 Bytes and the largest frame size 9014 Bytes where Microsoft Windows Server 2008 generated more jitter.
According to Figure 5-2-7, IPv6 on all of the Linux distributions involved in this study on all frame sizes produced similar jitter values which were lower than the jitter values produced by all of the Microsoft Windows operating systems involved.

Figure 5-2-8 below presents the Jumbo frames IPv4 jitter results of experiments conducted on various operating systems using UDP as the transmission protocol.

The following list is the analysis of the jitter values for all of the operating systems on UDP IPv4 presented in Figure 5-2-8 above:

- For all of the Microsoft Windows Server 2008 and Microsoft Windows 7 created more jitters than any of the operating systems involved in this study.
- Among the three Microsoft operating systems, Microsoft Windows Server 2003 produced less jitter values than the other two Microsoft operating systems.
- For most of the frame sizes, Microsoft Windows Server 2003 had very similar jitter values to that of all the Linux based operating systems except on frame sizes 4329 and 6203 where Microsoft Windows Server 2003 created more jitters than the Linux based operating systems. However, on the largest frame size 9014 Bytes, Microsoft Windows Server 2003 created less jitter than all of the Linux operating systems.
- On the two smallest frame sizes 1518 and 2455 Bytes, also frame sizes 4329, 7140 and the two largest frame sizes 8037 and 9014 Bytes, Microsoft Windows Server 2008 produced the highest values of jitter than any of the operating systems involved in this study.
- According to Figure 5-2-8, IPv4 on all of the Linux distributions and Microsoft Windows Server 2003 involved in this study on all frame sizes produced similar jitter values which were lower than the jitter values produced by all of the Microsoft Windows Server 2008 and Microsoft Windows 7.
Section 5-2-6 next presents the comparison of Jumbo frames IPv6 and IPv4 jitter results of experiments conducted on various operating systems using UDP as the transmission protocol.

5.2.6 Compare UDP Jitter Results on IPv4 & IPv6 Jumbo Frames

This section presents in line graphs a comparison of IPv6 and IPv4 values of jitter on all of the six operating systems involved in this study using Jumbo frames while UDP was the transmission protocol employed.

![Compare UDP Jumbo Frames Jitter on Various Operating Systems on IPv6 & IPv4](image)

Figure 5-2-9: Compare Jumbo frames UDP jitter results on IPv6 and IPv4 on various operating systems

The following list compares and analyses the Jumbo frames jitter results for all of the operating systems on UDP IPv6 and IPv4 presented in Figure 5-2-9 above:

- On both IPv6 and IPv4, Jumbo frames on all of the Linux distributions produced very similar values in terms of jitter on most of the frame sizes except on the two largest frame sizes where OpenSUSE IPv4 generated more jitter.
- Based on the graph showing in Figure 5-2-9, it can be said that on Microsoft Windows Server 2008 IPv4 on most of the frame sizes produced more jitter than IPv6. However, on frame sizes 3392 and 5266 Bytes IPv6 produced more jitter than IPv4.
- Based on the graph showing in Figure 5-2-9, it can be said that on Microsoft Windows 7, IPv6 on most of the frame sizes produced more jitter than IPv4. However, on frame sizes 2455 and 4329 Bytes IPv4 produced more jitter than IPv6.
- Based on the graph showing in Figure 5-2-9, it can be said that on Microsoft Windows Server 2003, IPv6 on most of the frame sizes produced more jitter than IPv4. However, on frame sizes 4329 and 6203 Bytes IPv4 produced more jitter than IPv6.
- According to Figure 5-2-9, IPv6 and IPv4 on all of the Linux distributions and IPv4 on Microsoft Windows Server 2003 involved in this study on all frame sizes produced similar jitter values which were lower than the jitter values produced by all of the Microsoft Windows Server 2008 and Microsoft Windows 7 IPv6 and IPv4.
Section 5.2.7 next presents the packet loss results of Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems using UDP as the transmission protocol.

5.2.7 **UDP Packets Dropped on IPv6 & IPv4 Jumbo Frame**

Figure 5.2-10 below presents the Jumbo frames IPv6 packets lost on experiments conducted on various operating systems using UDP as the transmission protocol.

![UDP IPv6 - Packets Dropped](image)

The following list is the analysis of the packets dropped for all of the operating systems on UDP IPv6 presented in Figure 5.2-10 above:

- Among all Microsoft Windows operating systems, Microsoft Windows Server 2003 recorded the highest in terms of packets dropped.
- Microsoft Windows Server 2008 was measured with the lowest packets dropped on most of the frame sizes apart from frame sizes 5266 and 9014 Bytes where Microsoft Windows 7 dropped 1.6% more packets than Microsoft Windows Server 2008.
- On the smallest frame size 1518 Bytes, Linux Ubuntu was measured with the highest number in terms of packets dropped which is 5.8% more than Microsoft Windows Server 2008.
- On frame size 2455 Bytes, Microsoft Windows Server 2008 and Microsoft Windows 7 were measured the lowest while Linux Fedora and Microsoft Windows Server 2003 dropped 1.8% more packets than Microsoft Windows Server 2008 and Microsoft Windows 7. However, Linux Ubuntu and OpenSUSE were measured with the highest of 2.6% more packets dropped than Microsoft Windows Server 2008 and Microsoft Windows 7.
- Frame sizes ranging from 4329 to 6203 Bytes, Microsoft Windows Server 2008 and Microsoft Windows 7 still measured with the lowest values in terms of packets dropped. However, on the same frame sizes, all of the Linux operating systems were recorded
with the highest 2.6% more packets dropped than Microsoft Windows Server 2008 and Microsoft Windows 7.

- Frame sizes 7140 and 8037 Bytes, Microsoft Windows Server 2008 was measured to be the lowest while the rest of the operating systems involved had similar values in terms of packets dropped. However on frame size 7140 Bytes Linux Fedora was measured higher than others but 2% higher than Microsoft Windows Server 2008.

- On frame size 8037 Bytes, Microsoft Windows Server 2008 was measured to be the lowest while Fedora was 3.4% higher than Microsoft Windows Server 2008. However, 1% higher than Ubuntu and OpenSUSE which were measured second highest and next were Microsoft Windows Server 2003 and Microsoft Windows 7 with 2% lower than Fedora.

- On the largest frame size 9014 Bytes, Microsoft Windows 7 was recorded to be the lowest and Microsoft Windows Server 2003 was the highest with 5.6% more packets dropped than Microsoft Windows 7.

- On the largest frame size 9014 Bytes, Microsoft Windows Server 2008 was measured with 1.2% more packets dropped than Microsoft Windows 7. OpenSUSE was 0.4% and OpenSUSE was 0.2% and Fedora was 0.6% more packets dropped than Microsoft Windows Server 2008.

- According to Figure 5-2-10, it can be concluded that the Linux based operating systems on IPv6 dropped more packets than the Windows based operating systems when UDP was employed as the transmission protocol.

Figure 5-2-11 below presents the Jumbo frames IPv4 packets loss on experiments conducted on various operating systems using UDP as the transmission protocol.

![Figure 5-2-11: UDP packets dropped on Jumbo frames Ipv4](image-url)
The following list is the analysis of the packets dropped for all of the operating systems on UDP IPv4 presented in Figure 5-2-11 above:

- On the smallest frame size 1518 Bytes, Microsoft Windows 7 and Microsoft Windows Server 2008 were measured with lowest amounts of packets dropped while, Fedora dropped 5.2% more packets.
- Linux Ubuntu and OpenSUSE on all of the frame sizes showed a very steady and similar performance however, Linux was measured with 0.2% more in terms of packets dropped.
- Frame sizes ranging from 4329 to 8037 Bytes, the three Linux distributions involved in this study were measured to have similar amounts of packets dropped.
- On frame size 8037 Bytes, Microsoft Windows Server 2008 was measured with the lowest in terms of packets lost while Microsoft Windows Server 2003 was the highest with 4.6% more packets lost.
- On the largest frame size 9014 Bytes, Microsoft Windows 7 and Linux Fedora were measured with the highest amounts of packets dropped while OpenSUSE, Fedora and Microsoft Windows Server 2008 were measured 1.1% lower.
- On the largest frame size also Microsoft Windows Server 2003 was measured with the lowest amounts of packets lost 8% lower than Microsoft Windows 7 and Fedora which were the highest.
- According to Figure 5-2-11, it can be concluded that, Microsoft Windows Server 2003 on most of the frame sizes 19.4% dropped more packets than the rest of the operating systems involved.

Section 5-2-8 next presents the comparison of packets dropped on IPv6 and IPv4 results on experiments conducted on various operating systems using UDP as the transmission protocol.

### 5.2.8 Compare Packets Dropped Results on IPv4 & IPv6 Jumbo Frames

This section presents in line graphs a comparison of IPv6 and IPv4 packets dropped on all of the six operating systems involved in this study using Jumbo frames while UDP was the transmission protocol employed.
The following list compares and analyses the Jumbo frames jitter results for all of the operating systems on UDP IPv6 and IPv4 presented in Figure 5-2-12 above:

- Based on the graph showing in Figure 5-2-12, it can be said that on the two small frame sizes, frame size 1518 Bytes on Microsoft Windows Server 2008, IPv6 dropped 0.5% more packets than IPv4. Frame size 2455 Bytes, IPv6 and IPv4 on Microsoft Windows Server 2008 had similar amounts in terms of packets dropped.

- Based on the graph showing in Figure 5-2-12, it can be said that the packets lost rate of IPv4 on Microsoft Windows Server 2008 on frame sizes ranging from 3392 to 9014 Bytes was up to 2.2% more than that of IPv6.

- Based on the graph showing in Figure 5-2-12, it can be said that the packets dropped rate of Microsoft Windows 7 was different to that of Server 2008. For instance, IPv6 dropped more packets on frame sizes 1518, 3392 and 7140 Bytes to a maximum of 4% more than IPv4. On the other hand, IPv4 on frame sizes 2455, 4329 to 6203, 8037 and 9014 Bytes dropped up to 3.4% more packets than IPv6.

- Based on the graph showing in Figure 5-2-12, it can be said that Microsoft Windows Server 2003 performance in terms of packets lost was, IPv4 on frame sizes ranging from 1518 to 8037 Bytes had dropped to a maximum of 4.8% more packets than IPv6. On the largest frame size 9014 Bytes, IPv6 dropped 10.4% more packets when compared to that of IPv4.

- Based on the graph showing in Figure 5-2-12, it can be said that IPv6 and IPv4 on Linux Fedora had similar amount of packets dropped on frame sizes ranging from 2455 to 5266 Bytes and on frame sizes 7140 and 9014 Bytes. However, on the smallest frame size 1518 Bytes, IPv4 dropped 1.6% more packets than IPv6. On frame sizes 6203 and 8037 Bytes, IPv6 dropped up to 1% more packets than IPv4.

- Based on the graph showing in Figure 5-2-12, it can be said that IPv6 and IPv4 on Linux Ubuntu had very similar amounts of packets dropped on frame sizes ranging from 2455...
to 5266 Bytes and on 7140 to 9014 Bytes. However, on the smallest frame size, IPv4 dropped 0.4% more packets than IPv6. On frame size 6203 Bytes, IPv6 dropped 0.6% more packets than IPv4.

- Based on the graph showing in Figure 5-2-12, it can be said that IPv6 and IPv4 on OpenSUSE had very similar amounts of packets dropped on frame sizes ranging from 2455 to 5266 Bytes and from 7140 to 9014 Bytes. However, on the smallest frame size, 1518 Bytes, IPv4 dropped 0.6% more packets than IPv6. On frame size 6203 Bytes IPv6 dropped 0.6% more packets than IPv4.

- Based on the graph showing in Figure 5-2-12, it can be concluded that IPv4 dropped more packets than IPv6 however, Microsoft Windows Server 2003 dropped 19.4% more packets than any of the operating systems.

Section 5-2-9 next presents the Jumbo frames IPv6 and IPv4 CPU utilisations of Router 1 on experiments conducted on various operating systems using UDP as the transmission protocol.

### 5.2.9 UDP CPU Utilisations on IPv6 & IPv4 Jumbo Frame on Router 1

Figure 5-2-13 below presents the Jumbo frames IPv6 CPU utilisations on Router 1 on experiments conducted on various operating systems using UDP as the transmission protocol.

![Figure 5-2-13: UDP IPv6 CPU usage on Router 1 results on Jumbo frames on various operating systems.](image)

The following list is the analysis of the CPU utilisations on Router 1 for all of the operating systems on UDP IPv6 presented in Figure 5-2-13 above:

- Apart from Microsoft Windows 7, on frame sizes ranging from 4329 to 9014 Bytes, Microsoft Windows Server 2003 used less CPU resources than Microsoft Windows Server 2008 and all the Linux based operating systems involved in this study.
- On the three smallest frame sizes ranging from 1518 to 3392 Bytes, Linux Fedora required less CPU resources than Microsoft Windows Server 2003 but to a maximum of 1.3% more than Microsoft Windows 7 on frame size 3392 Bytes in particular.

- Linux Ubuntu demonstrated a very stable performance on all of the frame sizes. However, Linux Ubuntu required more time of the CPU than Microsoft Windows Server 2003 and Microsoft Windows 7. Ubuntu utilised the CPU more than Linux Fedora on most of the frame sizes apart from frame size 4329 Bytes where Fedora required 0.8% more of the CPU than Ubuntu.

- Microsoft Windows Server 2008 showed a very unsteady performance on Router 1 for all of the frame sizes and resulted in utilising more CPU resources on some particular frame sizes such as 5266 and 7140 Bytes where Microsoft Windows Server 2008 utilised the CPU resources 1% more than OpenSUSE which was the next highest on frame size 7140 Bytes in particular.

- OpenSUSE also showed a very unsteady performance on Router 1 for all of the frame sizes and resulted in utilising more CPU resources on some particular frame sizes such as 3392 and 9014 Bytes where OpenSUSE utilised the CPU resources 9.2% more than any other operating system involved in this study.

- According to Figure 5-2-13, it can be concluded that, Microsoft Windows 7 on IPV6 on Router 1 required less CPU resources on all of the frame sizes than any of the operating systems involved in this study.

Figure 5-2-14 below presents the Jumbo frames IPv4 CPU utilisations on Router 1 on experiments conducted on various operating systems using UDP as the transmission protocol.

![UDP IPv4 - CPU Usage on Router 1](image)

**Figure 5-2-14: IPv4 CPU usage on Router 1 results on Jumbo frames on various operating systems**

The following list is the analysis of the CPU utilisations on Router 1 for all of the operating systems on UDP IPv4 presented in Figure 5-2-14 above:
- Microsoft Windows 7 on IPv4 showed the lowest in terms of CPU utilisations on all of the frame sizes.
- Fedora apart from the lowest frame size showed a 2.4% more than Microsoft Windows 7 on all of the frame sizes ranging from 2455 to 9014 Bytes.
- On the smallest frame size, Microsoft Windows Server 2003 showed the second lowest but used 0.8% more of the CPU resources than Microsoft Windows 7.
- On frame sizes 3392, 4329 Bytes and on the largest frame size 9014 Bytes, Microsoft Windows Server 2003 showed a very similar performance in terms of CPU utilisations.
- Frame sizes ranging from 5266 to 8037 Bytes, Microsoft Windows Server 2003 utilised the CPU time 0.8% more than Fedora Core.
- Linux Ubuntu demonstrated a very steady performance on all of the frame sizes. However, Ubuntu utilised the CPU resources 0.8% more than Fedora.
- Microsoft Windows Server 2008 on the smaller frame sizes used more CPU resources than Microsoft Windows Server 2003 but on the larger frame sizes ranging from 5266 to 8037 Bytes, Microsoft Windows Server 2003 used 0.2% more CPU resources when compared to Microsoft Windows Server 2008.
- According to Figure 5-2-14, it can be concluded that, OpenSUSE on all frame sizes required more of the CPU resources than any of the operating systems involved.

Section 5-2-10 next presents the comparison of Jumbo frames IPv6 and IPv4 CPU utilisations results on Router 1 from the experiments conducted on various operating systems using UDP as the transmission protocol.

**5.2.10 Compare UDP CPU Usage for Router 1 on IPv4 & IPv6 Jumbo Frames**

This section presents in line graphs a comparison of IPv6 and IPv4 CPU utilisations results on Router 1 as presented on Figure 5-2-13 and Figure 5-2-14 for all of the six operating systems involved in this study using Jumbo frames while UDP was the transmission protocol employed.
The following list compares UDP IPv6 and IPv4 CPU utilisation results measured on Router 1 from the experiments conducted using Jumbo frames for all of the operating systems as presented on Figure 5-2-15 above:

- Based on the graph presented on Figure 5-2-15, it can be said that IPv6 on Microsoft Windows Server 2008 required 2% more of the CPU resources on all of the frame sizes than IPv4.
- Based on the graph presented on Figure 5-2-15, it can be said that on the smaller frame sizes ranging from 1518 to 3392 Bytes, also on frame size 8037 Bytes, IPv4 required up to 0.8% more of the CPU resources than IPv6. However, on frame sizes ranging from 4329 to 7140 Bytes and 9014 Bytes, IPv6 required 0.2% more of the CPU time than IPv4.
- Based on the graph presented on Figure 5-2-15, it can be said that Microsoft Windows Server 2003 on the smallest frame size and on frame sizes ranging from 5266 to 9014 Bytes, IPv4 required up to 1.4% more of the CPU time than IPv6.
- Based on the graph presented on Figure 5-2-15, it can be said that Fedora IPv4 on the smallest frame size 1518 Bytes required 0.8% more of the CPU time than IPv6. However, frame sizes ranging from 2455 to 9014 Bytes, IPv6 required up to 1.8% more of the CPU resources than IPv4.
- Based on the graph presented on Figure 5-2-15, it can be said that Linux Ubuntu frame sizes 1518, 2455, 5266, and 8037 Bytes, IPv6 required 0.2% of the CPU time than IPv4. However, frame sizes 3392, 4329, 7140 and 9014 Bytes, IPv4 required up to 0.4% more of the CPU time than IPv6.
- Based on the graph presented on Figure 5-2-15, it can be said that on frame sizes ranging from 1518 to 5266 Bytes and also the largest frame size 9014 Bytes, IPv6...
required up to 6% more of the CPU time than IPv4. However, on frame size 6203 Byte, IPv4 used 1.2% more of the CPU resources than IPv6.

- Based on the graph presented on Figure 5-2-15, it can be concluded that IPv6 on Router 1 required more of the CPU resources than IPv4.

Section 5-2-11 next presents the CPU utilisation results of Jumbo frames IPv6 and IPv4 on Router 2 from the experiments conducted on various operating systems using UDP as the transmission protocol.

### 5.2.11 UDP CPU Utilisations on IPv6 & IPv4 Jumbo Frame on Router 2

Figure 5-2-16 below presents the Jumbo frames IPv6 CPU utilisations on Router 2 on experiments conducted on various operating systems using UDP as the transmission protocol.

The following list is the analysis of the CPU utilisations on Router 2 for all of the operating systems on UDP IPv6 presented in Figure 5-2-16:

- On smaller frame sizes ranging from 1518 to 5266 Bytes, OpenSUSE used less of the CPU time than any of the operating systems involved excluding frame size 4329 Bytes where Microsoft Windows 7 0.8% less than OpenSUSE.
- On the two small frame sizes 1518 and 2455 Bytes, Microsoft Windows 7 required .2% more of the CPU time than OpenSUSE however; Microsoft Windows Server 2008 on the same frame sizes required 0.2% more of the CPU time than Microsoft Windows 7.
- Linux Ubuntu on most of the frame sizes was recorded as the highest in terms of CPU utilisations than any other operating systems involved in this study. However, on frame
sizes 5266 and 9014 Bytes, Fedora was the highest user by 0.4% higher than Ubuntu on frame size 5266 Bytes.

- Excluding Ubuntu and Fedora, the three Microsoft Windows operating systems that involved in this study and OpenSUSE, they behave very closely to each other apart from the smallest frame size 1518 Bytes where Microsoft Windows Server 2003 required 1% more of the CPU time than OpenSUSE.

- Between Fedora and Ubuntu, on the smallest frame size, Ubuntu was 1.6% higher than Fedora; on frame size 2455 Bytes they produced similar value. However, on frame sizes 3392, 4329 and 8037 Bytes, Ubuntu used 1% more of the CPU resources than Fedora.

- According to Figure 5-2-16, it can be concluded that Linux Ubuntu on IPv6 and UDP as the transmission protocol on Router 2 required more of the CPU resources on most of the frame sizes than any other operating systems.

Following in Figure 5-2-17 below presents the Jumbo frames IPv4 CPU utilisations on Router 2 on experiments conducted on various operating systems using UDP as the transmission protocol.

![Figure 5-2-17: UDP IPv4 CPU usage on Router 2 on Jumbo frames on various operating systems](image)

The following list is the analysis of the CPU utilisations on Router 2 for all of the operating systems on UDP IPv4 presented in Figure 5-2-17 above:

- Both Microsoft Windows Server 2008 and Microsoft Windows 7 measured similar but the lowest in terms of CPU usage.

- Microsoft Windows Server 2003 on the small frame sizes ranging from 1518 to 4329 Bytes recorded similar values and patterns in terms of CPU utilisations. However, on frame sizes ranging from 5266 to 8037 Bytes required 1.4% more of the CPU resources than Microsoft Windows 7. On the largest frame size 9014 Bytes. Microsoft Windows Server 2003 measured to be the lowest 0.6% lower than Microsoft Windows 7.
- Linux Fedora demonstrated a very stable performance in terms of CPU utilisations for all frame sizes but still 1% more than Microsoft Windows 7 in some particular frame size. On the smallest frame size 1518 Bytes, OpenSUSE required 1.4% more of the CPU resources than Microsoft Windows 7 however, frame sizes 4329, 5266, 7140 and 9014 Bytes, OpenSUSE had similar values in terms of CPU utilisations. On frame size 6203 Bytes in particular, OpenSUSE unpredictably required 7% more of the CPU resources than Microsoft Windows 7.

- According to Figure 5-2-17, it can be concluded that Linux Ubuntu required more of the CPU time on most of the frame sizes.

Section 5-2-12 next presents the comparison of Jumbo frames IPv6 and IPv4 CPU utilisations results on Router 2 from the experiments conducted on various operating systems using UDP as the transmission protocol.

### 5.2.12 Compare UDP CPU Usage for Router 2 on IPv4 & IPv6 Jumbo Frames

This section presents in line graphs a comparison of IPv6 and IPv4 CPU utilisations results on Router 2 as presented on Figure 5-2-16 and Figure 5-2-17 for all of the six operating systems involved in this study using Jumbo frames while UDP was the transmission protocol employed.

**Figure 5-2-18: Comparison of UDP IPv6 & IPv4 CPU utilisations on Router 2**

The following list compares UDP IPv6 and IPv4 CPU utilisations results measured on Router 2 from the experiments conducted using Jumbo frames on all of the operating systems as presented on Figure 5-2-18:

- Based on the graph presented in Figure 5-2-18, it can be said that Microsoft Windows Server 2008 IPv6 required more of the CPU resources on most of the frame sizes than
IPv4. However, frame sizes 5266 and 8037 Bytes required up to 0.2% more of the CPU time than IPv6.

- Based on the graph presented in Figure 5-2-18, it can be said that on frame size 3392 Bytes and on the larger frame sizes ranging from 5266 to 9014 Bytes, Microsoft Windows 7 IPv6 utilised the CPU resources up to 1.8% more than IPv4. However, on frame sizes 1518, 2455 and 4329 Bytes IPv4 on Microsoft Windows 7 required 0.6% more of the CPU time than IPv6.

- Based on the graph presented in Figure 5-2-18, it can be said that, on the smaller frame sizes 1518 to 4329 Bytes and also on the largest frame size 9014 Bytes, IPv6 on Microsoft Windows Server 2003 required up to 0.8% more of the CPU resources than IPv4. However, on frame sizes ranging from 6203 to 8037 Bytes, IPv4 required up to 1.4% more of the CPU time than IPv6.

- Based on the graph presented in Figure 5-2-18, it can be said that, IPv6 on Fedora utilised more of the CPU resources than IPv4 on all of the frame sizes and up to 1.4% more on some particular frame sizes.

- Based on the graph presented in Figure 5-2-18, it can be said that, IPv4 on OpenSUSE on all of the frame sizes required more of the CPU time than IPv6.

- According to Figure 5-2-18, it can be concluded that, Linux Ubuntu on most of the frame sizes on both IPv6 and IPv4 utilised up to 0.6% more of the CPU time than the rest of the operating systems.

Section 5-2-13 next presents the comparison of throughput for the two transmission protocols (TCP & UDP) employed by this study on Jumbo frames. This the results from the experiments conducted on various operating systems using IPv6.

**5.2.13 Compare TCP & UDP Throughput on IPv6 Jumbo Frames**

This section presents in line graphs a comparison of throughput results on IPv6 for the two transmission protocols employed by this study. This result was an outcome of experiments conducted on various operating systems utilising Jumbo frames.
The following list compares Jumbo frames utilising TCP and UDP throughput results on a network employing IPv6 from the experiments conducted on all of the operating systems as presented on Figure 5-2-19:

- Based on the graph presented in Figure 5-2-19, it can be said that Microsoft Windows Server 2008 utilising TCP on IPv6 on smaller frame sizes ranging from 1518 to 3392 Bytes and on frame sizes 5266, 6203 and the largest frame size 9014 Bytes outperformed Microsoft Windows Server 2008 utilising UDP on IPv6 by up to 31%.

- Based on the graph presented in Figure 5-2-19, it can be said that Microsoft Windows Server 2008 utilising UDP on IPv6 on frame sizes 4329, 7140 and 8037 Bytes outperformed Microsoft Windows Server 2008 utilising TCP on IPv6 by up to 18% on frame size 8037 Bytes in particular.

- Based on the graph presented in Figure 5-2-19, it can be said that Microsoft Windows 7 utilising TCP on IPv6 on smaller frame sizes ranging from 1518 to 3392 Bytes and also on frame sizes 5266, 6203 and the largest frame size 9014 Bytes outperformed Microsoft Windows 7 utilising UDP on IPv6 by up to 31.6% on frame size 1518 Bytes in particular.

- Based on the graph presented in Figure 5-2-19, it can be said that Microsoft Windows 7 utilising UDP on IPv6 on frame sizes 4329, 7140 and 8037 Bytes outperformed Microsoft Windows 7 utilising TCP on IPv6 by up to 16.2% on frame size 8037 Bytes in particular.

- Based on the graph presented in Figure 5-2-19, it can be said that Microsoft Windows Server 2003 utilising TCP on IPv6 on smaller frame sizes ranging from 1518 to 3392 Bytes and also on frame sizes 5266, 6203 and the largest frame size 9014 Bytes outperformed Microsoft Windows Server 2003 utilising UDP on IPv6 by up to 38% on the largest frame size 9014 Bytes in particular.

**Figure 5-2-19: Comparison of TCP & UDP Jumbo frames throughput on IPv6**
Based on the graph presented in Figure 5-2-19, it can be said that Microsoft Windows Server 2003 utilising UDP on IPv6 on frame sizes 4329, 7140 and 8037 Bytes outperformed Microsoft Windows Server 2003 utilising TCP on IPv6 by up to 17.8% on frame size 8037 Bytes in particular.

Based on the graph presented in Figure 5-2-19, it can be said that Linux Fedora utilising TCP on IPv6 on smaller frame sizes ranging from 1518 to 3392 Bytes and also on frame sizes 5266, 6203, 8037 and the largest frame size 9014 Bytes outperformed Fedora utilising UDP on IPv6 by up to 15.6% on the largest frame size 9014 Bytes in particular.

Based on the graph presented in Figure 5-2-19, it can be said that Linux Fedora utilising UDP on IPv6 on frame sizes 4329 and 7140 Bytes outperformed Fedora utilising TCP on IPv6 by up to 7.2% on frame size 7140 Bytes in particular.

Based on the graph presented in Figure 5-2-19, it can be said that Linux Ubuntu utilising TCP on IPv6 on smaller frame sizes ranging from 1518 to 3392 Bytes and also on frame sizes 5266, 6203 and the largest frame size 9014 Bytes outperformed Ubuntu utilising UDP on IPv6 by up to 24.4% on the smallest frame size 1518 Bytes in particular.

Based on the graph presented in Figure 5-2-19, it can be said that Linux Ubuntu utilising UDP on IPv6 on frame sizes 4329, 7140 and 8037 Bytes outperformed Ubuntu utilising TCP on IPv6 by up to 17.6% on frame size 8037 Bytes in particular.

Based on the graph presented in Figure 5-2-19, it can be said that OpenSUSE utilising TCP on IPv6 on smaller frame sizes ranging from 1518 to 3392 Bytes and also on frame sizes 5266, 6203 and the largest frame size 9014 Bytes outperformed OpenSUSE utilising UDP on IPv6 by up to 21.2% on the smallest frame size 1518 Bytes in particular.

Based on the graph presented in Figure 5-2-19, it can be said that OpenSUSE utilising UDP on IPv6 on frame sizes 4329, 7140 and 8037 Bytes outperformed OpenSUSE utilising TCP on IPv6 by up to 18% on frame size 7140 Bytes in particular.

Based on the graph presented in Figure 5-2-19 and the analysis outlined above, it can be concluded that all of the operating systems involved in this study showed a very similar pattern in terms of their performance on a network utilising Jumbo frames on IPv6. Apart from Linux Fedora, this performance showed that TCP outperformed UDP on all of the operating systems and on most of the frame sizes except for frame sizes 4329, 7140 and 8037 Bytes. Linux Fedora on frame size 8037 TCP outperformed UDP by 10%.

Based on the graph presented in Figure 5-2-19 and the analysis outlined above, it can also be concluded that all of the operating systems involved in this study showed a very similar pattern in terms of their performance on a network utilising Jumbo frames on IPv6 on frame size 4329 Bytes where their performance in terms of throughput dropped by up to 74.6% when compared to the performance of UDP on IPv6.
Section 5.2.14 next presents the comparison of throughput for the two transmission protocols (TCP & UDP) employed by this study on Jumbo frames. This the results from the experiments conducted on various operating systems using IPv4.

### 5.2.14 Compare TCP & UDP Throughput on IPv4 Jumbo Frames

This section presents in line graphs a comparison of throughput results on IPv4 for the two transmission protocols employed by this study. This result was an outcome of experiments conducted on various operating systems utilising Jumbo frames.

![Compare Jumbo Frames TCP & UDP on IPv4](image)

The following list compares Jumbo frames utilising TCP and UDP throughput results on a network employing IPv4 from the experiments conducted on all of the operating systems as presented on Figure 5.2-20:

- Based on the graph presented in Figure 5-2-20, it can be said that Microsoft Windows Server 2008 utilising TCP on IPv4 on smaller frame sizes ranging from 1518 to 3392 Bytes and on frame sizes 5266, 6203 and the largest frame size 9014 Bytes outperformed Microsoft Windows Server 2008 utilising UDP on IPv4 by up to 43.6% on the smallest frame size.

- Based on the graph presented in Figure 5-2-20, it can be said that Microsoft Windows Server 2008 utilising UDP on IPv4 on frame sizes 4329, 7140 and 8037 Bytes outperformed Microsoft Windows Server 2008 utilising TCP on IPv4 by up to 2% on frame size 8037 Bytes in particular.

- Based on the graph presented in Figure 5-2-20, it can be said that Microsoft Windows 7 utilising TCP on IPv4 on smaller frame sizes ranging from 1518 to 3392 Bytes and on frame sizes 5266, 6203 and the largest frame size 9014 Bytes outperformed Microsoft Windows 7 utilising UDP on IPv4 by up to 30.4% on the smallest frame size.
Based on the graph presented in Figure 5-2-20, it can be said that Microsoft Windows 7 utilising UDP on IPv4 on frame sizes 4329, 7140 and 8037 Bytes outperformed Microsoft Windows 7 utilising TCP on IPv4 by up to 16.2% on frame size 8037 Bytes in particular.

Based on the graph presented in Figure 5-2-20, it can be said that Microsoft Windows Server 2003 utilising TCP on IPv4 on all of the frame sizes outperformed UDP on IPv4 except for frame size 4329 Bytes where UDP on IPv4 outperformed TCP by up to 53.8%.

Based on the graph presented in Figure 5-2-20, it can be said that Linux Fedora utilising TCP on IPv4 on smallest frame size 1518 Bytes and the largest frame size 9014 Bytes outperformed Fedora utilising UDP on IPv4 by up to 16.8% on the smallest frame size.

Based on the graph presented in Figure 5-2-20, it can be said that Linux Fedora utilising UDP on IPv4 on frame sizes ranging from 2455 to 8037 Bytes outperformed Fedora utilising TCP on IPv4 by up to 17.6% on frame size 7140 Bytes in particular.

Based on the graph presented in Figure 5-2-20, it can be said that Linux Ubuntu utilising TCP on IPv4 on smaller frame sizes ranging from 1518 to 3392 Bytes and on frame sizes 5266, 6203 and the largest frame size 9014 Bytes outperformed Ubuntu utilising UDP on IPv4 by up to 25.8% on the smallest frame size.

Based on the graph presented in Figure 5-2-20, it can be said that Linux Ubuntu utilising UDP on IPv4 on frame sizes 4329, 7140 and 8037 Bytes outperformed Ubuntu utilising TCP on IPv4 by up to 19% on frame size 8037 Bytes in particular.

Based on the graph presented in Figure 5-2-20, it can be said that OpenSUSE utilising TCP on IPv4 on smaller frame sizes ranging from 1518 to 3392 Bytes and on frame sizes 5266, 6203 and the largest frame size 9014 Bytes outperformed OpenSUSE utilising UDP on IPv4 by up to 21.8% on frame size 8037 Bytes in particular.

Based on the graph presented in Figure 5-2-20 and the analysis outlined above, it can be concluded that all of the operating systems involved in this study showed a very similar pattern in terms of their performance on a network utilising Jumbo frames on IPv6. Apart from Microsoft Windows Server 2003, this performance showed that TCP outperformed UDP on all of the operating systems and on most of the frame sizes except for frame sizes 4329, 7140 and 8037 Bytes where UDP outperformed TCP. Microsoft Windows Server 2003 on all of the frame sizes TCP outperformed UDP except for frame size 4329 Bytes where UDP outperformed TCP by 53.8%.

Based on the graph presented in Figure 5-2-19 and the analysis outlined above, it can also be concluded that all of the operating systems involved in this study showed a very similar pattern in terms of their performance on a network utilising Jumbo frames on IPv6.
on frame size 4329 Bytes where their performance in terms of throughput dropped by up to 78% when compared to the performance of UDP on IPv6.

Section 5-2-15 next presents the comparison of throughput for the two Internet protocols (IPv6 & IPv4) employed by this study on Jumbo frames. This the results from the experiments conducted on various operating systems using TCP.

5.2.15 Compare Jumbo Frames IPv6 & IPv4 Throughput on TCP

This section presents in line graphs a comparison of throughput results on IPv6 and IPv4 using TCP as transmission protocol employed by this study. This result was an outcome of experiments conducted on various operating systems utilising Jumbo frames.

![Figure 5-2-21: Comparison of Jumbo frames IPv6 & IPv4 throughput on TCP.](image)

The following list compares Jumbo frames IPv6 and IPv4 throughput results on a network employing TCP from the experiments conducted on all of the operating systems as presented on Figure 5-2-21:

- Based on the graph presented in Figure 5-2-21, it can be said that Microsoft Windows Server 2008 utilising TCP on IPv4 on frame sizes ranging from 1518, 3392, 5266, 7140 Bytes and the largest frame size 9014 Bytes outperformed Microsoft Windows Server 2008 utilising IPv6 by up to 1.8% on the smallest frame size.
- Based on the graph presented in Figure 5-2-21, it can be said that Microsoft Windows Server 2008 utilising IPv6 on frame size 6203 Bytes outperformed IPv4 by up to 2.4% while IPv6 and IPv4 on frame size 3392 yielded similar values in terms of throughput.
- Based on the graph presented in Figure 5-2-21, it can be said that Microsoft Windows 7 utilising TCP on IPv4 on frame sizes 1518, 7140 and 9014 Bytes outperformed IPv6 by up to 2.6% on the smallest frame size while on frame size 6203 Bytes IPv6 outperformed...
IPv4 by 6.2% however, on frames sizes 2455, 3392, 5266 and 8037 Bytes, IPv6 and IPv4 produced similar values in terms on throughput.

- Based on the graph presented in Figure 5-2-21, it can be said that Microsoft Windows Server 2003 utilising TCP with IPv6 on frame sizes ranging from 1518 to 3392 Bytes and 5266, 6203 and 9014 Bytes outperformed by up to 9% while on frame sizes 7140 and 8037 Bytes, IPv4 outperformed IPv6 by 12.4%.

- Based on the graph presented in Figure 5-2-21, it can be said that Linux Fedora utilising TCP on IPv4 on the smallest frame size 1518 Bytes and frame sizes 5266 and 7140 Bytes outperformed IPv6 by up to 1.6% on frame size 7140 Bytes however, IPv6 on the largest frame size 9014 Bytes outperformed IPv4 by 2.8% while on frame sizes 2455, 3392, 5266 and 6203 Bytes produced similar values in terms of throughput.

- Based on the graph presented in Figure 5-2-21, it can be said that Linux Ubuntu utilising TCP on frame sizes ranging from 1518 to 3392 Bytes and frame sizes 5266 and 9014 Bytes IPv6 and IPv4 produced similar values with regards to throughput while on frame sizes ranging from 6203 to 8037 Bytes IPv6 outperformed IPv4 by up to 2.8%.

- Based on the graph presented in Figure 5-2-21, it can be said that OpenSUSE utilising TCP on smaller frame sizes ranging from 1518 to 3392 Bytes and on frame sizes 5266, 6203 and the largest frame size 9014 Bytes produced similar throughput values while IPv4 outperformed IPv6 on frame size 7140 Bytes by 1.2% and IPv6 outperformed IPv4 by 0.8% on frame size 8037 Bytes.

- Based on the graph presented in Figure 5-2-21 and the analysis outlined above, it can be concluded that for the Microsoft Windows operating systems, on Microsoft Windows Server 2008 and Microsoft Windows 7, the smallest and the largest frame size IPv4 outperformed IPv6 while on frame 6203 Bytes IPv6 outperformed IPv4 however, the rest of the frames sizes both IPv6 and IPv4 produced very similar values in terms of throughput.

- Based on the graph presented in Figure 5-2-21 and the analysis outlined above, it can also be concluded that operating system Microsoft Windows Server 2003 frame sizes ranging from 1518 to 3392 Bytes and frame sizes 5266, 6203 and 9014 Bytes IPv6 outperformed IPv4 while frame sizes 7140 and 8037 Bytes IPv4 outperformed IPv6.

- Based on the graph presented in Figure 5-2-21 and the analysis outlined above, it can also be concluded that operating system Linux Fedora and Ubuntu frame sizes ranging from 2455 to 3392 Bytes and frame sizes 5266, 6203 IPv6 and IPv4 produced similar values in terms of throughput however, on frame size 1518, 7140 Bytes on Fedora IPv4 outperformed IPv6 and on the two largest frame sizes 8037 and 9014 Bytes IPv6 outperformed IPv4.

- Based on the graph presented in Figure 5-2-21 it can be concluded that for the frame size 4329 Bytes IPv6 and IPv4 on TCP Jumbo frames produced similar throughput
values however it was up to 86% lower when compared to Fedora on frame size 7140 Bytes.

Following in Figure 5-2-22 below presents a comparison of the Jumbo frames IPv6 and IPv4 throughput from experiments conducted on various operating systems using UDP as the transmission protocol.

![Compare IPv6 & IPv4 Jumbo Frames Throughput on Various OS on UDP](image)

Figure 5-2-22: Comparison of Jumbo frames IPv6 & IPv4 throughput on UDP

The following list compares Jumbo frames IPv6 and IPv4 throughput results on a network employing UDP from the experiments conducted on all of the operating systems as presented on Figure 5-2-22.

- Based on the graph presented in Figure 5-2-22, it can be said that Microsoft Windows Server 2008 utilising UDP on IPv6 on frame sizes 1518, 2455, 4329 Bytes and ranging from 6203 to 8037 Bytes outperformed IPv4 by up to 10.6% on the smallest frame size however, IPv4 on frame sizes 3392 and 9014 Bytes outperformed IPv6 by up to 8.6% on frame size 3392 Bytes while IPv6 and IPv4 on frame size 5266 Bytes yielded similar throughput values.

- Based on the graph presented in Figure 5-2-22, it can be said that Microsoft Windows 7 utilising UDP on IPv4 on frame sizes 1518 Bytes and frame sizes ranging from 3392 to 5266 Bytes and also 7140 Bytes outperformed IPv6 by up to 5.8% on frame size 3392 Bytes however, IPv6 on frame sizes 2455, 6203, 8037 and 9014 Bytes outperformed IPv4 by up to 5% on the largest frame size 9014 Bytes.

- Based on the graph presented in Figure 5-2-22, it can be said that Microsoft Windows Server 2003 utilising UDP on IPv6 on frame sizes ranging from 1518 Bytes to 8037 outperformed IPv4 by up to 29% on frame size 3392 Bytes however, IPv4 on the largest frame size 9014 Bytes outperformed IPv6 by 3.2%.
Based on the graph presented in Figure 5-2-22, it can be said that Linux Fedora utilising UDP on IPv6 on the smallest frame size 1518 Bytes outperformed IPv4 by 7.2% while on frame sizes 2455 and 3392 Bytes IPv6 and IPv4 produced similar throughput values however IPv4 on frame sizes ranging from 4329 to the largest frame size 9014 Bytes outperformed IPv6 by 25.8% on frame size 8037 Bytes in particular.

Based on the graph presented in Figure 5-2-22, it can be said that Linux Ubuntu utilising UDP on frame sizes ranging from 2455 Bytes to 4329 and frame size 7140 Bytes IPv6 and IPv4 yielded similar throughput values while IPv6 on the smallest frame size 1518 Bytes and 5266 Bytes outperformed IPv4 by 1.2% however, on frame sizes 6203, 8037 and 9014 Bytes IPv4 outperformed IPv6 by 4.8% on frame size 6203 Bytes in particular.

Based on the graph presented in Figure 5-2-22, it can be said that OpenSUSE utilising UDP on IPv6, on the smallest frame sizes 1518 Bytes and 5266 Bytes outperformed IPv4 by 1.2% while on frame sizes 2455, 3392, 6203 and 9014 Bytes IPv4 outperformed IPv6 by 4% however, on frame sizes 4329, 7140 and 8037 Bytes IPv6 and IPv4 yielded similar throughput values.

5.3 Applications Analysis

This section is concerned with the analysis of all results from the experiments conducted on applications such as DNS, Games and Voice over IP (VoIP) using Jumbo frames. Results in discussions also based on the chosen metric discussed in chapter 2, section 2.1.1, which are throughput, delay and jitter. The CPU utilisations and the packets dropped were also measured in the experiments so, this is also included in the analysis.

5.3.1 DNS Throughputs on IPv6 & IPv4 Jumbo Frames

This section presents the throughput results for the experiments conducted on a network employing Jumbo frames on IPv6. Figure 5-3-1 below presents the Jumbo frames throughput results of experiments conducted on various operating systems implemented on a network employing IPv6.

![DNS IPv6 Throughputs on Jumbo Frames](image1)

![DNS IPv4 Throughputs on Jumbo Frames](image2)

- Microsoft Windows Server 2008 and Server
- Linux Fedora produced the highest values in...
2003 were very similar in terms of throughput values.
- Linux Ubuntu was 2% less in terms of throughput values than Microsoft Windows Server 2008.
- OpenSUSE on the other hand generated 4% less throughput than Microsoft Windows Server 2008.
- Microsoft Windows 7 generated 8% less throughput when compared to that of Microsoft Windows Server 2008.
- Linux Fedora was lowest in terms of throughput values where it produced 11% less throughputs than Microsoft Windows Server 2008.
- Microsoft Windows Server 2008 was the second highest with regards to throughput values in which 2% lower than Fedora.
- Linux Ubuntu produced 9% lower throughput values than Fedora.
- OpenSUSE produced the lowest throughput values which 12% lower than Fedora.
- Microsoft Windows Server 2003 and Microsoft Windows 7 produced very similar values in terms of throughput values however, 1% lower than Ubuntu but 10% lower than Fedora.

Section 5-3-2 next presents the comparison of Jumbo frames DNS IPv6 and IPv4 throughput results from the experiments conducted on various operating systems

### 5.3.2 Compare DNS IPv6 & IPv4 Throughput results on Jumbo Frames

This section presents in bar graphs a comparison of IPv6 and IPv4 throughput results as presented on Figure 5-3-1 and Figure 5-3-2 for all of the six operating systems involved in this study using Jumbo frames.

![Figure 5-3-4: Compare DNS IPv6 & IPv4 throughputs on Jumbo frames](image)

- Based on the graph presented on Figure 5-3-4, it can be said that DNS on IPv6 and IPv4 on Microsoft Windows Server 2008 produced very similar values with regards to throughput.
- Based on the graph presented on Figure 5-3-4, it can be said that DNS on IPv6 and IPv4 on Microsoft Windows 7 produced very similar values with regards to throughput however, 8% lower than Microsoft Windows Server 2008.
Based on the graph presented on Figure 5-3-4, it can be said IPv6 on Microsoft Windows Server 2003 yields 8% more values in terms of throughput than IPv4.

Based on the graph presented on Figure 5-3-4, it can be said that IPv4 on Linux Fedora yields 13% more values in terms of throughput than IPv6.

Based on the graph presented on Figure 5-3-4, it can be said that IPv6 on Linux Ubuntu yields 5% more values in terms of throughput than IPv4.

Based on the graph presented on Figure 5-3-4, it can be said that IPv6 on OpenSUSE produced 6% more throughput values than IPv4.

Section 5-3-3 next presents the delay results of Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.

### 5.3.3 DNS Delay in IPv6 & IPv4 Jumbo Frames

This section presents the delay results for the experiments conducted on a network employing Jumbo frames on IPv6. Figure 5-3-5 presents the Jumbo frames delay results of experiments conducted on various operating systems implemented on a network employing IPv6.

**Figure 5-3-5: DNS IPv6 Delay on Jumbo frames**

- Microsoft Windows Server 2008 created up to a maximum of 64% more delays than any of the operating systems involved in this study.
- Microsoft Windows Server 2003 generated less delay than any of the operating systems involved.
- Microsoft Windows 7 generated 45% less delay values than Microsoft Windows Server 2008.
- Linux Fedora created 41% less delay when compared to Microsoft Windows Server 2008 which had the highest delay values than any of the operating systems involved in this study.

**Figure 5-3-6: DNS IPv4 delay on Jumbo frames**

- Microsoft Windows Server 2008 created more delays than any other operating systems involved in this study.
- Linux Fedora was the second highest in terms of delay values however, Fedora created 24% less delay when compared with Microsoft Windows Server 2008.
- Microsoft Windows Server 2003 created 30% less delay than Microsoft Windows Server 2008 however, only 6% less than Linux Fedora.
- Linux Ubuntu generated less delay than the previous operating systems however, 8%
- Linux Ubuntu created 50% less delay values than Microsoft Windows Server 2008 and 9% when compared to Linux Fedora which had the highest in terms of delay values among all of the Linux distributions that employed by this study.
- OpenSUSE had 3% less delay than Fedora but 44% less delay values when compared to Microsoft Windows Server 2008 which had the highest in terms of delay values.

Section 5-3-4 next presents the comparison of Jumbo frames DNS IPv6 and IPv4 delay results from the experiments conducted on various operating systems

**5.3.4 Compare DNS IPv6 & IPv4 Delay results on Jumbo Frames**

This section presents in bar graphs a comparison of IPv6 and IPv4 delay results as presented on Figure 5-3-5 and Figure 5-3-6 for all of the six operating systems involved in this study employing Jumbo frames.

- Based on the graph presented on Figure 5-3-7, it can be said that DNS on IPv6 and IPv4 on Microsoft Windows Server 2008 produced highest values with regards to delay however, IPv6 generated 8% higher delay values than IPv4.
- Based on the graph presented on Figure 5-3-7, it can be said that DNS on IPv4 on Microsoft Windows 7 generated 3% more delay than IPv6.
- Based on the graph presented on Figure 5-3-7, it can be said that DNS on IPv4 on Microsoft Windows Server 2003 created 26% more delay than IPv6.
- Based on the graph presented on Figure 5-3-7, it can be said that DNS on IPv4 on Linux Fedora created 9% more delay than IPv6.
- Based on the graph presented on Figure 5-3-7, it can be said that DNS on IPv4 on Linux Ubuntu created 10% more delay than IPv6.
- Based on the graph presented on Figure 5-3-7, it can be said that DNS on IPv4 on OpenSUSE created 1% more delay than IPv6.
- Based on the analysis outlined above it can be said that IPv4 created more delays than IPv6 when employed by Microsoft Windows 7 and Server 2003 and the three Linux operating systems involved in this study however, on Microsoft Windows Server 2008, IPv6 created more delays than IPv4.

Section 5-3-5 next presents the jitter results of Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.

**5.3.5 DNS Jitter on IPv6 & IPv4 Jumbo Frames**

This section presents the jitter results for the experiments conducted on a network employing Jumbo frames on IPv6. Figure 5-3-8 and Figure 5-3-9 presents the Jumbo frames jitter results of experiments conducted on various operating systems implemented on a network employing IPv6 and IPv4.

- **Figure 5-3-8: DNS IPv6 Jitter on Jumbo frames**
  - Microsoft Windows Server 2008 and Linux Fedora created the highest values in terms of jitter than any of the operating systems involved in this study
  - Microsoft Windows 7 and OpenSUSE were the second highest in terms of jitter values
  - Linux Ubuntu created less jitter than Microsoft Windows 7 and OpenSUSE
  - Microsoft Windows Server 2003 generated the lowest in terms of jitter

- **Figure 5-3-9: DNS IPv4 Jitter on Jumbo frames**
  - OpenSUSE produced the highest values in terms of jitter while Ubuntu was the second highest
  - All of the Microsoft Windows operating systems produced very similar values with regards to jitter which lower than OpenSUSE and Ubuntu
  - Linux Fedora produced the lowest values in terms of jitter among all the operating systems involved in this study
Section 5-3-6 next presents the comparison of Jumbo frames DNS IPv6 and IPv4 jitter results from the experiments conducted on various operating systems.

### 5.3.6 Compare DNS IPv6 & IPv4 Jitter results on Jumbo Frames

This section presents in bar graphs a comparison of IPv6 and IPv4 jitter results as presented on Figure 5-3-8 and Figure 5-3-9 for all of the six operating systems involved in this study employing Jumbo frames.

![Compare DNS IPv6 & IPv4 Jitter on Jumbo Frames](image_url)

- Based on the graph presented on Figure 5-3-10, it can be said that DNS on IPv6 on Microsoft Windows Server 2008 produced higher values with regards to jitter than IPv4.
- Based on the graph presented on Figure 5-3-10, it can be said that DNS on IPv6 and IPv4 on Microsoft Windows 7 produced similar values with regards to jitter.
- Based on the graph presented on Figure 5-3-10, it can be said that DNS on IPv4 on Microsoft Windows Server 2003 produced higher values with regards to jitter than IPv6.
- Based on the graph presented on Figure 5-3-10, it can be said that DNS on IPv6 on Linux Fedora produced higher jitter values than IPv4.
- Based on the graph presented on Figure 5-3-10, it can be said that DNS on IPv4 on Linux Ubuntu produced more jitter than IPv6.
- Based on the graph presented on Figure 5-3-10, it can be said that DNS on IPv4 on OpenSUSE produced more jitter than IPv6.
- Based on the analysis outlined above it can be said that IPv4 created more jitter than IPv6 when employed by Server 2003 and Linux Ubuntu and OpenSUSE however, on Microsoft Windows Server 2008 and Linux Fedora, IPv6 created more jitter than IPv4 while Microsoft Windows 7 IPv6 and IPv4 generated similar amounts of jitter.

Section 5-3-7 next presents the CPU utilisations results of Jumbo frames IPv6 and IPv4 on Router 1 from the experiments conducted on various operating systems.
### 5.3.7 DNS CPU Usage on Router 1 on IPv6 & IPv4 Jumbo Frames

This section presents the DNS IPv6 and IPv4 CPU utilisations results on Router 1 for the experiments conducted on a network employing Jumbo frames.

Figure 5-3-11 and Figure 5-3-12 below presents the Jumbo frames DNS CPU utilisations results on Router 1 of experiments conducted on various operating systems implemented on a network employing IPv6 and IPv4.

- IPv6 on Microsoft Windows Server 2003 showed the lowest CPU usage values among all of the operating systems involved.
- IPv6 on Microsoft Windows Server 2008 held the second lowest values in terms of CPU utilisations in which 0.9% higher than Microsoft Windows Server 2003.
- Linux Ubuntu yielded the highest values in terms of CPU utilisations.
- Linux Fedora was the second highest which 3% lower than Ubuntu which was the highest.
- OpenSUSE required 4% less of the CPU time than Ubuntu.
- Among the Microsoft Windows operating systems involved in this study, Microsoft Windows 7 was the highest in terms of CPU utilisations which 2% higher than Microsoft Windows Server 2008 but 15% lower than Ubuntu.

---

**Figure 5-3-11: DNS IPv6 CPU usage on Router 1**

- Server 2008
- Windows 7
- Server 2003
- Fedora 12
- Ubuntu 10.10
- OpenSUSE 11.3

**Figure 5-3-12: DNS IPv4 CPU usage on Router 1**

- Microsoft Windows Server 2003 demonstrated to be the lowest in terms of CPU utilisations.
- Linux Ubuntu demonstrated to be the operating system required the CPU resources the most which 20% more than Microsoft Windows Server 2003.
- On IPv4 Microsoft Windows 7 required 18% less of the CPU time than Ubuntu.
- Linux Fedora required 10% less of the CPU time than Ubuntu.
- OpenSUSE required 4% less of the CPU resources the Ubuntu.
- Microsoft Windows Server 2008 required 1% more of the CPU time than Microsoft Windows Server 2003.
Section 5-3-8 next presents the comparison of Jumbo frames DNS IPv6 and IPv4 CPU utilisations on Router 1 results from the experiments conducted on various operating systems.

5.3.8 Compare DNS IPv6 & IPv4 Usage results on Router 1 on Jumbo Frames

This section presents in bar graphs a comparison of IPv6 and IPv4 CPU utilisations results on Router 1 as presented on Figure 5-3-11 and Figure 5-3-12 for all of the six operating systems involved in this study employing Jumbo frames.

![Compare DNS IPv6 & IPv4 CPU Usage on Router 1](image)

*Figure 5-3-13: Compare DNS IPv6 & IPv4 on Jumbo frames CPU utilisations on Router 1*

- Based on the graph presented on Figure 5-3-13, it can be said that IPv4 on Microsoft Windows Server 2008 used 0.5% more of the CPU resources than IPv6.
- Based on the graph presented on Figure 5-3-13, it can be said that IPv6 and IPv4 demonstrated similar values in terms of CPU utilisations however, 2% more than Microsoft Windows Server 2008.
- Based on the graph presented on Figure 5-3-13, it can be said that Microsoft Windows Server 2003 produced the lowest values with regards to CPU utilisations which 2% lower than Microsoft Windows 7 and similar to Microsoft Windows Server 2008.
- Based on the graph presented on Figure 5-3-13, it can be said that IPv6 on Linux Fedora required 4% more of the CPU time than IPv4.
- Based on the graph presented on Figure 5-3-13, it can be said that IPv6 and IPv4 on Linux Ubuntu required the CPU time the most when compared to all of the operating systems involved in this study however, IPv4 required 3% more of the CPU time than IPv6.
- Based on the graph presented on Figure 5-3-13, it can be said that IPv4 on OpenSUSE required 3% more of the CPU time than IPv6.
- Based on the analysis outlined above it can be said that IPv4 on Microsoft Windows Server 2008, Linux Ubuntu and OpenSUSE required more of the CPU time than IPv6. IPv6 and IPv4 on Microsoft Windows Server 2003 and Microsoft Windows 7 utilised the CPU time equally however, Microsoft Windows 7 utilised the CPU time 2% more than
Microsoft Windows Server 2003. IPv6 on Linux Fedora required more of the CPU resources that IPv4 however, 4% less than IPv6 on Ubuntu.

Section 5-3-9 next presents the CPU utilisations results of Jumbo frames IPv6 and IPv4 on Router 2 from the experiments conducted on various operating systems.

5.3.9 DNS CPU Usage on Router 2 on IPv6 & IPv4 Jumbo Frames
This section presents the DNS IPv6 and IPv4 CPU utilisations results on Router 2 for the experiments conducted on a network employing Jumbo frames. Figure 5-3-14 and Figure 5-3-15 below presents the Jumbo frames DNS CPU utilisations results on Router 2 of experiments conducted on various operating systems implemented on a network employing IPv6 and IPv4.

Figure 5-3-14: DNS IPv6 CPU usage on Router 2
- On router 2, Microsoft Windows Server 2008 and Microsoft Windows Server 2003 were measured to be the lowest in terms of CPU utilisations.
- Microsoft Windows 7 required 2% more of the CPU time than Microsoft Windows Server 2008.
- Linux Fedora required 11% more of the CPU time than Microsoft Windows 7.
- Linux Ubuntu was measured with the highest values in terms of CPU utilisations which 7% more than Fedora but 16% more than Microsoft Windows 7.
- OpenSUSE was 11% lower than Ubuntu but 4% lower than Fedora however it was measured 5% higher than Microsoft Windows 7 in terms of CPU utilisations

Figure 5-3-15: DNS IPv4 CPU usage on Router 2
- On router 2, Microsoft Windows Server 2008 and Microsoft Windows Server 2003 were measured to be the lowest in terms of CPU utilisations
- Microsoft Windows 7 required 11% more of the CPU time than Microsoft Windows Server 2008
- Linux Fedora required 1% less than Microsoft Windows 7 and 11% more of the CPU time than Microsoft Windows Server 2003
- Linux Ubuntu was measured with the highest values in terms of CPU utilisations which 10% more than Fedora but 12% more than Microsoft Windows 7
- OpenSUSE was 13% lower than Ubuntu but 1% lower than Fedora however it was measured 8% higher than Microsoft Windows Server 2008 in terms of CPU utilisations

Section 5-3-10 next presents the comparison of Jumbo frames DNS IPv6 and IPv4 CPU utilisations on Router 2 results from the experiments conducted on various operating systems.
**5.3.10 Compare DNS IPv6 & IPv4 CPU Usages on Router 2 on Jumbo Frames**

This section presents in bar graphs a comparison of IPv6 and IPv4 CPU utilisations results on Router 2 as presented on Figure 5-3-14 and Figure 5-3-15 for all of the six operating systems involved in this study employing Jumbo frames.

- Based on the graph presented on Figure 5-3-16, it can be said that in terms of CPU utilisations, Microsoft Windows Server 2008 and Microsoft Windows Server 2003 were measured with the lowest values in which IPv6 and IPv4 produced similar percentages with regards to CPU utilisations.
- Based on the graph presented on Figure 5-3-16, it can be said that IPv6 on Linux Fedora required 2% more of the CPU resources than IPv4.
- Based on the graph presented on Figure 5-3-13, it can be said that IPv4 on Linux Ubuntu was measured with the highest percentages in terms of CPU utilisations which was 5% more than IPv6 and 11% more than IPv6 on Fedora which was measured to be the second highest.
- Based on the graph presented on Figure 5-3-13, it can be said that IPv4 OpenSUSE required 2% more of the CPU time than IPv6 while IPv4 on Microsoft Windows 7 required 9% more of the CPU time than IPv6.
- Based on the graph presented on Figure 5-3-13, it can be concluded that IPv6 and IPv4 on Microsoft Windows Server 2008 and Server 2003 required similar amount of the CPU resources while IPv4 on Microsoft Windows 7 and Linux Ubuntu and OpenSUSE required more than IPv6 however, IPv6 on Linux Fedora required more of the CPU time than IPv4.

Section 5-3-11 next presents the games throughput results of Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.
5.3.11 Games Throughput Results on IPv6 & IPv4 Jumbo Frames

This section presents the throughput results of all the games involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-17 below presents the games throughput results of experiments conducted on various operating systems implemented on a network employing IPv6.

![Games Throughput on Jumbo Frames IPv6](image)

- Based on the graph presented on Figure 5-3-17, it can be said that Linux Ubuntu produced the highest values in terms of throughput for Counter Strike A (CSa) on IPv6.
- Based on the graph presented on Figure 5-3-17, it can be said that, Microsoft Windows Server 2003 was the second highest in terms of throughput values which was 6% less than Ubuntu with regards to throughput values.
- Based on the graph presented on Figure 5-3-17, it can be said that, Microsoft Windows Server 2008 was 23% less than Ubuntu with regards to throughput values which Microsoft Windows 7, Linux Fedora and OpenSUSE produced similar values with regards to throughput.
- Based on the graph presented on Figure 5-3-17, it can be said that, throughput values for Counter Strike I (CSI), all of the operating systems involved in this study produced similar values.
- Based on the graph presented on Figure 5-3-17, it can be said that, for the game Quake3, throughput values was higher than CSa and CSI. All of the operating systems involved in this study produced very similar values in terms of throughput however; Microsoft Windows Server 2008 was higher by 14% then the rest.

Figure 5-3-18 below presents the games throughput results of experiments conducted on various operating systems implemented on a network employing IPv4.
Based on the graph presented on Figure 5-3-18, it can be said that Microsoft Windows Server 2008 produced the highest values in terms of throughput for CSa on IPv4. Based on the graph presented on Figure 5-3-18, it can be said that, Microsoft Windows 7 was the second highest in terms of throughput values which was 1.5% less than Microsoft Windows Server 2008 with regards to throughput values. Based on the graph presented on Figure 5-3-18, it can be said that, Microsoft Windows Server 2003 was the lowest which was 1.5% less than Microsoft Windows Server 2008 with regards to throughput values. Based on the graph presented on Figure 5-3-18, it can be said that, throughput values for CSi, all of the operating systems involved in this study produced similar values. Based on the graph presented on Figure 5-3-18, it can be said that, for the game Quake3, throughput values was higher than CSa and CSi. All of the operating systems involved in this study produced very similar values in terms of throughput however, OpenSUSE was slightly higher by 0.1% than the rest of the operating systems.

Section 5-3-12 next presents the comparison of Jumbo frames games IPv6 and IPv4 throughput results from the experiments conducted on various operating systems

**5.3.12 Compare Games Throughput Results on IPv6 & IPv4 Jumbo Frames**

This section presents in bar graphs a comparison of games throughput results on IPv6 and IPv4 as presented on Figure 5-3-19 and Figure 5-3-20 for all of the six operating systems involved in this study employing Jumbo frames.
The following list compares games throughput on Jumbo frames utilising IPv6 and IPv4 from the experiments conducted on all of the operating systems as presented on Figure 5-3-18.

- Based on the graph presented in Figure 5-3-19, it can be said that IPv4 on Microsoft Windows Server 2008 utilising IPv4 on CSa game yielded more throughput values than IPv6 however, on CSa game, IPv4 on Microsoft Windows Server 2008 was the highest in terms of throughput values which 38.2% higher than IPv6.

- Based on the graph presented in Figure 5-3-19, it can be said that IPv6 on Microsoft Windows 7 and IPv4 on Microsoft Windows Server 2003, IPv6 on Linux Fedora, IPv4 on Linux Ubuntu, IPv6 and IPv4 on OpenSUSE, they all yielded the same value in terms of throughput which 41.8% less than Microsoft Windows Server 2008 IPv4 on game CSa.

- Based on the graph presented in Figure 5-3-19, it can be said that IPv4 on Microsoft Windows 7 was the second highest in terms of throughput values however; it was 30.4% less than IPv4 on Microsoft Windows Server 2008.

- Based on the graph presented in Figure 5-3-19, it can be said that IPv6 on Microsoft Windows Server 2003 and Linux Ubuntu, they were 5.8% less when compared to Microsoft Windows 7 in terms of throughput values.

- Based on the graph presented in Figure 5-3-19, it can be said that all of the operating systems involved in this study yielded similar values in terms of throughput for the game CSi on both IPv6 and IPv4.

- Based on the graph presented in Figure 5-3-19, it can be said that for the game Quake3, IPv6 on all of the operating systems involved outperformed IPv4 with regards to throughput however, IPv6 on Microsoft Windows Server 2008 yielded the highest values for the game Quake3 which 5% more than Linux Fedora which was the second highest.

- Based on the graph presented in Figure 5-3-19, it can be concluded that for the CSa game, IPv4 for Microsoft Windows Server 2008, Microsoft Windows 7 and Linux Fedora produced higher throughput values than IPv6.
Based on the graph presented in Figure 5-3-19, it can be concluded that IPv6 and IPv4 on all of the operating systems involved in this study on game CSi yielded similar values with regards to throughput however, throughput values on games CSi was the lowest out of all the games engaged in this study for instance, for Microsoft Windows Server 2008 IPv6, throughput values for CSi was 7.6% less than CSA and 74.9% less than Quake3.

Based on the graph presented in Figure 5-3-19, it can be concluded that IPv6 for all of the operating systems involved in this study on the game Quake3 yielded the highest throughput values out of all the games tested in this study.

Section 5-3-13 next presents the games delay results of Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.

5.3.13 Games Delay Results on IPv6 & IPv4 Jumbo Frames

This section presents the delay results of all the games involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-20 below presents the games delay results of experiments conducted on various operating systems implemented on a network employing IPv6.

This section presents the delay results of all the games involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-20 above also presents the games delay results from experiments conducted on various operating systems implemented on a network employing IPv6.

- Based on the graph presented in Figure 5-3-20, it can be said that Microsoft Windows Server 2008 created the highest amount of delay out of all the operating systems engaged in this study on all of the games involved.
- Based on the graph presented in Figure 5-3-20, it can be said that Microsoft Windows Server 2003 had the lowest result in terms of delay values which 63% lower than Microsoft Windows Server 2008 on all of the games involved.
Based on the graph presented in Figure 5-3-20, it can be said that, Microsoft Windows 7 created similar amount of delay on all of the games involved in this study which 44% less than Microsoft Windows Server 2008 and 19% higher than Microsoft Windows Server 2003.

Based on the graph presented in Figure 5-3-20, it can be said that, Linux Ubuntu created similar amount of delay on all of the games involved in this study which 50% lower than Microsoft Windows Server 2008 and 13% higher than Microsoft Windows Server 2003.

Based on the graph presented in Figure 5-3-20, it can be said that, Linux Fedora and OpenSUSE generated similar values in terms of delay on both CSa and CSi games however, they also generated similar values for the games Quake3 but slightly higher by 2% than the other two games.

Based on the graph presented in Figure 5-3-20, it can be concluded that, Microsoft Windows Server 2008 on IPv6 created the highest amount of delay on all of the games involved in this study while Microsoft Windows Server 2003 generated the lowest values in terms of delay.

Figure 5-3-21 below presents the games delay results from experiments conducted on various operating systems implemented on a network employing IPv4.

![Graph of Games Delay on Jumbo Frames IPv4](image)

**Figure 5-3-21: Games delay results on IPv4 Jumbo frames**

This section presents the delay results of all the games involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-21 above also presents the games delay results from experiments conducted on various operating systems implemented on a network employing IPv4.

Based on the graph presented in Figure 5-3-21, it can be said that Microsoft Windows Server 2008 created the highest amount of delay out of all the operating systems engaged in this study on all of the games involved.
Based on the graph presented in Figure 5-3-21, it can be said that, Linux Ubuntu generated the lowest amount of delay on the Quake3 game which 50% lower than Microsoft Windows Server 2008.

Based on the graph presented in Figure 5-3-21, it can be said that, Microsoft Windows 7 generated similar amount of delay on all the games involved in this study.

Based on the graph presented in Figure 5-3-21, it can be said that, Microsoft Windows Server 2003 generated similar amount of delay on games CSi and Quake3 however, it was slightly higher by 1% than the amount generated on the CSa game.

Based on the graph presented in Figure 5-3-21, it can be said that, Linux Fedora generated similar amount of delay on all of the games however, it was slightly higher by 2% on the Quake3 game and 41% lower than Microsoft Windows Server 2008 which makes Fedora the second highest with regards to delay values.

Based on the graph presented in Figure 5-3-21, it can be said that, OpenSUSE generated more delay on the Quake3 games however; it was less than 1% more than the amount of delay generated on the CSa and CSi games.

Based on the graph presented in Figure 5-3-21, it can be concluded that, Microsoft Windows Server 2008 on IPv4 created the highest amount of delay on all of the games involved in this study while Linux Ubuntu generated the lowest values in terms of delay on the Quake3 game.

Section 5-3-14 next presents the comparison of Jumbo frames games IPv6 and IPv4 throughput results from the experiments conducted on various operating systems

5.3.14 Compare Games Delay Results on IPv6 & IPv4 Jumbo Frames

This section presents in bar graphs a comparison of the amount of delay generated by the games involved in this study on IPv6 and IPv4 as presented on Figure 5-3-20 and Figure 5-3-21 for all of the six operating systems employing Jumbo frames.

![Figure 5-3-22: Compare Jumbo frames IPv6 and IPv4 delay on games](image-url)
The following list compares Jumbo frames IPv6 and IPv4 delay on games from the experiments conducted on all of the operating systems as presented on Figure 5-3-22 above.

- Based on the graph presented in Figure 5-3-22, it can be said that, IPv6 on Microsoft Windows Server 2008 on all of the games engaged in this study yielded 6% more delay than IPv4.
- Based on the graph presented in Figure 5-3-22, it can be said that, Microsoft Windows 7 IPv6 generated similar amount of delay on all of the games which was 3% less than IPv4.
- Based on the graph presented in Figure 5-3-22, it can be said that, Microsoft Windows Server 2003 IPv6 generated similar amount of delay on all of the games which was the lowest among all of the operating systems involved in this study which 25% less than IPv4 and 63% lower than Microsoft Windows Server 2008 which generated the highest in terms of delay.
- Based on the graph presented in Figure 5-3-22, it can be said that, Linux Fedora IPv6 generated similar amount of delay and also IPv4 however, IPv4 generated 10% more delay than IPv6.
- Based on the graph presented in Figure 5-3-22, it can be said that, IPv6 on Linux Ubuntu generated similar amount of delay on all of the games while IPv4 generated 11% more delay on CSa and CSi games however, on the Quake3 game IPv4 generated 7% less delay than IPv6.
- Based on the graph presented in Figure 5-3-22, it can be said that, OpenSUSE IPv6 generated similar amount of delay on all of the games while IPv4 on the CSa and Quake3 games had similar amount but on the CSi game, IPv4 generated 3% more in terms of delay.
- Based on the graph presented in Figure 5-3-22, it can be concluded that IPv6 on Microsoft Windows Server 2008 yielded the highest amount of delay values on all of the games tested in this study while IPv6 on Microsoft Windows Server 2003 generated the lowest. IPv4 on Microsoft Windows Server 2008 also produced the highest while Linux Ubuntu on the Quake3 game produced the lowest.

Section 5-3-15 next presents the games jitter results of Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.

**5.3.15 Games Jitter Results on IPv6 & IPv4 Jumbo Frames**

This section presents the jitter results of all the games involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-23 below
presents the games jitter results of experiments conducted on various operating systems implemented on a network employing IPv6.

<table>
<thead>
<tr>
<th>Name of Games</th>
<th>Games Jitter on Jumbo Frames IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Csa</td>
<td></td>
</tr>
<tr>
<td>Csi</td>
<td></td>
</tr>
<tr>
<td>Quake3</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5-3-23: Games Jitter results on IPv6 Jumbo frames**

This section presents the jitter results of all the games involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-23 above also presents the games jitter results from experiments conducted on various operating systems implemented on a network employing IPv6.

- Based on the graph presented in Figure 5-3-23, it can be said that Microsoft Windows Server 2008 created the highest amount of jitter out of all the operating systems engaged in this study on the CSa game while the rest of the operating systems produced similar amount of jitter which were less than 1% lower than Microsoft Windows Server 2008.

- Based on the graph presented in Figure 5-3-23, it can be said that OpenSUSE created the highest amount of jitter out of all the operating systems engaged in this study on the CSI game while the rest of the operating systems produced similar amount of jitter which were less than 1% lower than OpenSUSE.

- Based on the graph presented in Figure 5-3-23, it can be said that on the Quake3 game, all of the operating systems engaged in this study generated similar amount of jitter.

- Based on the graph presented in Figure 5-3-23, it can be concluded that on all of the games involved in this study, IPv6 on all of the operating systems generated less jitter on Quake3 than the rest of the games involved in this study.

Figure 5-3-24 below presents the games jitter results from experiments conducted on various operating systems implemented on a network employing IPv4.
This section presents the jitter results of all the games involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-24 above also presents the games jitter results from experiments conducted on various operating systems implemented on a network employing IPv4.

- Based on the graph presented in Figure 5-3-24, it can be said that Microsoft Windows Server 2008 and Microsoft Windows 7 created similar amount of jitter and less than the rest of all the operating systems engaged in this study on the CSa.
- Based on the graph presented in Figure 5-3-24, it can be said that all of the operating systems generated similar amount of jitter on the CSI game.
- Based on the graph presented in Figure 5-3-24, it can be said that all of the operating systems generated similar amount of jitter on the Quake3 game.
- Based on the graph presented in Figure 5-3-24, it can be concluded that on the Quake3 game, all of the operating systems engaged in this study generated similar amount in terms of jitter but the highest out of all the games involved. For the CSa and CSI games, all of the operating system generated similar amount of jitter but lower than Quake3 while Microsoft Windows Server 2008 and Microsoft Windows 7 created less jitter than the others on the CSa game.

Section 5-3-16 next presents the comparison of Jumbo frames games IPv6 and IPv4 jitter results from the experiments conducted on various operating systems

**5.3.16 Compare Games Jitter Results on IPv6 & IPv4 Jumbo Frames**

This section presents in bar graphs a comparison of the amount of jitter generated by the games involved in this study on IPv6 and IPv4 as presented on Figure 5-3-25 and Figure 5-3-26 for all of the six operating systems employing Jumbo frames.
The following list compares Jumbo frames IPv6 and IPv4 jitter on games from the experiments conducted on all of the operating systems as presented on Figure 5-3-25 above.

- Based on the graph presented in Figure 5-3-25, it can be said that IPv6 on Microsoft Windows Server 2008 on the CSa game produced the highest amount of jitter than IPv4 for all of the operating systems.

- Based on the graph presented in Figure 5-3-25, it can be said that IPv6 on Microsoft Windows 7 produced higher jitter than IPv4 however, IPv6 on Microsoft Windows 7 created similar amount of jitter than IPv6 and IPv4 on Microsoft Windows Server 2003, Linux Fedora, Ubuntu and OpenSUSE.

- Based on the graph presented in Figure 5-3-25, it can be said that IPv6 and IPv4 on all of the operating systems generated similar amount of jitter on the CSI game except for IPv6 on OpenSUSE which produced higher amount of jitter.

- Based on the graph presented in Figure 5-3-25, it can be said that on the Quake3 game, all of the operating systems on IPv6 generated less amount of jitter than IPv4.

- Based on the graph presented in Figure 5-3-25, it can be concluded that IPv6 on Microsoft Windows Server 2008 generated more jitter than IPv4 on the CSa game while IPv6 on OpenSUSE produced the higher jitter values than IPv4.

- Based on the graph presented in Figure 5-3-25, it can be concluded that IPv4 on the Quake3 game created more jitter the IPv6 on all of the operating systems.

Section 5-3-17 next presents the games CPU utilisations on Router 1 results of Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.
5.3.17 Games CPU utilisations Results on Router 1 IPv6 & IPv4 Jumbo Frames

This section presents the CPU utilisations on Router 1 results of all the games involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-26 below presents the games CPU utilisations on Router 1 results of experiments conducted on various operating systems implemented on a network employing IPv6.

Based on the graph presented in Figure 5-3-26, it can be said that Microsoft Windows Server 2008 on Router1 using IPv6 required similar amount of CPU resources on the games CSa and Quake3 while on CSI it required 8% more of the CPU time.

Based on the graph presented in Figure 5-3-26, it can be said that Microsoft Windows 7 on Router1 using IPv6 required similar amount of CPU resources on the games CSI and Quake3 while on CSa it required 3% more of the CPU time.

Based on the graph presented in Figure 5-3-26, it can be said that Microsoft Windows Server 2003 on Router 1 using IPv6 required similar amount of CPU resources on the games CSa and CSI while on Quake3 it required 5% more of the CPU time.

Based on the graph presented in Figure 5-3-26, it can be said that Linux Fedora on Router1 using IPv6 required similar amount of CPU resources on the games CSa and CSI while on Quake3 it required 3% less of the CPU time.

Based on the graph presented in Figure 5-3-26, it can be said that Linux Ubuntu on Router1 using IPv6 on the game CSI required 1% more of CPU resources than game CSa and 7% more than game Quake3.
Based on the graph presented in Figure 5-3-26, it can be said that Linux Ubuntu on Router1 using IPv6 on the game CSi required 4% more of CPU resources than game CSa and 1% more than game Quake3.

Based on the graph presented in Figure 5-3-26, it can be concluded that on a network employing Jumbo frames on games while using IPv6, OpenSUSE required more CPU resources than any of the operating systems involved in this study especially on the game CSi which is 4% more than Linux Ubuntu which was the second highest. OpenSUSE was also the highest in terms of CPU utilisations for the game Quake3 while had similar value with regards to CPU utilisations with Linux Ubuntu on the game CSa which again the highest.

Figure 5-3-27 below presents the games CPU utilisations results from experiments conducted on various operating systems implemented on a network employing IPv4 on Router 1.

Figure 5-3-27: Games CPU utilisations on Router1 results for IPv4 Jumbo frames

This section presents the CPU utilisations results of all the games involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-27 above also presents the games CPU utilisations results from experiments conducted on various operating systems implemented on a network employing IPv4 on Router 1.

Based on the graph presented in Figure 5-3-27, it can be said that Microsoft Windows Server 2008 on Router1 using IPv4 on the games CSa required 1% more of the CPU time than CSi and 1% less than Quake3.

Based on the graph presented in Figure 5-3-27, it can be said that Microsoft Windows7 on Router1 using IPv4 on the games CSa required 18% more of the CPU time than CSi and 17% less than Quake3.
- Based on the graph presented in Figure 5-3-27, it can be said that Microsoft Windows Server 2003 on Router1 using IPv4 required similar of the CPU time on all of the games involved in this study.
- Based on the graph presented in Figure 5-3-27, it can be said that Linux Fedora on Router1 using IPv4 on the games CSi and Quake3 required similar amount of the CPU time which 3% less than CSa.
- Based on the graph presented in Figure 5-3-27, it can be said that Linux Ubuntu on Router 1 using IPv4 on the games CSi and Quake3 required similar amount of the CPU time which 3% less than CSa.
- Based on the graph presented in Figure 5-3-27, it can be said that Linux Ubuntu on Router 1 using IPv4 on the games CSi and Quake3 required similar amount of the CPU time which 4% less than CSa.
- Based on the graph presented in Figure 5-3-27, it can be concluded that on a network employing Jumbo frames on games while using IPv4, Microsoft Windows 7 required the CPU time more than any of the operating systems involved in this study which was 4% more than Linux Ubuntu which was the second highest on the game CSa.
- Based on the graph presented in Figure 5-3-27, it can be concluded that on a network employing Jumbo frames on games while using IPv4, Linux Ubuntu required the CPU time more than any of the operating systems involved in this study for the game CSi and Quake3 which was 5% more than OpenSUSE which was the second highest for both of the games.

Section 5-3-18 next presents the comparison of Jumbo frames games IPv6 and IPv4 CPU utilisations results from the experiments conducted on various operating systems

5.3.18 Compare Games CPU Usage Results on IPv6 & IPv4 Jumbo Frames on Router 1

This section presents in bar graphs a comparison of the amount of CPU utilisations on Router 1 as required by the games involved in this study on IPv6 and IPv4 as presented on Figure 5-3-26 and Figure 5-3-27 for all of the six operating systems employing Jumbo frames.
The following list compares Jumbo frames IPv6 and IPv4 CPU utilizations on games from the experiments conducted on all of the operating systems as presented on Figure 5-3-28 above.

- Based on the graph presented in Figure 5-3-28, it can be said that, IPv4 on Microsoft Windows 7 on the CSa game was the highest in terms of CPU utilizations for all of the operating systems and the games engaged in this study however, Microsoft Windows 7 IPv4 on the CSa game was 17% more than IPv6.
- Based on the graph presented in Figure 5-3-28, it can be said that, IPv4 on Linux Ubuntu on the CSa game was the second highest in terms of CPU utilizations which was 4% less than Microsoft Windows 7 and 3% more than Linux Ubuntu IPv6.
- Based on the graph presented in Figure 5-3-28, it can be said that Microsoft Windows Server 2003 yielded similar values which was the lowest in terms of CPU utilizations on both IPv6 and IPv4 on the CSa game which was 23% less than Microsoft Windows 7 IPv4.
- Based on the graph presented in Figure 5-3-28, it can be said that Microsoft Windows Server 2008 IPv4 yielded 2% more in terms of CPU utilizations than IPv6 on the CSa game but still 22% less than Microsoft Windows 7.
- Based on the graph presented in Figure 5-3-28, it can be said that, IPv6 on OpenSUSE on the CSi game was the highest in terms of CPU utilizations for all of the operating systems which was 8% more than IPv4.
- Based on the graph presented in Figure 5-3-28, it can be said that, Linux Ubuntu IPv6 and IPv4 on the CSi game was the second highest and produced similar values in terms of CPU utilizations which was 3% less than OpenSUSE IPv6 and 4% more than OpenSUSE IPv4.
- Based on the graph presented in Figure 5-3-28, it can be said that, IPv6 on Linux Fedora on the CSi game was 4% higher than IPv4 in terms of CPU utilisations but was 9% lower than OpenSUSE IPv6.

- Based on the graph presented in Figure 5-3-28, it can be said that Microsoft Windows Server 2003 yielded similar values which was the lowest in terms of CPU utilisations on both IPv6 and IPv4 on the CSi game which was 20% less than OpenSUSE IPv6.

- Based on the graph presented in Figure 5-3-28, it can be said that, IPv4 on Microsoft Windows 7 on the CSi game was 2% higher than IPv6 in terms of CPU utilisations but was 15% lower than OpenSUSE IPv6.

- Based on the graph presented in Figure 5-3-28, it can be said that, IPv6 on Microsoft Windows Server 2008 on the CSi game was 8% higher than IPv4 in terms of CPU utilisations but was 11% lower than OpenSUSE IPv6.

- Based on the graph presented in Figure 5-3-28, it can be said that, IPv6 on OpenSUSE IPv6 on the Quake3 game again was the highest in terms of CPU utilisations for all of the operating systems which was 7% higher than IPv4 and 2% higher than Linux Ubuntu IPv4 which was the second highest.

- Based on the graph presented in Figure 5-3-28, it can be said that, IPv4 on Linux Ubuntu on the Quake3 game was 5% higher than IPv6 in terms of CPU utilisations.

- Based on the graph presented in Figure 5-3-28, it can be said that, IPv6 on Linux Fedora on the Quake3 game was higher than IPv4 in terms of CPU utilisations however, it was less than 1%.

- Based on the graph presented in Figure 5-3-28, it can be said that, IPv6 on Microsoft Windows Server 2003 on the Quake3 game was 5% higher than IPv4 in terms of CPU utilisations but 14% less than OpenSUSE IPv6 which was the highest.

- Based on the graph presented in Figure 5-3-28, it can be said that, IPv4 on Microsoft Windows 7 on the Quake3 game was 3% higher than IPv4 in terms of CPU utilisations however, Microsoft Windows 7 IPv6 yielded similar values to that of IPv4 Microsoft Windows Server 2008 in terms of CPU utilisations.

- Based on the graph presented in Figure 5-3-28, it can be said that, IPv4 on Microsoft Windows Server 2008 on the Quake3 game was 3% higher than IPv4 in terms of CPU utilisations.

- Based on the graph presented in Figure 5-3-28, it can be concluded that on a network employing Jumbo frames on games while using IPv4, Microsoft Windows 7 required the CPU time more than any of the operating systems involved in this study for the game CSa which was 4% more than Linux Ubuntu IPv4 which was the second highest. IPv6 on OpenSUSE on the CSi and Quake3 games required more of the CPU time.

- Based on the graph presented in Figure 5-3-28, it can be concluded that IPv6 on Linux Ubuntu on the game CSa yielded similar values in terms of CPU utilisations to that of
Linux Ubuntu on the game CSi on both IPv6 and IPv4 which was 7% less than IPv4 Microsoft Windows 7 on CSa game.

Section 5-3-19 next presents the games CPU utilisations on Router 2 results of Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.

**5.3.19 Games CPU utilisations Results on Router 2 IPv6 & IPv4 Jumbo Frames**

This section presents the CPU utilisations on Router 2 results of all the games involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-29 below presents the games CPU utilisations on Router 2 results of experiments conducted on various operating systems implemented on a network employing IPv6.

![Games CPU Usage on Router 2 on Jumbo Frames IPv6](image)

- Based on the graph presented in Figure 5-3-29, it can be said that Microsoft Windows Server 2008 and Microsoft Windows Server 2003 on Router2 using IPv6 was the lowest in terms of CPU utilisations on all of the games.
- Based on the graph presented in Figure 5-3-29, it can be said that Microsoft Windows 7 on Router 2 using IPv6 was the highest in terms of the CPU utilisations on the game Quake3 which was 21% more than Microsoft Windows 7 on the CSi game and 25% more than CSa game.
- Based on the graph presented in Figure 5-3-29, it can be said that Linux Fedora on Router2 using IPv6 yielded 6% more values in terms of CPU utilisations on the games CSa than Microsoft Windows 7 and 2% more on the CSi game however, 21% less than Microsoft Windows 7 on the Quake3 game.
- Based on the graph presented in Figure 5-3-29, it can be said that Linux Ubuntu on Router 2 using IPv6 yielded the second highest values in terms of CPU utilisations on the all of the games however, on the CSa game it was 7% more than Fedora and 8%
more than Fedora on the CSi game. On the Quake3 game, Linux Ubuntu was 4% more than Fedora.

- Based on the graph presented in Figure 5-3-29, it can be said that OpenSUSE on Router 2 using IPv6 yielded 9% less values in terms of CPU utilisations on the games CSa than Linux Ubuntu and 10% less on the CSi game and also, 3% less on the Quake3 game.

- Based on the graph presented in Figure 5-3-29, it can concluded that Microsoft Windows 7 utilised the CPU resources the most on the Quake3 game and Linux Ubuntu on the games CSa and CSi when employing Jumbo frames with IPv6.

Figure 5-3-30 below presents the games CPU utilisations results from experiments conducted on various operating systems implemented on a network employing Jumbo frames with IPv4 on Router 2.

![Figure 5-3-30: Games CPU utilisations on Router2 on Jumbo frames IPv4](image)

This section presents the CPU utilisations results of all the games involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-30 above also presents the games CPU utilisations results from experiments conducted on various operating systems implemented on a network employing IPv4 on Router 2.

- Based on the graph presented in Figure 5-3-30, it can be said that Microsoft Windows Server 2008 on Router2 using IPv4 required 1% more of the CPU time on the game CSa than on the game CSi and on the Quake3 game it required less than 1% more than the CSi game.

- Based on the graph presented in Figure 5-3-30, it can be said that Microsoft Windows Server 2003 on Router 2 using IPv4 yielded the lowest values in terms of CPU utilisations on the games CSa and CSi while on the game Quake3 it required 3% more of the CPU time.
Based on the graph presented in Figure 5-3-30, it can be said that Microsoft Windows 7 on Router 2 using IPv4 yielded the highest values in terms of CPU utilisations on the games CSa while on the game CSi it required 18% less of the CPU time and 17% less on the game Quake3.

Based on the graph presented in Figure 5-3-30, it can be said that Linux Fedora on Router 2 using IPv4 required 13% less of the CPU time than Microsoft Windows 7 on the games CSa however Fedora required 2% more than it required for the game CSi and only 1% more than it required for the game Quake3.

Based on the graph presented in Figure 5-3-30, it can be said that Linux Ubuntu yielded the second highest on all of the games on Router 2 using IPv4 however, it was 2% less of the CPU time on the games CSa than Microsoft Windows 7 which was the highest but 3% more than Ubuntu required for the game CSi and 7% more than it required for the game Quake3.

Based on the graph presented in Figure 5-3-30, it can be said that OpenSUSE on Router 2 using IPv4 yielded similar values for the games CSa and CSi which was 1% more than it required for the game Quake3 in terms of CPU utilisations.

Based on the graph presented in Figure 5-3-30, it can concluded that Microsoft Windows 7 utilised the CPU resources the most on the CSa game and Linux Ubuntu was the second highest which was 2% less than Microsoft Windows 7 with regards to CPU utilisations however, on the games CSa and CSi when employing Jumbo frames with IPv4, Linux Ubuntu yielded the highest values in terms of CPU utilisations.

Section 5-3-20 next presents the comparison of Jumbo frames games IPv6 and IPv4 CPU utilisations results from the experiments conducted on various operating systems

5.3.20 Compare Games CPU Usage Results on IPv6 & IPv4 Jumbo Frames on Router 2

This section presents in bar graphs a comparison of the amount of CPU utilisations on Router 2 as required by the games involved in this study on IPv6 and IPv4 as presented on Figure 5-3-29 and Figure 5-3-30 for all of the six operating systems employing Jumbo frames.
The following list compares Jumbo frames IPv6 and IPv4 CPU utilisations on games from the experiments conducted on all of the operating systems as presented on Figure 5-3-31 above.

- Based on the graph presented in Figure 5-3-31, it can be said that, IPv4 on Microsoft Windows 7 on the CSa game was the highest in terms of CPU utilisations which was 3% more than Linux Ubuntu IPv4 which was the second highest. Linux Ubuntu IPv4 was 4% higher than Ubuntu IPv6 in terms of CPU utilisations while on Linux Fedora IPv6 was the second highest which was 7% less than Ubuntu IPv6. OpenSUSE IPv6 and IPv4 yielded similar values in terms of CPU utilisations however it was 6% less than Ubuntu IPv6.

- Based on the graph presented in Figure 5-3-31, it can be said that, on the CSI game IPv6 and IPv4 on Microsoft Windows Server 2008 and 2003 yielded similar values with regards to CPU utilisations which was the lowest when compared to the rest of the operating systems involved in this study.

- Based on the graph presented in Figure 5-3-31, it can be said that, Linux Ubuntu both IPv6 and IPv4 produced the highest values in terms of CPU utilisations however, IPv4 was slightly higher by 0.5% than IPv6 in which it was 10% higher than OpenSUSE where IPv6 and IPv4 required similar amount of the CPU time.

- Based on the graph presented in Figure 5-3-31, it can be said that, OpenSUSE and Linux Fedora IPv4 produced similar values in terms of CPU utilisations in which it was 2% lower than Fedora IPv6.

- Based on the graph presented in Figure 5-3-31, it can be said that, IPv6 on Microsoft Windows 7 4% higher than IPv4 however, it was less than 2% lower than IPv6 on Fedora and 9% less than Ubuntu IPv6.

- Based on the graph presented in Figure 5-3-31, it can be said that, on the Quake3 game IPv6 and IPv4 on Microsoft Windows Server 2008 and 2003 yielded similar throughput results however, IPv4 on Microsoft Windows Server 2003 was 3% higher.

### Figure 5-3-31: Compare Jumbo frames IPv6 & IPv4 CPU utilisation on games on Router2

<table>
<thead>
<tr>
<th>Name of Games</th>
<th>CSa</th>
<th>CSI</th>
<th>Quake3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server 2008 IPv6</td>
<td><img src="image1" alt="Graph" /></td>
<td><img src="image2" alt="Graph" /></td>
<td><img src="image3" alt="Graph" /></td>
</tr>
<tr>
<td>Server 2008 IPv4</td>
<td><img src="image4" alt="Graph" /></td>
<td><img src="image5" alt="Graph" /></td>
<td><img src="image6" alt="Graph" /></td>
</tr>
<tr>
<td>Windows 7 IPv6</td>
<td><img src="image7" alt="Graph" /></td>
<td><img src="image8" alt="Graph" /></td>
<td><img src="image9" alt="Graph" /></td>
</tr>
<tr>
<td>Windows 7 IPv4</td>
<td><img src="image10" alt="Graph" /></td>
<td><img src="image11" alt="Graph" /></td>
<td><img src="image12" alt="Graph" /></td>
</tr>
<tr>
<td>Server 2003 IPv6</td>
<td><img src="image13" alt="Graph" /></td>
<td><img src="image14" alt="Graph" /></td>
<td><img src="image15" alt="Graph" /></td>
</tr>
<tr>
<td>Server 2003 IPv4</td>
<td><img src="image16" alt="Graph" /></td>
<td><img src="image17" alt="Graph" /></td>
<td><img src="image18" alt="Graph" /></td>
</tr>
<tr>
<td>Fedora IPv6</td>
<td><img src="image19" alt="Graph" /></td>
<td><img src="image20" alt="Graph" /></td>
<td><img src="image21" alt="Graph" /></td>
</tr>
<tr>
<td>Fedora IPv4</td>
<td><img src="image22" alt="Graph" /></td>
<td><img src="image23" alt="Graph" /></td>
<td><img src="image24" alt="Graph" /></td>
</tr>
<tr>
<td>Ubuntu IPv6</td>
<td><img src="image25" alt="Graph" /></td>
<td><img src="image26" alt="Graph" /></td>
<td><img src="image27" alt="Graph" /></td>
</tr>
<tr>
<td>Ubuntu IPv4</td>
<td><img src="image28" alt="Graph" /></td>
<td><img src="image29" alt="Graph" /></td>
<td><img src="image30" alt="Graph" /></td>
</tr>
<tr>
<td>OpenSUSE IPv6</td>
<td><img src="image31" alt="Graph" /></td>
<td><img src="image32" alt="Graph" /></td>
<td><img src="image33" alt="Graph" /></td>
</tr>
<tr>
<td>OpenSUSE IPv4</td>
<td><img src="image34" alt="Graph" /></td>
<td><img src="image35" alt="Graph" /></td>
<td><img src="image36" alt="Graph" /></td>
</tr>
</tbody>
</table>
Based on the graph presented in Figure 5-3-31, it can be said that, IPv6 on Microsoft Windows 7 was the highest out of all the operating systems in which it was 25% higher than Microsoft Windows 7 IPv4.

Based on the graph presented in Figure 5-3-31, it can be said that, IPv4 on Linux Fedora and Ubuntu both required more of the CPU time than IPv6. IPv4 on Fedora was 1% more than IPv6 and IPv4 on Ubuntu was 3% more than IPv6 Ubuntu which made IPv4 Ubuntu the second highest for the game Quake3 however, it was 14% less than IPv6 on Microsoft Windows 7.

Based on the graph presented in Figure 5-3-31, it can be said that, IPv6 on OpenSUSE yielded similar CPU utilisations results to that of Ubuntu IPv4 however, that was 2% more than the results of IPv4 on OpenSUSE.

Based on the graph presented in Figure 5-3-31, it can be concluded that, IPv6 and IPv4 on Microsoft Windows Server 2008 and 2003 on the games CSa and CSI yielded similar values with regards to CPU utilisations apart from IPv4 on the Quake3 game where Microsoft Windows Server 2003 yielded 3% more than IPv6.

Based on the graph presented in Figure 5-3-31, it can be concluded that, IPv6 on Microsoft Windows 7 required more of the CPU time than IPv4 on the games CSi and Quake3 while IPv4 on the CSa game yielded more values than IPv6.

Based on the graph presented in Figure 5-3-31, it can be concluded that, IPv6 on Linux Fedora yielded more than IPv4 on the games CSa and CSI while IPv4 yielded more than IPv5 on the Quake3 game however for Fedora, IPv6 was the highest in terms of CPU utilisations.

Based on the graph presented in Figure 5-3-31, it can be concluded that, IPv6 and IPv4 OpenSUSE yielded similar values on the games CSa and CSI while on the game Quake3 IPv6 yielded 2% more than IPv4 in terms of CPU utilisations.

Based on the graph presented in Figure 5-3-31, it can be concluded that, on all of the games for Linux Ubuntu IPv4 utilised the CPU resources than IPv6 moreover, Ubuntu was the second highest on the games CSa and Quake3 while on the CSI game Ubuntu was the highest in fact IPv4.

Section 5-3-21 next presents the games packets dropped results on Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.

**5.3.21 Games Packets Dropped Results for IPv6 & IPv4 Jumbo Frames**

This section presents the packets dropped results for all the games involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-32 below presents the games packets dropped results from experiments conducted on various operating systems implemented on a network employing IPv6.
Based on the graph presented in Figure 5-3-32, it can be said that, there were no packets dropped for all of the operating systems on the games CSa and CSi.

Based on the graph presented in Figure 5-3-32, it can be said that on the game Quake3 Microsoft Windows Server 2008 on with IPv6 on Jumbo frames yielded the lowest values in terms of packets dropped while Microsoft Windows Server 2003 was the highest which 3% more than Microsoft Windows Server 2008.

Based on the graph presented in Figure 5-3-32, it can be said that, Linux Ubuntu was the second highest in terms of packets dropped which less than 1% than Microsoft Windows Server 2003 and only slightly higher than Microsoft Windows 7. Linux Fedora was 3% higher than Microsoft Windows Server 2008 and only slightly higher than OpenSUSE.

Based on the graph presented in Figure 5-3-32, it can be concluded that for all of the operating systems on IPv6, there was no packets lost for the games CSa and CSi while they all had packets lost on the Quake3 game.

Figure 5-3-33 below presents the games packets dropped results from experiments conducted on various operating systems implemented on a network employing Jumbo frames with IPv4.
Figure 5-3-33: Games packets dropped on Jumbo frames IPv4

- Based on the graph presented in Figure 5-3-33, it can be said that, there was no packets dropped for all of the operating systems on the games CSa and CSi.
- Based on the graph presented in Figure 5-3-33, it can be said that on the game Quake3 Microsoft Windows Server 2008 on with IPv4 on Jumbo frames yielded the highest values in terms of packets dropped while Linux Ubuntu was the lowest which 2% lower than Microsoft Windows Server 2008.
- Based on the graph presented in Figure 5-3-33, it can be said that, Microsoft Windows 7 was the second highest in terms of packets dropped which 1% less than Microsoft Windows Server 2008. Linux Fedora and Microsoft Windows Server 2003 yielded similar values which 1% lower than Microsoft Windows 7 and only slightly higher than OpenSUSE.
- Based on the graph presented in Figure 5-3-33, it can be concluded that for all of the operating systems on IPv4, there was no packets lost for the games CSa and CSi while they all had packets lost on the Quake3 game.

Section 5-3-22 next presents the comparison of Jumbo frames games IPv6 and IPv4 packets dropped results from the experiments conducted on various operating systems

**5.3.22 Compare Games Packets Dropped Results on IPv6 & IPv4 Jumbo Frames**

This section presents in bar graphs a comparison of the amount of lost packets on all of the games involved in this study on IPv6 and IPv4 as presented on Figure 5-3-32 and Figure 5-3-33 for all of the six operating systems employed Jumbo frames.
The following list compares Jumbo frames IPv6 and IPv4 packets dropped on games from the experiments conducted on all of the operating systems as presented on Figure 5-3-34 above.

- Based on the graph presented in Figure 5-3-34, it can be said that, all of the operating systems either IPv6 or IPv4 on Jumbo frames lost any packets on games CSa and CSi.
- Based on the graph presented in Figure 5-3-34, it can be said that, IPv4 on Microsoft Windows Server 2008 dropped more packets than any of the other operating systems involved in this study.
- Based on the graph presented in Figure 5-3-34, it can be said that, IPv4 on Microsoft Windows 7 dropped 1% more packets than IPv6 while IPv6 on Microsoft Windows Server 2003 dropped very similar amount of packets to that of IPv4 however it was 1% less than IPv4 Microsoft Windows 7.
- Based on the graph presented in Figure 5-3-34, it can be said that, IPv4 on Linux Fedora dropped 1% more packets than IPv6 while Ubuntu IPv6 dropped 0.5% more than IPv4 however, OpenSUSE IPv4 lost 1% more packets than IPv6.
- Based on the graph presented in Figure 5-3-34, it can be concluded that, IPv4 on four operating systems namely Microsoft Windows Server 2008, Microsoft Windows 7, Linux Fedora and OpenSUSE lost more packets than IPv6 while IPv6 on Microsoft Windows Server 2003 and Linux Ubuntu dropped more packets than IPv4.

Section 5-3-22 next presents the Voice over IP (VoIP) throughput results on Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.
5.3.23 VoIP Throughput Results for IPv6 & IPv4 Jumbo Frames

This section presents the throughput results for all Voice over IP CODECs involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-35 below presents the VoIP throughput results from experiments conducted on various operating systems implemented on a network employing IPv6.

- Based on the graph presented in Figure 5-3-35, it can be said that, for the G.711.1 CODEC Microsoft Windows Server 2003 yielded the highest throughput values however, it was only slightly higher than Linux Ubuntu.
- Based on the graph presented in Figure 5-3-35, it can be said that, for the G.711.1 CODEC Microsoft Windows Server 2008 yielded similar values with regards to throughput to that of Linux Fedora however, they were only 0.17% less than Microsoft Windows Server 2003.
- Based on the graph presented in Figure 5-3-35, it can be said that, for the G.711.1 CODEC Microsoft Windows 7 was 0.32% lower than Microsoft Windows Server 2003 while OpenSUSE was the lowest by 2% lower than Microsoft Windows Server 2003.
- Based on the graph presented in Figure 5-3-35, it can be said that, for the G.711.2 CODEC Microsoft Windows Server 2008 and Microsoft Windows 7 produced similar values in terms of throughput to that in G.711.1 CODEC.
- Based on the graph presented in Figure 5-3-35, it can be said that, for the G.711.2 CODEC OpenSUSE produced the highest values of throughput which was 2% more than what it produced for the G.711.1 CODEC.
- Based on the graph presented in Figure 5-3-35, it can be said that, for the G.711.2 CODEC Microsoft Windows Server 2003, Linux Fedora and Ubuntu produced less throughput values than what they produced in G.711.1 CODEC however, the difference was less than 1%.
Based on the graph presented in Figure 5-3-35, it can be said that, for the G.723.1 CODEC all of the operating systems yielded similar values in terms of throughput however, it was 1.9% lower than the throughput values of OpenSUSE in the G.711.2 CODEC.

Based on the graph presented in Figure 5-3-35, it can be said that, for the G.729.2 CODEC all of the operating systems yielded similar values in terms of throughput however, it was slightly higher when compared to their throughput values in the G.723.1 CODEC but 1.7% lower than the throughput values of OpenSUSE in the G.711.2 CODEC.

Based on the graph presented in Figure 5-3-35, it can be said that, for the G.729.3 CODEC all of the operating systems yielded similar values in terms of throughput however, it was slightly lower when compared to their throughput values in the G.729.2 CODEC but 1.8% lower than the throughput values of OpenSUSE in the G.711.2 CODEC.

Based on the graph presented in Figure 5-3-35, it can be concluded that, for a network utilising the VoIP technology while employing Jumbo frames with IPv6, all of the operating systems yielded more throughput values when using the G.711.1 and G.711.2 CODECs with OpenSUSE being the highest in the G.711.2 CODEC while the rest of the VoIP CODECs engaged in this study yielded up to 2% lower throughput.

Figure 5-3-36 below presents the VoIP throughput results from experiments conducted on various operating systems implemented on a network employing Jumbo frames with IPv4.

![VoIP Throughput on Jumbo Frames IPv4](image)

Based on the graph presented in Figure 5-3-36, it can be said that, for the G.711.1 CODEC Microsoft Windows 7 yielded the highest throughput values however, it was only slightly higher than Microsoft Windows Server 2008 and Server 2003.
Based on the graph presented in Figure 5-3-36, it can be said that, for the G.711.1 CODEC Microsoft Windows Server 2008 yielded similar values with regards to throughput to that of Microsoft Windows Server 2003 which was the second highest.

Based on the graph presented in Figure 5-3-36, it can be said that, for the G.711.1 CODEC Linux Fedora, Ubuntu and OpenSUSE yielded similar values in terms of throughput.

Based on the graph presented in Figure 5-3-36, it can be said that, for the G.711.2 CODEC Microsoft Windows 7, Linux Fedora and OpenSUSE produced similar values in terms of throughput.

Based on the graph presented in Figure 5-3-36, it can be said that, for the G.711.2 CODEC Microsoft Windows Server 2008, Server 2003 and Linux Ubuntu produced similar values of throughput however it was slightly lower when compared to that of Microsoft Windows 7, Linux Fedora and OpenSUSE.

Based on the graph presented in Figure 5-3-36, it can be said that, for the G.723.1 CODEC all of the operating systems yielded similar values in terms of throughput however, it was 1.8% lower than the throughput values of Microsoft Windows 7 in the G.711.1 CODEC.

Based on the graph presented in Figure 5-3-36, it can be said that, for the G.729.2 CODEC all of the operating systems yielded similar values in terms of throughput however, it was slightly higher when compared to their throughput values in the G.723.1 CODEC but 1.7% lower than the throughput values of Microsoft Windows 7 in the G.711.1 CODEC.

Based on the graph presented in Figure 5-3-36, it can be said that, for the G.729.3 CODEC all of the operating systems yielded similar values in terms of throughput however, it was slightly lower when compared to their throughput values in the G.729.2 CODEC but 1.8% lower than the throughput values of Microsoft Windows 7 in the G.711.1 CODEC.

Based on the graph presented in Figure 5-3-36, it can be concluded that, for a network utilising the VoIP technology while employing Jumbo frames with IPv4, all of the operating systems yielded more throughput values when using the G.711.1 and G.711.2 CODECs with Microsoft Windows 7 being the highest in the G.711.1 CODEC while the rest of the VoIP CODECs engaged in this study yielded up to 2% lower throughput.

Section 5-3-24 next presents the comparison of Jumbo frames VoIP throughput on IPv6 and IPv4 results from the experiments conducted on various operating systems.
5.3.24 Compare VoIP Throughput Results on IPv6 & IPv4 Jumbo Frames

This section presents in bar graphs a comparison of the VoIP throughput results on all of the VoIP CODECs involved in this study on IPv6 and IPv4 as presented on Figure 5-3-35 and Figure 5-3-36 for all of the six operating systems employed Jumbo frames.

![Compare Jumbo frames VoIP Codecs on IPv6 & IPv4 on Various OS](image)

The following list compares Jumbo frames IPv6 and IPv4 Voice over IP throughput results from the experiments conducted on all of the operating systems as presented on Figure 5-3-37 above.

- Based on the graph presented in Figure 5-3-37, it can be said that, for the G.711.1 CODEC Microsoft Windows Server 2003 IPv6 yielded the highest throughput values while Microsoft Windows 7 IPv4 and Linux Ubuntu IPv6 yielded similar values and they were the second highest which slightly less than Microsoft Windows Server 2003 IPv6.

- Based on the graph presented in Figure 5-3-37, it can be said that, for the G.711.1 CODEC OpenSUSE IPv6 yielded the lowest throughput values which 1.6% lower than OpenSUSE IPv4.

- Based on the graph presented in Figure 5-3-37, it can be said that, for the G.711.1 CODEC Microsoft Windows Server 2008 and Microsoft Windows 7 IPv4 yielded higher throughput values when compared to IPv6 throughput values with Microsoft Windows 7 IPv4 values slightly higher than Microsoft Windows Server 2008 IPv4 throughput values and Microsoft Windows Server 2008 IPv6 yielded 0.15% higher throughput values than Microsoft Windows 7 IPv6.

- Based on the graph presented in Figure 5-3-37, it can be said that, for the G.711.1 CODEC Linux Fedora IPv4, Ubuntu IPv4 and OpenSUSE IPv4 produced similar values with regards to throughput however, it was slightly lower than IPv6 throughput values yielded by Linux Fedora and Ubuntu with Ubuntu IPv6 being the highest among them.
Based on the graph presented in Figure 5-3-37, it can be concluded that, for the G.711.1 CODEC among the six operating systems involved in this study, three operating systems namely Microsoft Windows Server 2008, Microsoft Windows 7 and OpenSUSE IPv4 produced more throughput than IPv6 on the other hand, IPv6 on Microsoft Windows Server 2003, Linux Fedora and Ubuntu produced more throughput values than IPv4 however, IPv6 on Microsoft Windows Server 2003 and Linux Ubuntu yielded higher throughput values than the highest from IPv4 which was Microsoft Windows 7.

Based on the graph presented in Figure 5-3-37, it can be said that, for the G.711.2 CODEC OpenSUSE IPv6 yielded the highest throughput values while Microsoft Windows 7 IPv4 and Linux Fedora IPv4 yielded similar values and they were the second highest which slightly less than OpenSUSE IPv6.

Based on the graph presented in Figure 5-3-37, it can be said that, for the G.711.2 CODEC Linux Fedora IPv6 yielded the lowest throughput values which 4% lower than OpenSUSE IPv6.

Based on the graph presented in Figure 5-3-37, it can be said that, for the G.711.2 CODEC IPv6 on Microsoft Windows 7 yielded similar throughput values when compared to IPv4 throughput values on Microsoft Windows Server 2008, Server 2003 and IPv4 values on OpenSUSE.

Based on the graph presented in Figure 5-3-37, it can be concluded that, for the G.729.2 CODEC, between the six operating systems engaged in this study, three operating systems namely Microsoft Windows Server 2008, Linux Ubuntu and OpenSUSE IPv6 produced more throughput than IPv4 on the other hand, IPv4 on Microsoft Windows 7, Microsoft Windows Server 2003 and Linux Fedora produced more throughput values than IPv6 however, IPv6 on OpenSUSE yielded highest throughput values which 0.2% higher than the highest of IPv4 produced by Microsoft Windows 7.

Based on the graph presented in Figure 5-3-37, it can be concluded that, for the CODECs G.723.1 and G.729.3, all of the operating systems involved in this study on both IPv6 and IPv4 produced similar values of throughput however, it was significantly lower than the throughput values yielded by the operating systems on CODECs G.711.1 and G.711.2.

Based on the graph presented in Figure 5-3-37, it can be concluded that, for the G.729.2 CODEC, all of the operating systems involved in this study on both IPv6 and IPv4 produced similar values of throughput however, it was only slightly higher than the
throughput values yielded by the operating systems on CODECs G.723.1 while it was significantly lower than throughput values on CODECs G.711.1 and G.711.2.

Section 5-3-25 next presents the VoIP delay results on Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.

5.3.25 VoIP Delay Results for IPv6 & IPv4 Jumbo Frames

This section presents the delay results for all Voice over IP CODECs involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-38 below presents the VoIP throughput results from experiments conducted on various operating systems implemented on a network employing IPv6.

![VoIP Delay Results on Jumbo Frames IPv6](image)

- Based on the graph presented in Figure 5-3-38, it can be said that, Microsoft Windows Server 2008 on all of the VoIP CODECs created more delay values than any of the operating systems involved in this study however, for the G.711.1, G.711.2 and G.723.1 CODECs, Microsoft Windows Server 2008 generated similar delay values and similar values for the G.729.2 and G.729.3 CODECs and also slightly higher.
- Based on the graph presented in Figure 5-3-38, it can be said that, Microsoft Windows Server 2003 on all of the VoIP CODECs created similar delay values which was the lowest than any of the operating systems involved in this study.
- Based on the graph presented in Figure 5-3-38, it can be said that, Linux Ubuntu on all of the VoIP CODECs created similar delay values which was 51% less than Microsoft Windows Server 2008 and the second the lowest.
- Based on the graph presented in Figure 5-3-38, it can be said that, Linux Fedora on all of the VoIP CODECs created similar delay values which was 43% less than Microsoft Windows Server 2008 and the second the highest.
- Based on the graph presented in Figure 5-3-38, it can be said that, OpenSUSE on all of the VoIP CODECs created similar delay values which was 44% less than Microsoft Windows Server 2008.
Based on the graph presented in Figure 5-3-38, it can be said that, Microsoft Windows 7 on all of the VoIP CODECs created similar delay values which was 46% less than Microsoft Windows Server 2008.

Based on the graph presented in Figure 5-3-38, it can be concluded that, on a network that utilised the Jumbo frames together with the Voice over IP technology on IPv6, for all of the operating systems engaged in this study, Microsoft Windows Server 2008 created the highest delay while Microsoft Windows Server 2003 generated the lowest delay of 85% lower than Microsoft Windows Server 2008.

Figure 5-3-39 below presents the VoIP delay results from experiments conducted on various operating systems implemented on a network employing Jumbo frames with IPv4.

- Based on the graph presented in Figure 5-3-39, it can be said that, Microsoft Windows Server 2008 on all of the VoIP CODECs created more delay values than any of the operating systems involved in this study however, for the G.711.1, G.711.2 and G.723.1 CODECs, Microsoft Windows Server 2008 generated similar delay values and similar values for the G.729.2 and G.729.3 CODECs and also slightly higher.

- Based on the graph presented in Figure 5-3-39, it can be said that, Linux Ubuntu on all of the VoIP CODECs created similar delay values which was 53% lower than Microsoft Windows Server 2008 and also the lowest than any of the operating systems involved in this study.

- Based on the graph presented in Figure 5-3-39, it can be said that, OpenSUSE on all of the VoIP CODECs created similar delay values which was 41% less than Microsoft Windows Server 2008 and the second the lowest.
Based on the graph presented in Figure 5-3-39, it can be said that, Microsoft Windows 7 on all of the VoIP CODECs created similar delay values which was 36% less than Microsoft Windows Server 2008.

Based on the graph presented in Figure 5-3-39, it can be said that, Microsoft Windows Server 2003 on all of the VoIP CODECs created similar delay values which was 34% less than Microsoft Windows Server 2008.

Based on the graph presented in Figure 5-3-39, it can be said that, Linux Fedora on all of the VoIP CODECs created similar delay values which was 27% less than Microsoft Windows Server 2008.

Based on the graph presented in Figure 5-3-39, it can be concluded that, on a network that utilised the Jumbo frames together with the Voice over IP technology on IPv4, for all of the operating systems engaged in this study, Microsoft Windows Server 2008 created the highest delay while Linux Ubuntu generated the lowest delay of 53% lower than Microsoft Windows Server 2008.

Section 5-3-26 next presents the comparison of Jumbo frames VoIP delay on IPv6 and IPv4 results from the experiments conducted on various operating systems.

5.3.26 Compare VoIP Delay Results on IPv6 & IPv4 Jumbo Frames

This section presents in bar graphs a comparison of the VoIP delay results on all of the VoIP CODECs involved in this study on IPv6 and IPv4 as presented on Figure 5-3-38 and Figure 5-3-39 for all of the six operating systems employed Jumbo frames.

Figure 5-3-40: Compare VoIP delay results on Jumbo frames IPv6 & IPv4

The following list compares Jumbo frames IPv6 and IPv4 Voice over IP delay results from the experiments conducted on all of the operating systems as presented on Figure 5-3-40 above.
Based on the graph presented in Figure 5-3-40, it can be said that, Microsoft Windows Server 2008 on all of the VoIP CODECs created more delay values on both IPv6 and IPv4 than any of the operating systems involved in this study however, for all of the VoIP CODECs Microsoft Windows Server 2008 generated similar delay values for IPv6 and also similar values for IPv4 but IPv6 generated 5% more delay than IPv4.

Based on the graph presented in Figure 5-3-40, it can be said that, Microsoft Windows 7 IPv4 generated 5% more delay than IPv6 however Microsoft Windows 7 IPv6 generated 46% less delay than Microsoft Windows Server 2008 IPv6.

Based on the graph presented in Figure 5-3-40, it can be said that, Microsoft Windows 7 IPv4 generated 5% more delay than IPv6 however Microsoft Windows 7 IPv6 generated 46% less delay than Microsoft Windows Server 2008 IPv6.

Based on the graph presented in Figure 5-3-40, it can be said that, Microsoft Windows Server 2003 on all of the VoIP CODECs created less delay values on IPv6 than any of the operating systems involved in this study however, for all of the VoIP CODECs Microsoft Windows Server 2008 generated similar delay values for IPv6 and also for IPv4 but IPv4 on Microsoft Windows Server 2003 created 46% more delay values the IPv6 and 85% less than IPv6 Microsoft Windows Server 2008.

Based on the graph presented in Figure 5-3-40, it can be said that, IPv6 on Linux Ubuntu on all of the VoIP CODECs created similar delay values for all of the VoIP CODECs while IPv4 on Ubuntu generated similar values on all of the CODECs which was 6% lower but 51% lower than IPv6 on Microsoft Windows Server 2008.

Based on the graph presented in Figure 5-3-40, it can be said that, IPv6 on OpenSUSE on all of the VoIP CODECs created similar delay values for all of the VoIP CODECs while IPv4 on OpenSUSE generated similar values on all of the CODECs which was 2% lower but 44% lower than IPv6 on Microsoft Windows Server 2008.

Based on the graph presented in Figure 5-3-40, it can be concluded that IPv4 on three of the operating systems involved in this study namely Microsoft Windows 7, Microsoft Windows Server 2003 and Linux Fedora generated more delay values than IPv6 while IPv6 on Microsoft Windows Server 2008, Linux Ubuntu and OpenSUSE created more delay than IPv4.

Based on the graph presented in Figure 5-3-40, it can be concluded that Jumbo frames on Microsoft Windows Server 2008 on both IPv6 and IPv4 generated more delays on all of the VoIP CODECs than any of the operating systems involved in this study.

Based on the graph presented in Figure 5-3-40, it can be concluded that Jumbo frames on IPv6 on Microsoft Windows Server 2003 generated the lowest delays values on all of the VoIP CODECs than any of the operating systems involved in this study which was 85% lower than IPv6 on Microsoft Windows Server 2008 which was the highest in terms of delay values.
Section 5-3-27 next presents the VoIP jitter results on Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.

### 5.3.27 VoIP Jitter Results for IPv6 & IPv4 Jumbo Frames

This section presents the jitter results for all Voice over IP CODECs involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-41 below presents the VoIP jitter results from experiments conducted on various operating systems implemented on a network employing IPv6.

Figure 5-3-41: VoIP jitter results on Jumbo frames IPv6

- Based on the graph presented in Figure 5-3-41, it can be said that, on the VoIP CODEC G.711.1, operating systems Microsoft Windows Server 2008, Server 2003, Linux Fedora, and Ubuntu generated similar amounts of jitter values while Microsoft Windows 7 and OpenSUSE generated 1% more jitter.

- Based on the graph presented in Figure 5-3-41, it can be said that, on the VoIP CODEC G.711.2, operating systems Microsoft Windows 7, Server 2003, Linux Fedora, Ubuntu and OpenSUSE generated similar jitter values while Microsoft Windows Server 2008 generated 2% more jitter.

- Based on the graph presented in Figure 5-3-41, it can be said that, on the VoIP CODEC G.723.1, operating systems Microsoft Windows 7, Server 2003 required similar amounts of jitter which was 1% more than Microsoft Windows Server 2008 which generated the lowest jitter however, Linux Fedora was the highest in terms of jitter values which was 3% more than Microsoft Windows 7 and Microsoft Windows Server 2003 and OpenSUSE generated 1% less jitter than Fedora and Ubuntu was 1% less than OpenSUSE.

- Based on the graph presented in Figure 5-3-41, it can be said that, on the VoIP CODEC G.729.2, operating systems Microsoft Windows 7 was the highest which was 1% more
than operating systems Microsoft Windows Server 2008, Server 2003, Linux Fedora, Ubuntu and OpenSUSE in which they all generated similar amounts of jitter.

- Based on the graph presented in Figure 5-3-41, it can be said that, on the VoIP CODEC G.729.3, all of the operating systems generated similar amounts of jitter.

- Based on the graph presented in Figure 5-3-41, it can be concluded that, Microsoft Windows Server 2003 on all of the VoIP CODECs included in this study generated similar amounts of jitter while the other operating systems vary depending on CODECs employed.

Figure 5-3-42 below presents the VoIP jitter results from experiments conducted on various operating systems implemented on a network employing Jumbo frames with IPv4.

- Based on the graph presented in Figure 5-3-42, it can be said that, on the VoIP CODEC G.711.1, operating systems Microsoft Windows Server 2008 and Linux Ubuntu generated similar amounts of jitter values while Microsoft Windows 7, Microsoft Windows Server 2003 and Linux Fedora generated similar amounts of jitter which was 1% less and OpenSUSE generated 2% less jitter than Microsoft Windows Server 2008 and Linux Ubuntu.

- Based on the graph presented in Figure 5-3-42, it can be said that, on the VoIP CODEC G.711.2, all of the operating systems involved in this study generated similar values in terms of jitter which was 3% less than the jitter values on Microsoft Windows Server 2008 and Linux Ubuntu on the G.711.1 CODEC.

- Based on the graph presented in Figure 5-3-42, it can be said that, on the VoIP CODEC G.723.1, operating systems Microsoft Windows Server 2008, Server 2003 and Linux Ubuntu generated similar values in terms of jitter while Microsoft Windows 7, Linux Fedora and OpenSUSE generated similar jitter values which was 1% higher than the others.
Based on the graph presented in Figure 5-3-42, it can be said that, on the VoIP CODEC G.729.2, all of the operating systems involved in this study generated similar values in terms of jitter which was similar values on the G.711.2 CODEC.

Based on the graph presented in Figure 5-3-42, it can be said that, on the VoIP CODEC G.729.3, operating systems Microsoft Windows Server 2008, Microsoft Windows 7 and Linux Ubuntu generated similar values in terms of jitter while Microsoft Windows Server 2003, Linux Ubuntu and OpenSUSE generated similar jitter values which was 1% lower than the others.

Based on the graph presented in Figure 5-3-42, it can be concluded that, all of the operating systems on CODECs G.711.2 and G.729.2 generated similar amounts of jitter and that was similar to the amounts generated by Microsoft Windows Server 2008, Server 2003 and Linux Ubuntu on CODEC G.723.1 and also similar to the amounts generated by operating systems Microsoft Windows Server 2003, Linux Ubuntu and OpenSUSE on CODEC G.729.3 while on CODEC G.711.1 the amounts of jitter generated by all of the operating systems were up to 3% more.

Section 5-3-27 next presents the comparison of Jumbo frames VoIP jitter on IPv6 and IPv4 results from the experiments conducted on various operating systems

### 5.3.27 Compare VoIP Jitter Results on IPv6 & IPv4 Jumbo Frames

This section presents in bar graphs a comparison of the VoIP jitter results on all of the VoIP CODECs involved in this study on IPv6 and IPv4 as presented on Figure 5-3-41 and Figure 5-3-42 for all of the six operating systems employed Jumbo frames.

![Compare VoIP Jitter Results on Jumbo Frames IPv6 & IPv4](figure.png)

The following list compares Jumbo frames IPv6 and IPv4 Voice over IP jitter results from the experiments conducted on all of the operating systems as presented on Figure 5-3-43 above
- Based on the graph presented in Figure 5-3-43, it can be said that, IPv4 on Microsoft Windows Server 2008 and IPv4 on Linux Ubuntu on CODEC G.711.1 produced similar values to that of IPv6 on Linux Fedora on G.723.1 CODEC which were the highest values in terms of jitter.

- Based on the graph presented in Figure 5-3-43, it can be said that, on CODEC G.711.1, IPv4 on operating systems Microsoft Windows 7, Server 2003 and Linux Fedora produced similar values in terms of jitter which was 1% lower when compared to the amount of jitter generated by IPv4 on Microsoft Windows Server 2008 and Linux Ubuntu however, it was 1% higher than IPv6 on Microsoft Windows 7 and IPv6 and IPv4 on OpenSUSE not only that but 2% higher than the amount of jitter generated by IPv6 on operating system Microsoft Windows Server 2008, Server 2003, Linux Fedora and Ubuntu.

- Based on the graph presented in Figure 5-3-43, it can be said that, on CODEC G.711.2, IPv6 on operating systems Microsoft Windows Server 2008 generated the highest amount of jitter which was 2% higher IPv4 on Microsoft Windows Server 2008 however, IPv4 on Microsoft Windows Server 2008 generated similar amount of jitter to that of IPv6 and IPv4 on operating systems Microsoft Windows Server 2003, Microsoft Windows 7, Linux Fedora, Ubuntu and OpenSUSE.

- Based on the graph presented in Figure 5-3-43, it can be said that, on CODEC G.723.1 IPv6 on Microsoft Windows Server 2008 generated the lowest values in terms of jitter which was 1% lower than IPv4. IPv4 on Microsoft Windows Server 2008 had similar jitter values with IPv6 on Microsoft Windows 7 and that was 1% lower than Microsoft Windows 7 IPv4 which had similar jitter values to that of IPv4 on Linux Fedora and OpenSUSE and IPv6 on Ubuntu in which they yielded 1% more jitter values than that generated by IPv4 on Microsoft Windows Server 2008, Server 2003 Linux Ubuntu and IPv6 on Microsoft Windows 7 and Microsoft Windows Server 2003.

- Based on the graph presented in Figure 5-3-43, it can be said that, on CODEC G.729.2 apart from IPv6 on Microsoft Windows 7 which created 1% more jitter IPv4, all of the rest of the operating systems involved in this study on both IPv6 and IPv4 generated similar amounts jitter to that of IPv4 on Microsoft Windows 7.

- Based on the graph presented in Figure 5-3-43, it can be said that, on CODEC G.729.3 IPv4 on Microsoft Windows Server 2008, Microsoft Windows 7 and Linux Fedora generated similar amounts of jitter which was 1% more when compared to IPv6 however, IPv6 on operating systems Microsoft Windows Server 2008, Microsoft Windows 7 and Linux Fedora generated similar amounts of jitter to that of IPv6 and IPv4 on operating systems Microsoft Windows Server 2003, Linux Ubuntu and OpenSUSE.
Section 5-3-29 next presents the VoIP CPU utilisations results on Router 1 on Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.

### 5.3.29 VoIP CPU Usage Results for IPv6 & IPv4 Jumbo Frames on Router 1

This section presents the CPU utilisations results on Router 1 for all Voice over IP CODECs involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-44 presents the VoIP CPU utilisations results on Router 1 from experiments conducted on various operating systems implemented on a network employing IPv6.

![VoIP CPU Usage on Router1 - Jumbo Frames IPv6](image)

- Based on the graph presented in Figure 5-3-44, it can be said that, Microsoft Windows Server 2008 required similar amounts of the CPU time on CODECs G.711.1, G.711.2 and G.723.1 while on CODEC G.729.2 required 8% more of the CPU time which was 4% more than what G.729.3 CODEC on IPv6 required.

- Based on the graph presented in Figure 5-3-44, it can be said that, Microsoft Windows 7 required similar amounts of the CPU time on CODECs G.711.2, G.723.1, G.729.2 and G.729.3 while on CODEC G.711.1 required 2% more of the CPU time.

- Based on the graph presented in Figure 5-3-44, it can be said that, Microsoft Windows Server 2003 required similar amounts of the CPU time on all of the VoIP CODECs involved in this study however it was 2% lower than Microsoft Windows 7 on the G.711.1 CODEC.

- Based on the graph presented in Figure 5-3-44, it can be said that, Linux Fedora required similar amounts of the CPU time on VoIP CODECs G.711.1, G.711.2 and G.729.3 however, on CODECs G.723.1 and G.729.2 Linux Fedora utilised slightly more of the CPU resources.

- Based on the graph presented in Figure 5-3-44, it can be said that, Linux Ubuntu required similar amounts of the CPU time on VoIP CODECs G.711.1, G.711.2 and
G.729.3 however, it was 2% less than the CPU utilisations of CODECs G.723.1 and 3% more than the amount of the CPU time required by G.729.3 CODEC.

Based on the graph presented in Figure 5-3-44, it can be said that, OpenSUSE required similar amounts of the CPU time on VoIP CODECs G.711.1, G.723.1 and G.729.2 however, it was 4% more than the CPU utilisations of CODECs G.711.2 and 4% more than the amount of the CPU time required by G.729.3 CODEC which was the highest in terms of CPU utilisations.

Figure 5-3-45 below presents the VoIP CPU usage on Router 1 results from experiments conducted on various operating systems implemented on a network employing Jumbo frames with IPv4

![VoIP CPU Usage on Router 1 - Jumbo Frames IPv4](image)

- Based on the graph presented in Figure 5-3-45, it can be said that, Microsoft Windows Server 2008 required similar amounts of the CPU time on CODECs G.711.1, G.711.2, G.723.1 and CODEC G.729.3 while CODEC G.729.2 required 5% more of the CPU time.
- Based on the graph presented in Figure 5-3-45, it can be said that, Microsoft Windows 7 required similar amounts of the CPU time on CODECs G.723.1, G.729.2 and G.729.3 while on CODECs G.711.1 and G.711.2 required 2% more of the CPU time.
- Based on the graph presented in Figure 5-3-45, it can be said that, Microsoft Windows Server 2003 required similar amounts of the CPU time on all of the VoIP CODECs involved in this study however it was 1% lower than Microsoft Windows 7 on the G.711.2 CODEC.
- Based on the graph presented in Figure 5-3-45, it can be said that, Linux Fedora required similar amounts of the CPU time on VoIP CODECs G.711.2 and G.723.1 however, on CODECs G.729.2 and G.729.3 Linux Fedora utilised similar amounts of CPU resources which 1% more and on CODEC G.711.1 Fedora required 2% more of
the CPU resources when compared to the CPU usage of CODECs G.711.2 and G.723.1.

- Based on the graph presented in Figure 5-3-45, it can be said that, Linux Ubuntu required similar amounts of the CPU time on VoIP CODECs G.711.1 and G.729.2 while CODECs G.711.2 and G.729.3 yielded similar values in terms of CPU utilisations however, it was 1% less than the CPU utilisations of CODECs G.711.1 and G.729.2 and 1% more than the amount of the CPU time required by G.723.1CODEC.

- Based on the graph presented in Figure 5-3-45, it can be said that, OpenSUSE required similar amounts of the CPU time on VoIP CODECs G.711.1, G.711.2, G.729.2 and G.729.3 however, it was 3% less than the CPU utilisations of CODECs G.723.1 which was the highest in terms of CPU utilisations on all of the CODECs and all of the operating systems.

Section 5-3-30 next presents the comparison of Jumbo frames VoIP CPU utilisations on IPv6 and IPv4 results from the experiments conducted on various operating systems on Router 1

5.3.30 Compare VoIP IPv6 & IPv4 Jumbo Frames CPU Usage Results on Router1

This section presents in bar graphs a comparison of the VoIP CPU utilisations results on Router 1 on all of the VoIP CODECs involved in this study on IPv6 and IPv4 as presented on Figure 5-3-44 and Figure 5-3-45 for all of the six operating systems employed Jumbo frames.
The following list compares Jumbo frames IPv6 and IPv4 Voice over IP CPU utilisations results on Router 1 from the experiments conducted on all of the operating systems as presented on Figure 5-3-46 above.

- Based on the graph presented in Figure 5-3-46, it can be said that, IPv6 and IPv4 on Microsoft Windows Server 2008 produced similar values in terms of CPU utilisations on Router 1 on VoIP CODECs G.711.1, G.711.2 and G.723.1 however, on CODECs G.729.2 and G.729.3 IPv6 entailed CPU resources more than IPv4 by 4%.

- Based on the graph presented in Figure 5-3-46, it can be said that, IPv6 on Microsoft Windows 7 required 1% more of the CPU time on Router 1 on VoIP CODECs G.711.1 than IPv4 however, on CODEC G.711.2 IPv4 utilised 1% more of the CPU time than IPv6 while on CODECs G.723.1, G.729.2 and G.729.3 IPv6 and IPv4 produced similar values in terms of CPU utilisations.

- Based on the graph presented in Figure 5-3-46, it can be said that, Microsoft Windows Server 2003 on Router 1 on VoIP CODECs G.711.1, G.711.2 and G.729.2 produced similar values in terms of CPU utilisations to that of IPv4 however, on CODECs G.723.1 and G.729.3 IPv4 entailed more of the CPU resources than IPv6.

- Based on the graph presented in Figure 5-3-46, it can be said that, Linux Fedora on Router 1 on VoIP CODECs G.711.2, G.723.1 and G.729.2 IPv4 entailed up to 2% more CPU resources than IPv6 however, on CODECs G.711.1 and G.729.3 IPv6 required up to 2% more of the CPU time than IPv4.

- Based on the graph presented in Figure 5-3-46, it can be said that, Linux Ubuntu on Router 1 on VoIP CODECs G.711.1, G.711.2, G.729.2 and G.729.3 IPv4 entailed up to 4% more CPU resources than IPv6 however, on CODEC G.723.1 IPv6 required 2% more of the CPU time than IPv4.

- Based on the graph presented in Figure 5-3-46, it can be said that, OpenSUSE on Router 1 on VoIP CODECs G.711.1 and G.729.2 IPv6 and IPv4 required similar amount of the CPU time while IPv4 on CODECs G.711.2 and G.723.1 entailed up to 4% of the CPU time than IPv6 however, on CODEC G.729.3 IPv6 entailed 4% more CPU resources than IPv4.

Section 5-3-31 next presents the VoIP CPU utilisations results on Router 2 on Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.

5.3.31 VoIP CPU Usage Results for IPv6 & IPv4 Jumbo Frames on Router 2

This section presents the CPU utilisations results on Router 2 for all Voice over IP CODECs involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-47 below presents the VoIP CPU utilisations results on Router 2 from
experiments conducted on various operating systems implemented on a network employing IPv6.

Figure 5-3-47: VoIP IPv6 Jumbo frames CPU usage on Router 2

- Based on the graph presented in Figure 5-3-47, it can be said that, Microsoft Windows Server 2008 on CODEC G.711.1 required 2% more of the CPU time than Microsoft Windows 7 and Server 2003 however, it was 5% less than the requirement of Linux Fedora.

- Based on the graph presented in Figure 5-3-47, it can be said that, Microsoft Windows Server 2003 was the lowest in terms of CPU utilisations on all of the VoIP CODECs which was 13% less than Linux Ubuntu which produced the highest values in terms of CPU utilisations.

- Based on the graph presented in Figure 5-3-47, it can be said that, Linux Fedora on CODECs G.711.1 and G.711.2 required similar amounts of the CPU time also on VoIP CODECs G.723.1 and G.729.3 Fedora utilised similar amounts of CPU resources which was 1% less however, Fedora on G.729.2 required 5% more of the CPU time than G.723.

- Based on the graph presented in Figure 5-3-47, it can be said that, Linux Ubuntu when compared to Fedora, it required 4% more of the CPU time on CODEC G.711.1 3% more on G.711.2 and 5% more on G.723.1, 1% more on G.729.2 and 6% more on G.729.3.

- Based on the graph presented in Figure 5-3-47, it can be said that, OpenSUSE required similar amounts of the CPU time on VoIP CODECs G.711.1, G.723.1, G.729.2 and G.729.3 however, it was 2% less on the G.711.2 CODEC.

Figure 5-3-48 below presents the VoIP CPU usage on Router 2 results from experiments conducted on various operating systems implemented on a network employing Jumbo frames with IPv4.
Based on the graph presented in Figure 5-3-48, it can be said that, Microsoft Windows Server 2008 on CODEC G.711.1 required 2% more of the CPU time than Microsoft Windows 7 and Server 2003 however, it was 4% less than the requirement of Linux Fedora.

Based on the graph presented in Figure 5-3-48, it can be said that, Microsoft Windows 7 on G.711.1 required 2% more of the CPU time however Microsoft Windows 7 required similar amount of the CPU time on G.711.2, G.723.1, G.729.2 and G.729.3.

Based on the graph presented in Figure 5-3-48, it can be said that, Microsoft Windows Server 2003 was the lowest in terms of CPU utilisations on all of the VoIP CODECs which was 15% less than Linux Ubuntu which produced the highest values in terms of CPU utilisations on CODEC G.723.1.

Based on the graph presented in Figure 5-3-48, it can be said that, Linux Fedora on CODECs G.711.1, G.711.2, G.723.1 and G.729.3 required similar amounts of the CPU time while on VoIP CODEC G.729.2 Fedora utilised 1% more of the CPU resources.

Based on the graph presented in Figure 5-3-48, it can be said that, Linux Ubuntu when compared to Fedora, it required 3% more of the CPU time on CODEC G.711.1, 6% more on G.711.2 and 8% more on G.723.1, 5% more on G.729.2 and 6% more on G.729.3.

Based on the graph presented in Figure 5-3-48, it can be said that, OpenSUSE required similar amounts of the CPU time on VoIP CODECs G.711.1, G.711.2, G.729.2 and G.729.3 however, it was 2% more on the G.723.1 CODEC.

Section 5-3-32 next presents the comparison of Jumbo frames VoIP CPU utilisations on IPv6 and IPv4 results from the experiments conducted on various operating systems on Router 2
5.3.32 Compare VoIP IPv6 & IPv4 Jumbo Frames CPU Usage Results on Router2

This section presents in bar graphs a comparison of the VoIP CPU utilisations results on Router 2 on all of the VoIP CODECs involved in this study on IPv6 and IPv4 as presented on Figure 5-3-47 and Figure 5-3-48 for all of the six operating systems employed Jumbo frames.

The following list compares Jumbo frames IPv6 and IPv4 Voice over IP CPU utilisations results on Router 2 from the experiments conducted on all of the operating systems as presented on Figure 5-3-49 above:

- Based on the graph presented in Figure 5-3-49, it can be said that, IPv4 on Microsoft Windows Server 2008 produced similar values in terms of CPU utilisations on Router 2 on CODECs G.711.2, G.723.1, G.729.2 and G.729.3 however, on CODECs G.729.2 and G.729.3 IPv6 entailed CPU resources slightly more than IPv4. On CODEC G.711.1 IPv6 required 2% more of the CPU time while IPv4 required 1% more than IPv6.

- Based on the graph presented in Figure 5-3-49, it can be said that, IPv6 on Microsoft Windows 7 CPU utilisations values on Router 2 were slightly more on VoIP CODECs G.711.2 and G.729.3 than IPv4 however, on CODEC G.711.1 IPv4 utilised 1% more of the CPU time than IPv6 while on CODECs G.723.1 and G.729.2 IPv6 and IPv4 produced similar values in terms of CPU utilisations.

- Based on the graph presented in Figure 5-3-49, it can be said that, Microsoft Windows Server 2003 on Router 2 IPv6 and IPv4 produced similar values in terms of CPU utilisations to on all of the VoIP CODECs engaged in this study.

- Based on the graph presented in Figure 5-3-49, it can be said that, Linux Fedora on Router 2 on VoIP CODECs G.711.1, G.711.2 and G.729.2 IPv6 entailed up to 4% more
CPU resources than IPv4 however, on CODECs G.723.1 and G.729.3 IPv4 CPU utilisations values were slightly higher than IPv6.

- Based on the graph presented in Figure 5-3-49, it can be said that, Linux Ubuntu on Router 2 on VoIP CODECs G.711.2, G.723.1 and G.729.3 IPv4 entailed up to 3% more CPU resources than IPv6 however, on CODEC G.711.1 IPv6 required 3% more of the CPU time than IPv4 while IPv6 and IPv4 required similar amount of the CPU time on G.729.2 CODEC.

- Based on the graph presented in Figure 5-3-49, it can be said that, OpenSUSE on Router 2 on VoIP CODECs G.711.1, G.729.2 and G.729.3 IPv6 and IPv4 required similar amount of the CPU time while IPv4 on CODECs G.711.2 and G.723.1 entailed up to 2% of the CPU time than IPv6.

Section 5-3-33 next presents the VoIP packets dropped results on Router 2 on Jumbo frames IPv6 and IPv4 from the experiments conducted on various operating systems.

**5.3.33 VoIP Packets Dropped Results for IPv6 & IPv4 Jumbo Frames on Router2**

This section presents the CPU packets dropped results on Router 2 for all Voice over IP CODECs involved in this study from the experiments conducted on a network employing Jumbo frames. Figure 5-3-50 below presents the VoIP packets dropped results on Router 2 from experiments conducted on various operating systems implemented on a network employing IPv6.

- Based on the graph presented in Figure 5-3-50, it can be said that, on IPv6 there was no packets dropped for all of the operating systems on VoIP CODECs G.723.1, G.729.2 and G.729.3.
Based on the graph presented in Figure 5-3-50, it can be said that on CODEC G.711.1 Microsoft Windows Server 2008 with IPv6 on Jumbo frames dropped 7% more packets than Microsoft Windows Server 2008 on G.711.2 CODEC.

Based on the graph presented in Figure 5-3-50, it can be said that on CODEC G.711.1 Microsoft Windows 7 with IPv6 on Jumbo frames dropped 9% more packets than Microsoft Windows 7 on G.711.2 CODEC.

Based on the graph presented in Figure 5-3-50, it can be said that on CODEC G.711.1 Microsoft Windows Server 2003 with IPv6 on Jumbo frames dropped 1% more packets than Microsoft Windows 7.

Based on the graph presented in Figure 5-3-50, it can be said that on CODEC G.711.1 Microsoft Windows Server 2003 with IPv6 on Jumbo frames dropped 1% more packets than Microsoft Windows 7.

Based on the graph presented in Figure 5-3-50, it can be said that on CODEC G.711.1 Linux Fedora with IPv6 on Jumbo frames dropped 9% less packets than Microsoft Windows Server 2003 which dropped the most packets on G.711.1 CODEC.

Based on the graph presented in Figure 5-3-50, it can be said that on CODEC G.711.1 Linux Ubuntu with IPv6 on Jumbo frames packets dropped values was slightly higher than Fedora but 8% less than Microsoft Windows Server 2003 which dropped the most packets on G.711.1 CODEC.

Based on the graph presented in Figure 5-3-50, it can be said that on CODEC G.711.1 OpenSUSE with IPv6 on Jumbo frames packets dropped values was slightly higher than Linux Ubuntu.

Based on the graph presented in Figure 5-3-50, it can be said that on CODEC G.711.2 operating systems Microsoft Windows Server 2003, Linux Fedora, Ubuntu and OpenSUSE dropped no packets on VoIP CODEC G.711.2.

Figure 5-3-51 below presents the VoIP packets dropped results from experiments conducted on various operating systems implemented on a network employing Jumbo frames with IPv4.

Figure 5-3-51: VoIP packets dropped on Jumbo frames IPv4
Based on the graph presented in Figure 5-3-51, it can be said that, on IPv4 there was no packets dropped for all of the operating systems on VoIP CODECs G.723.1, G.729.2 and G.729.3.

Based on the graph presented in Figure 5-3-51, it can be said that on CODEC G.711.1 Microsoft Windows Server 2008 with IPv6 on Jumbo frames dropped 8% more packets than Microsoft Windows Server 2008 on G.711.2 CODEC however, it was 5% less than Microsoft Windows 7 which dropped the most on G.711.1 CODEC.

Based on the graph presented in Figure 5-3-51, it can be said that on CODEC G.711.1 Microsoft Windows Server 2003 with IPv6 on Jumbo frames dropped 1% less packet than Microsoft Windows Server 2008 but 6% less than Microsoft Windows 7 and 7% more than it dropped on G.711.2 CODEC.

Based on the graph presented in Figure 5-3-51, it can be said that on CODEC G.711.1 Linux Fedora with IPv6 on Jumbo frames dropped 6% less packets than Microsoft Windows Server 2003 but 12% less than Microsoft Windows 7 which dropped the most packets on G.711.1 CODEC.

Based on the graph presented in Figure 5-3-51, it can be said that on CODEC G.711.1 Linux Ubuntu with IPv6 on Jumbo frames dropped 1% less packet than Linux but 13% less than Microsoft Windows 7 which dropped the most packets on G.711.1 CODEC.

Based on the graph presented in Figure 5-3-51, it can be said that on CODEC G.711.1 OpenSUSE with IPv6 on Jumbo frames dropped 3% more packets than Linux Ubuntu however it was 10% less than Microsoft Windows 7.

Based on the graph presented in Figure 5-3-51, it can be said that on CODEC G.711.2 operating systems Microsoft Windows 7, Linux Fedora, Ubuntu and OpenSUSE dropped no packets on VoIP CODEC G.711.2.

In section 5-3-34 the comparison of Jumbo frames VoIP packets dropped on IPv6 and IPv4 results from the experiments conducted on various operating systems is presented.

**5.3.34 Compare VoIP Packets Dropped Results on IPv6 & IPv4 Jumbo Frames**

In this section, a comparison of the VoIP packets dropped results on all of the VoIP CODECs involved in this study on IPv6 and IPv4 are presented in Figure 5-3-50 and Figure 5-3-51 for all of the six operating systems employed Jumbo frames.
The following list compares Jumbo frames IPv6 and IPv4 packets dropped on VoIP from the experiments conducted on all of the operating systems as presented on Figure 5-3-52.

- Based on the graph presented in Figure 5-3-52, it can be said that, IPv4 Microsoft Windows Server 2008 on CODEC G.711.1 dropped 3% more packets than IPv4 on Jumbo frames however, on G.711.2 IPv4 dropped 1% more packets than IPv6.

- Based on the graph presented in Figure 5-3-52, it can be said that, IPv4 Microsoft Windows 7 on CODEC G.711.1 dropped 6% more packets than IPv6 on Jumbo frames which was the highest however, on G.711.2 IPv6 dropped 1% more packets than IPv4.

- Based on the graph presented in Figure 5-3-52, it can be said that, IPv6 Microsoft Windows Server 2003 on CODEC G.711.1 dropped 1% more packets than IPv4 while IPv6 dropped no packets on G.711.2 when IPv4 dropped 3%.

- Based on the graph presented in Figure 5-3-52, it can be said that, IPv4 Linux Fedora on CODEC G.711.1 dropped 3% more packets than IPv6 on Jumbo frames while IPv6 on Linux Ubuntu packets dropped rate was slightly higher than IPv6 Fedora however, IPv4 on Ubuntu dropped 2% more than IPv6.

- Based on the graph presented in Figure 5-3-52, it can be said that, IPv4 on OpenSUSE on CODEC G.711.1 dropped 4% more packets than IPv6 however, it was 10% lower than IPv4 on Microsoft Windows 7 which was the highest.

- Based on the graph presented in Figure 5-3-52, it can be said that, IPv6 on Microsoft Windows Server 2003 and IPv6 and IPv4 on operating systems Linux Fedora, Ubuntu and OpenSUSE dropped no packets on the G.711.2 CODEC.
5.4 Summary

In this chapter, the analysis of the experimental results of this study has been presented in both line and column graphs. In this chapter there were three main sections that started with analysing the performance of Jumbo frames TCP and UDP throughput, delay jitter and the CPU utilisations on Router 1 and 2 and also the percentages of packets lost on both IPv6 and IPv4. This was followed by comparing TCP and UDP and also IPv6 and IPv4 on all of the operating systems involved.

This chapter also included the analysis of the performance of all the applications traffic that was engaged in this study which were, DNS, Games and VoIP. The analysis of these were based on Jumbo frames throughput, delay jitter and the CPU utilisations on Router 1 and 2 and also the percentages of packets lost on both IPv6 and IPv4. This was followed by comparing IPv6 and IPv4 on all of the operating systems involved. The discussions of this chapter will be covered in Chapter six next.
Chapter 6: Discussion

In this chapter is the findings established as the result of the experiments conducted in this study are discussed. Chapter 3 delineated a number of testable, falsifiable and practical hypotheses. The aim of this study was to analyse the performance of IPv6 versus IPv4 on various operating systems using Jumbo frames. As a result, six operating systems were selected and they were, Microsoft Windows Server 2008, Microsoft Windows Server 2003 and Microsoft Windows 7 Professional. Three operating systems selected from Linux distributions were, Linux Fedora Core, Linux Ubuntu and OpenSUSE. To conduct this study, a traffic generator tool (D-ITG) was selected. This tool was used during the experiments to generate the required traffic for the experiments. After the data collection phase, five metrics were extracted from the collected data for analysis. These metrics were; throughput, one way delay, jitter, packets dropped and the CPU utilisations of the two software routers. Nine Jumbo frame sizes were employed on both IPv6 and IPv4, and TCP and UDP were both employed as transmission protocols. In the following sections the findings obtained from the data analysis in chapter five are discussed.

6.1 TCP Performance on Jumbo Frames with IPv6 and IPv4

After analysing the throughput results for all of the operating systems running on IPv6 using TCP as the transport protocol, it can be said that throughput values are very similar regardless of the platforms, except on some particular frame sizes.

In this section the reasons behind the results presented in section 5.1.1. are explained. The design of the network is a major aspect that influences the behaviour of the protocol; and this design tends to also have direct effects on its performance. The main advantages of TCP can be summarised as follows:

TCP guarantees three things:

- Your data gets there,
- It gets there in order.
- It gets there without duplication (Lay Networks, 2010).

One very important feature of TCP as explained by Ram, Fedeli, Cox & Rixner (2008), “TCP is a connection-based protocol and all communications required that a connection be established first”. That one statement contains all of the three responsibilities stated by Lay Networks earlier. A TCP connection will be established first in order to make sure that data will get there, and that no one else is using that connection, and that the order of the data will not be interrupted.
Based on the graph in Figure 5-1-1, it can be said that, TCP performance on all of the operating systems involved had very similar output with regards to throughput values. It is noted also that throughput values in some particular frame sizes were high for Microsoft Windows 7. It appears that TCP performances have a tendency to react diversely depending on frame sizes. TCP makes sure that data sent by the sender gets to the receiver in order, and without duplications. Due to the fact that TCP is a reliable service and provides features such as congestion control, however, due to the fact that TCP is a reliable service, delays will be introduced whenever a bit error or packet loss occurs. This delay is caused by retransmission of the broken packet, along with any successive packets that may have already been sent (Lay Networks, 2010).

For the frame size 4329 Bytes in particular, all of the operating systems involved behaved unusually and produced a very low performance in terms of throughput values on both IPv6 and IPv4. It is beyond the scope of this study to investigate this matter more in-depth in order to find out the reasons why this particular frame size behaved this way. However out of curiosity, this study conducted further investigations on Microsoft Windows Server 2008 using IPv6 only to find the range of frame sizes along the neighbours' of 4329 behaved in the same manner. Figure 6-1 shows the results of these extra experiments presented in line graphs.

![IPv6 - TCP Throughput Extra Test](image)

Figure 6-1: IPv6 - TCP extra experiments on Microsoft Windows Server 2008

According to Figure 6-1, the frame size in question 4329 Bytes is in the midst of the range that has very unusual low throughput values. As illustrated in the line graph above, it can be said that frame sizes ranging from 4096 Bytes to 4465 Bytes produced very similar values in terms of throughput. The command that was used to generate the necessary traffic for these experiments is listed as:

```bash
0 50 100 150 200 250 300 350 400 450 500
3400 3500 3700 4000 4050 4070 4090 4095 4100 4200 4350 4400 4465 4470 4500
Throughput in Mbps
Jumbo Frame sizes in Bytes
Windows Server 2008
```
This command means that –a defines the receiver’s address which is an IPv6 address, -rp defines the destination port number to use and –C defines how many packets to send every seconds which in this case was 180000 packets. The –t option defines the duration for the test which in this case was the test running for 60 seconds and the –T option defines the protocol to use for sending the packets. Following in Figure 6-2 and Figure 6-3 below presents the results produced by the ITG Decoder.

<table>
<thead>
<tr>
<th>TCP 4095</th>
<th>TCP 4096</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL RESULTS</strong></td>
<td><strong>TOTAL RESULTS</strong></td>
</tr>
<tr>
<td>Number of flows = 60</td>
<td>Number of flows = 60</td>
</tr>
<tr>
<td>Total time = 60,001,200 s</td>
<td>Total time = 40,021,000 s</td>
</tr>
<tr>
<td>Total packets = 80,789</td>
<td>Total packets = 29,379</td>
</tr>
<tr>
<td>Minimum delay = 0.001,000 s</td>
<td>Minimum delay = 0.024,000 s</td>
</tr>
<tr>
<td>Maximum delay = 2,984,000 s</td>
<td>Maximum delay = 0.756,000 s</td>
</tr>
<tr>
<td>Average delay = 0.001,222 s</td>
<td>Average delay = 0.162,110 s</td>
</tr>
<tr>
<td>Average jitter = 0.004701 s</td>
<td>Average jitter = 0.001382 s</td>
</tr>
<tr>
<td>Delay standard deviation = 0.063,986 s</td>
<td>Delay standard deviation = 0.055,204 s</td>
</tr>
<tr>
<td>Bytes received = 323,321,951</td>
<td>Bytes received = 321,979,495</td>
</tr>
<tr>
<td>Average bitrate = 437,532,173,957 kbit/s</td>
<td>Average bitrate = 1,012,905,041 kbit/s</td>
</tr>
<tr>
<td>Average packet rate = 13349.377696 pkt/s</td>
<td>Average packet rate = 492.084434 pkt/s</td>
</tr>
<tr>
<td>Packets dropped = 0 (0.00 %)</td>
<td>Packets dropped = 0 (0.00 %)</td>
</tr>
<tr>
<td>Error times = 0</td>
<td>Error times = 0</td>
</tr>
</tbody>
</table>

Figure 6-2: TCP throughput results on frame size 4095

Figure 6-3: TCP throughput results on frame size 4096

The results shown above in Figure 6-2 are for frame size 4095 Bytes and Figure 6-3 for frame size 4096 Bytes. On these two different frame sizes, the same commands were used to generate the necessary traffic. It can be seen that 60 flows were sent for both frame sizes, and the duration was 60 seconds for both. The average delay and jitter values for the frame size 4096 Bytes was significantly higher when compared to that of frame size 4095 Bytes. As a result, the average packet rate for the frame size 4096 Bytes dropped considerably to just over 492 packets per seconds. When comparing that to over 13349 packets per seconds of frame size 4095 Bytes, another important fact to note here is that the total packets number. The total packets received for frame size 4095 was 801789 while only 29536 received for frame size 4096, which is a difference of 772253 packets. According to Figure 6-2 and 6-3, there is 772253 packets difference. However, Figure 6-3 recorded no packets dropped, and as mentioned earlier in this section, it is out of the scope of this study to go in-depth and investigate the reasons behind those missing packets.

With regards to delay results, the comparison presented in Figure 5-1-6 in Chapter 5 showed that IPv4 on the three Microsoft Windows operating systems generated more delay than IPv6. However, in contrary IPv6 on Linux Fedora and Ubuntu created more delay than IPv4. With OpenSUSE, IPv6 yielded the lowest values in terms of delay while IPv4 on Ubuntu generated the least amount of delay. As a result, the comparison presented in Figure 5-1-6 in chapter 5 concluded that for Jumbo frames employing TCP as transmission protocol, IPv6
created less delay than IPv4. There have been number delay results shown negative delays such as Figure 5-1-4. According to Raisanen, Grotefeld & Morton (2002), RFC 3432 “negative delay value might be useful as part of a stream when trying to discover a distribution of the delay errors.” However, network performance true measurement for delay should be measured in nanoseconds but in this study, delay was measured in milliseconds. Raisanen, Grotefeld & Morton (2002) also stated in RFC 3432 that, “depending on measurement topology, delay values may be as low as 100 usec to 10 msec, whereby it may be important for source and destination to synchronise very closely.” All of the machines employed in this study were synchronised before each test run however, negative delay cannot be avoided.

With regards to jitter results, Stevens (2004) defined jitter as, “timing noise that causes hit errors in high-speed data transmission lines.” If this is the definition of jitter then there may be high jitter values for some of the frame sizes with lower throughput however, Stevens (2004) went on and explained the relationship between jitter and throughput as “if the data rate of a system is increased, the magnitude of jitter measured in seconds is roughly unchanged, but measured as a fraction of a bit period, it increases proportionally with the data rate and causes errors.” The comparison presented in Figure 5-1-9 in chapter 5 showed that on some frame sizes such as frame sizes ranging from 1518 to 3392 Bytes and 5266 Bytes, all of the operating systems engaged in this study generated similar amount of jitter. However, frame size 4329 Bytes in particular, all of the operating systems on both IPv6 and IPv4 produced large amount of jitter. This discussion earlier mentioned the low throughput of frame size 4329 Bytes and now the same frame size had a very high jitter values. The relationship between throughput and jitter as stated by Tektronix (2003) is, “whatever its source, jitter can significantly reduce margin in an otherwise sound design. For example, excessive jitter can increase the bit error rate (BER) of a communications signal by incorrectly transmitting a data bit stream.” In this regard, it explains the point in the graph that frame size 4329 Bytes had very low throughput but very high jitter reading.

With regards to TCP IPv6 CPU utilisations on Router 1, all of the operating systems appeared to behave differently based on frame size. However, based on the graph presented on Figure 5-1-10, Linux distributions showed that they required more of the CPU resources especially OpenSUSE. For instance, OpenSUSE on IPv6 using TCP as transmission protocol on frame sizes 3392 and 9014 Bytes required 43% more of the CPU time than Linux Ubuntu. Thus, Ubuntu on frame sizes 7140 and 8037 Bytes required up to 9% more of the CPU time while all of the Microsoft Windows operating systems showed a very similar pattern on the entire frame sizes except on frame size 2455 Bytes where Microsoft Windows 7 yielded the lowest values in terms of CPU utilisations.
Regarding the CPU utilisations of all operating systems engaged in this study on Router 1 while using TCP on IPv4, OpenSUSE and Linux Ubuntu again performed unsteadily and resulted in high CPU utilisation values. On frame sizes 4329, 5266 and 6203 Bytes, Ubuntu yielded the highest percentage values in terms of CPU utilisations while OpenSUSE required the CPU time the most on the rest of the frame sizes. With regards to the Microsoft Windows based operating systems, they all demonstrated a very similar pattern on most of the frame sizes apart from frame sizes ranging from 7140 to 9014 Bytes where Microsoft Windows Server 2003 used more CPU resources than Microsoft Windows Server 2008 and Microsoft Windows 7. while on frame sizes ranging from 5266 to 9014 Bytes, Microsoft Windows Server 2008 used the least of the CPU resources.

When comparing the CPU utilisations of IPv6 and IPv4 on Router 1 when employing TCP as transmission protocol, all of the Linux distributions on the large frame sizes ranging from 5266 to 9014 Bytes, IPv6 required more of the CPU resources than IPv4 except on frame size 6203 Bytes where OpenSUSE IPv4 used more CPU resources than IPv6. Microsoft Windows Server 2008 and 2003 performed similarly on the two small frame sizes. Microsoft Windows Server 2003 on frame sizes 3392, 5266 and 6203 Bytes, IPv6 required more CPU resources than IPv4 while on frame sizes ranging from 7140 to 9014 Bytes IPv4 used more CPU resources than IPv6 however, between the Microsoft Windows based operating systems, IPv6 on Microsoft Windows Server 2008 IPv6 used more of the CPU resources than IPv4 on all of the Jumbo frame sizes.

On frame sizes 2455, 4329, 5266 and 7140 Bytes, IPv6 and IPv4 on OpenSUSE demonstrated very similar percentages in terms of CPU utilisations however, based on the graph presented in Figure 5-1-12, it can be said that the three Linux distributions on both IPv6 and IPv4 utilised more of the CPU resources than all of the Microsoft operating systems. For instance, OpenSUSE on both IPv6 and IPv4 utilised the CPU resources the most on most of the frame sizes except for frame sizes 7140 and 8037 Bytes where Ubuntu used the most.

Based on the graph in Figure 5-1-12, it can be said that IPv6 on most of the operating systems for most of the frame sizes required more time of the CPU than IPv4 while Fedora on IPv4 on frame size 3392 Bytes in particular, required more of the CPU time than IPv6. Microsoft Windows 7 on IPv4 on frame sizes ranging from 2455 to 6203 Bytes and also 8037 Bytes needed more of the CPU time than IPv6. The next section discusses the CPU utilisations of IPv6 and IPv4 on Router 2 while employing TCP as the transmission protocol.
Regarding the IPv6 CPU usages on Router 2, OpenSUSE performed unsteadily and resulted on a high percentage usage of the CPU resources on most of the frame sizes except for frame sizes 2455 and 7140 Bytes where Linux Ubuntu was the highest in terms of CPU utilisations by 0.5% while all of the Microsoft operating systems demonstrated a very similar pattern throughout for most of the frame sizes apart from frame size 2455 Bytes where Microsoft Windows 7 required less CPU resources than Microsoft Windows Server 2008 and Microsoft Windows 2003. In terms of the operating systems’ CPU utilisations, the three Linux distributions required more of the CPU time on all of the Jumbo frame sizes but Fedora used less CPU resources while OpenSUSE was the highest in terms of CPU utilisations on most of the Jumbo frame sizes.

For the IPv4, OpenSUSE used a very high percentage of the CPU resources on two particular frame sizes, 20% more on 3392 Bytes and 7% more on 8037 Bytes, than Ubuntu, while all of the Microsoft operating systems demonstrated a very similar pattern for most of the frame sizes apart from frame sizes ranging from 7140 to 8037 Bytes where Microsoft Windows Server 2003 used more CPU resources than Linux Fedora in particular.

According to the graph presented in Figure 5-1-15, when comparing IPv6 and IPv4 CPU utilisations, it can be seen that all of the Linux distributions on both IPv6 and IPv4 used more CPU resources than all of the Microsoft Windows operating systems except for frame sizes 7140 and 8037 Bytes where Microsoft Windows Server 2003 IPv4 required more of the CPU resources. IPv6 on all of the Microsoft Windows operating systems required more of the CPU time on most of the frame sizes apart from the large frame sizes ranging from 7140 to 9014 Bytes where IPv4 on Microsoft Windows Server 2003 required more of the CPU time. On the other hand, IPv6 on OpenSUSE required more of the CPU time on all of the frame sizes excluding frame sizes 3392 and 9037 Bytes where IPv4 required 14% more of the CPU time than IPv6 while Microsoft Windows Server 2008 used more of the CPU resources on all of the frame sizes apart from frame size 8037 Bytes where IPv4 used 0.5% more than IPv6.

Based on the graph showed in Figure 5-1-15, it can be said that IPv6 on Fedora on all of the frame sizes consumed the CPU resources more than IPv4 while IPv4 on Linux Ubuntu on frame sizes 3392, 5266 Bytes and larger frame sizes ranging from 7140 to 9014 Bytes utilised more of the CPU time than IPv4. Conversely, on the smaller frame sizes 1518, 2455 and 4329, and 6203 Bytes, IPv6 required more of the CPU resources than IPv4 and IPv6 on OpenSUSE for most of the frame sizes utilised more of the CPU time than IPv4 apart from frame sizes 3392 and 8037 Bytes where IPv4 consumed more of the CPU resources.
As discussed in the above paragraphs, Microsoft Windows operating systems seem to be very stable and consistent in terms of CPU utilisations however, IPv6 on Microsoft Windows Server 2008 and Microsoft Windows 7 required more CPU time on most frame sizes while Microsoft Windows Server 2003 on both IPv6 and IPv4 utilised the CPU resources more, dominated by IPv4 on all frame sizes hence, utilisations never exceed 10% excluding IPv4 on Microsoft Windows Server 2003 on frame sizes 7140 and 8037 Bytes which used 5% more. All of the Linux distributions engaged in this study seem to utilise CPU resources more than Microsoft Windows based operating systems however, IPv6 on Linux Fedora and OpenSUSE required more CPU time while IPv4 on Linux Ubuntu on most frame sizes required more CPU time than IPv6.

6.2 UDP Performance on Jumbo Frames IPv6 and IPv4

According to the analysis of IPv6 performance while employing UDP as the transmission protocol, it can be seen in Figure 5-2-1 that on small frame sizes ranging from 1518 to 3392 Bytes, Linux Fedora outperformed the rest of the operating systems involved. Yet, out of all the Linux based operating systems engaged in this study, frame sizes ranging from 4329 to 9014 Bytes, OpenSUSE outperformed Fedora and Ubuntu by 3%. Regarding the Microsoft Windows based operating systems, frame sizes ranging from 1518 to 3392 Bytes, Microsoft Windows 7 produced the lowest throughput values while Microsoft Windows Server 2003 yielded the lowest values on the largest frame size 9014 Bytes.

In terms of throughput performance on IPv6 while employing UDP as the transmission protocol, it can be concluded that on frame sizes ranging from 1518 to 3392 Bytes, Microsoft Windows Server 2003 and Linux Fedora outperformed the rest of the operating systems. On frame sizes ranging from 4329 to 8037 Bytes, all of the operating systems yielded similar performance while the three Linux distributions outperformed all the Microsoft Windows operating systems on the largest frame size 9014 Bytes.

According to the analysis of IPv4 performance while employing UDP as the transmission protocol, it showed in Figure 5-2-2 that on the three smallest frame sizes ranging from 1518 to 3392 Bytes, Fedora outperformed the rest of the operating systems by up to 9% while on frame sizes ranging from 4329 to 8037 Bytes, all of the operating systems generated very similar performance with regards to throughput values excluding Microsoft Windows Server 2003 which produced the lowest throughput values on all of the frame sizes.

On the largest frame size, OpenSUSE and Ubuntu yielded very similar throughput values and 7.6% more than Fedora which yielded the second highest throughput value. With regards to the Microsoft operating systems, Microsoft Windows 7 performed better than
Microsoft Windows Server 2008 by 3.6% and Microsoft Windows Server 2003 was the lowest in terms of throughput by 8.2% lower than Microsoft Windows Server 2008.

The comparison results of IPv6 and IPv4 presented in Figure 5-2-3 showed that the performance of all of the Linux distributions produced similar values in terms of throughput on frame sizes ranging from 4329 to 8037 Bytes excluding Fedora IPv6 on frame size 8037 in particular, which dropped by 25.6% while both IPv6 and IPv4 on Fedora yielded very similar throughput values excluding the smallest frame size where IPv4 dropped by 7%. Figure 5-2-3 showed that IPv4 on Microsoft Windows Server 2003 produced the lowest throughput values on most of the frame sizes apart from IPv4 on Microsoft Windows Server 2008 on the smallest frame size where throughput values dropped by 10.6% while IPv6 on Microsoft Windows Server 2008 on most of the frame sizes required more of the CPU time than IPv4. However, frame sizes 3392 and 9014 Bytes, IPv4 utilises 2% more of the CPU time than IPv6. Frame sizes, 2455, 6203 and 9014 Bytes, IPv6 required 5% more of the CPU time than IPv4 and IPv6 on Microsoft Windows Server 2003 on all of the frame sizes required 23% more of the CPU resources than IPv4 except for the largest frame size 9014 Bytes where IPv4 utilised more of the CPU time by 3%.

The analysis results of experiments conducted on UDP did not have the same problem and Carvalho, Veiga, Marques, Pacheco and Reis (2010) stated that, "UDP is connectionless, as it sends data without ever establishing a connection". This is one of the main reasons that from the analysis on section 5-2-8 earlier that UDP had packets dropped but not TCP. However, Le,Kuthethoor, Hansupichon, Sésa, Strohm, Hadynski, Kiwior and Parker (2009) rightly stated that, "UDP does not attempt to recover the lost packets nor fix out-of-order delivery problem, resulting in an even higher data loss rate." As a result of this behaviour of UDP, it did not try to retransmit any of the packets instead it just dropped it. Based on the graph presented in Figure 5-2-3, it can be concluded that on frame sizes 2455 and 3392 Bytes, Linux Fedora achieved the highest throughput values of 419.96 Mbps for frame size 2455 Bytes and 422.48 Mbps for frame size 3392 Bytes on Both IPv6 and IPv4.

With regards to delay on IPv6 while using UDP as transmission protocol, the graph presented in Figure 5-2-4 showed that all of the Linux distributions involved in this study produced the similar values in terms of delay on all frame sizes while the Microsoft Windows based operating systems created more delays than the Linux based operating systems however, between the three Microsoft operating systems, Microsoft Windows Server 2003 produced the lowest delay values than the other two operating systems up to a maximum of 20% on the largest frame size 9014 Bytes. Microsoft Windows Server 2008 and Microsoft Windows 7 produced the similar delay values on frame sizes 4329, 5266 and 9014 Bytes.
and on frame size 2455 Bytes, Microsoft Windows 7 produced 8.2% more delays than Microsoft Windows Server 2008. Based on the graph presented in Figure 5-2-4, it can be concluded that on all of the frame sizes, all of the Microsoft Windows operating systems involved in this study created more delays than all of the Linux based operating systems on IPv6 and using UDP as transmission protocol.

With regards to delay on IPv4 while using UDP as transmission protocol, the graph presented in Figure 5-2-5 showed that all of the Linux distributions involved in this study produced the similar and the lowest values in terms of delay on all frame sizes while all of the Microsoft operating systems involved in this study created more delays. Among the three Microsoft operating systems, Microsoft Windows Server 2003 produced less delay values than the other two operating systems by maximum of 22.8% less than Microsoft Windows 7 on the smallest frame size 1518 Bytes and on the smallest frame size, Microsoft Windows Server 2008 produced the highest amount of delays than any of the operating systems involved in this study. However, 3.6% more delays than Microsoft Windows 7 however, on frame sizes ranging from 2455 to 4329 Bytes, and on frame sizes 6203 and 8037 Bytes, Microsoft Windows 7 created more delay than any of the operating systems involved by up to 4.4% more than Microsoft Windows Server 2008. Apart from the smallest frame size 1518 Bytes, frame sizes 5266, 7140 and the largest frame size 9014 Bytes, Microsoft Windows Server 2008 produced the highest delay values than any of the operating systems involved in this study by up to 2% on frame size 5266 Bytes more than Microsoft Windows 7 in particular.

The graph in Figure 5-2-6 presented a comparison of the delay results on both IPv6 and IPv4 while UDP was employed as the transmission protocol. In this comparison, it demonstrated that all of the Linux distributions produced very similar values in terms of delay on all of the frame sizes and also yielded the lowest delay values. Figure 5-2-6 also showed that while IPv6 and IPv4 on Microsoft Windows Server 2003 created similar delay values on frame sizes ranging from 4329 to 8037 Bytes, on frames ranging from 1518 to 3392 Bytes and 9014 Bytes IPv6 created more delays than IPv4. IPv6 on Microsoft Windows Server 2008 on frame sizes ranging from 2455 to the largest frame size 9014 Bytes created up to 3% more delays than IPv4. In this comparison it can be concluded that Microsoft Windows operating systems, IPv6 created more delays on most of the Jumbo frame sizes than IPv4 however, between the three Microsoft Windows operating systems engaged in this study, it can be concluded that Microsoft Windows Server 2008 and Microsoft Windows 7 generated more delays than Microsoft Windows Server 2003 however, IPv6 on Microsoft Windows Server 2003 created more delays than IPv4. It can also be concluded that IPv6 and IPv4 produced similar delay values and also the lowest.
With regards to jitter, Figure 5-2-7 presented the jitter results on UDP as the transmission protocol used on IPv6. Figure 5-2-7 showed that all of the Linux distributions involved in this study produced similar values in terms of jitter on all frame sizes however, all of the Microsoft operating systems created more jitters than the Linux based operating systems. Among the three Microsoft operating systems, Microsoft Windows Server 2003 produced less jitter than the other two Microsoft operating systems to a maximum of 1% on frame size 9014 Bytes while on most of the frame sizes, Microsoft Windows 7 created more jitter than Microsoft Windows Server 2008 apart from frame sizes 6203 Bytes and the largest frame size 9014 Bytes where Microsoft Windows Server 2008 generated more jitter.

Figure 5-2-8 presented the jitter results for IPv4 while using UDP as the transmission protocol. In this graph it showed that all of the Linux distributions produced the very similar jitter values while Microsoft Windows Server 2008 and Microsoft Windows 7 created more jitters than any of the operating systems involved in this study. However, among the three Microsoft operating systems, Microsoft Windows Server 2003 produced less jitter values than the other two Microsoft operating systems while on most of the frame sizes, Microsoft Windows Server 2003 had very similar jitter values to that of all the Linux based operating systems except on frame sizes 4329 and 6203 where Microsoft Windows Server 2003 created more jitters than the Linux based operating systems. However, on the largest frame size 9014 Bytes, Microsoft Windows Server 2003 created less jitter than all of the Linux operating systems.

Figure 5-2-9 presented a comparison of IPv6 and IPv4 jitter results while employing UDP as the transmission protocol. In this presentation, it can be concluded that on both IPv6 and IPv4, Jumbo frames on all of the Linux distributions produced very similar values in terms of jitter on most of the frame sizes except on the two largest frame sizes where OpenSUSE IPv4 generated more jitter. Microsoft Windows Server 2008 on the other hand, IPv4 on most of the frame sizes produced more jitter than IPv6. However, on frame sizes 3392 and 5266 Bytes IPv6 produced more jitter than IPv4. IPv6 on Microsoft Windows 7 on most of the frame sizes produced more jitter than IPv4 while IPv6 on Microsoft Windows Server 2003 on most frame sizes yielded more jitter values than IPv4 however, frame sizes 2455 and 4329 Bytes IPv4 generated more jitter than IPv6.

With regards to the CPU utilisations of IPv6 on Router 1, Figure 5-2-13 presented the CPU usage results when UDP was the transmission protocol used on IPv6. Figure 5-2-13 firstly showed that IPv6 on Microsoft Windows 7 was measured with the lowest values in terms of CPU utilisations on all of the frame sizes. However, Microsoft Windows Server 2008 showed on all frame sizes high utilisations of the CPU resources and on some particular frame sizes...
such as 5266 and 7140 Bytes a 1% more than OpenSUSE which was the second. On the three smallest frame sizes ranging from 1518 to 3392 Bytes, Linux Fedora required less of the CPU time than Microsoft Windows Server 2003 but to a maximum of 1.3% more than Microsoft Windows 7 on some particular frame size however, Linux Ubuntu demonstrated a very stable performance on all of the frame sizes in which Ubuntu required more time of the CPU than Microsoft Windows Server 2003 and Microsoft Windows 7. Ubuntu utilised the CPU more than Linux Fedora on most of the frame sizes apart from frame size 4329 Bytes where Fedora required 0.8% more of the CPU than Ubuntu.

With regards to the CPU utilisations of IPv4 on Router 1, Figure 5-2-14 presented the CPU usage results when UDP was the transmission protocol used on IPv4. Figure 5-2-14 showed that Microsoft Windows 7 required less of the CPU time than any of the operating systems involved in this study while OpenSUSE on IPv4 and UDP utilised the CPU resources the most in fact a 20.6% more. Among the three Linux based operating systems, Fedora produced the lowest value in terms of CPU utilisations which was 16.5% less than OpenSUSE and 4.1% more than Microsoft Windows 7. Linux Ubuntu demonstrated a very steady line on the graph however; it was 7.4% more than Fedora but 16.5% less than OpenSUSE. With regards to the Microsoft Windows operating systems, Microsoft Windows Server 2008 and 2003, they both displayed a very unsteady performance on the graph presented in Figure 5-2-14. For instance, for frame sizes ranging from 1518 to 4329 and 9014 Bytes, Microsoft Windows Server 2003 was lower by up to 5.6% and on the larger frame sizes it was higher by up to 5.3%.

Figure 5-2-14 in chapter 5 compared the CPU utilisations of IPv6 and IPv4 while UDP was employed to be the transmission protocol. According to Figure 5-2-14 it can be concluded that IPv6 on most frame sizes for OpenSUSE and Microsoft Windows Server 2008 required more of the CPU time than IPv4 while on frame sizes 3392 and 8037 Bytes IPv6 and IPv4 yielded similar values in terms of CPU utilisations. Linux Fedora IPv6 on frame sizes ranging from 2455 to 9014 Bytes required up to 9.4% more of the CPU time than IPv4 while on 1518 Bytes IPv4 utilised the CPU resources more than IPv6. Linux Ubuntu and Microsoft Windows 7 there were not much different between IPv6 and IPv4 in terms of CPU usages and Microsoft Windows Server 2003 IPv6 utilised the CPU resources more on frames sizes ranging from 2455 to 4329 Bytes and IPv4 on the rest of the frame sizes required more of the CPU time.

With regards to the CPU utilisations of IPv6 on Router 2, Figure 5-2-16 presented the CPU usage results when UDP was the transmission protocol used on IPv6. Based on Figure 5-2-16, it can be said that OpenSUSE on IPv6 utilised the CPU resources less Linux Ubuntu was clearly the highest user of the CPU resources. For the Microsoft Windows based operating
systems were measured to be high in terms of CPU utilisations on the four small frames sizes while lower on the larger frame sizes with the two largest frame sizes were measured to be the lowest. With regards to IPv4 CPU utilisations on Router 2, Figure 5-2-17 showed that the Microsoft Windows based operating systems on IPv4 were all measured with the lowest values in terms on CPU usages. Linux Ubuntu was clearly measured to be the highest user of the CPU resources while Fedora demonstrated a very steady performance however it was 3.9% higher than Microsoft Windows 7 and 7.1% more than Linux Ubuntu. Figure 5-2-18 presented a comparison of IPv6 and IPv4 CPU utilisations on Router 2. Based on Figure 5-2-18 it can be concluded that IPv6 on Microsoft Windows Server 2008 and Linux Fedora required more of the CPU time than IPv4 while IPv4 on OpenSUSE utilised the CPU resources than IPv6. IPv6 on Microsoft Windows Server 2003 on frame sizes ranging from 1518 to 4329 Bytes and the largest frame size 9014 Bytes utilised the CPU resources than IPv4 while IPv4 on the rest of the frame sizes required more of the CPU time. Linux Ubuntu was clearly the highest user of the CPU resources however, IPv6 on frame sizes 1518, 6203 and 8037 Bytes used the CPU time more while IPv4 on the rest of the frame sizes required more of the CPU time.

As explained earlier in this section, UDP had packets dropped as justified by its behaviour however, White (2001) stated that, “UDP is a flexible, connectionless protocol that does not attempt to ensure the delivery of intact data, i.e., UDP is dubbed “unreliable”.” Following will discussed the results of the packets dropped analysed in chapter five. Figure 5-2-10 presented the results of packets dropped analysed in chapter five. Figure 5-2-10 showed that all of the Linux operating systems engaged in this study were consistent however the packets dropped rates were higher than any of the operating systems involved. Figure 5-2-10 also showed that for the Microsoft Windows based operating systems, Microsoft Windows Server 2008 yielded the lowest values however, on some frame sizes, Microsoft Windows 7 yielded similar values and on the largest frame size Microsoft Windows 7 was the lowest in terms of packets dropped values.

With regards to the packets dropped rate of IPv4 on UDP, Figure 5-2-11 presented the results which showed that Microsoft Windows Server 2003 yielded the highest values in terms of packets lost however, on the largest frame size, Microsoft Windows Server 2003 dropped and yielded the lowest value. The three Linux based operating systems demonstrated a very stable and consistent packets dropped rate however, it was higher than the packets dropped rate of Microsoft Windows Server 2008 and Microsoft Windows 7. When comparing IPv6 and IPv4 packets dropped rate when utilising UDP as the transmission protocol as presented by Figure 5-2-12, it can be concluded that the three Linux based operating systems performance in terms of packets dropped, both on IPv6 and IPv4 were very consistent. However, Figure 5-2-12 showed that IPv4 on Microsoft Windows
Server 2003 dropped the most packets while IPv6 on Microsoft Windows Server 2008 yielded the lowest values in terms of packets dropped.

### 6.3 Comparison of TCP and UDP Performance on Jumbo Frames IPv6 and IPv4

Also in chapter five, Figure 5-2-19 presented a comparison of the performance of the two transmission protocols (TCP & UDP) employed on IPv6. According to Figure 5-2-19, it can be concluded that frame sizes ranging from 1518 to 6203 Bytes and the largest frame size 9014 Bytes, TCP outperformed UDP on all of the operating systems by up to 31% on the smallest frame size in particular. On frame sizes 4329, 7140 and 8037 Bytes, UDP outperformed TCP by 18% excluding frame size 4329 Bytes where UDP outperformed TCP by 77%. However, on frame size 8037 Bytes in particular, TCP on Linux Fedora outperformed UDP by 50.5%.

Also in chapter five, Figure 5-2-20 presented a comparison of the performance of the two transmission protocols (TCP & UDP) employed on IPv4. According to Figure 5-2-20, it can be concluded that excluding frame size 4329 Bytes where UDP outperformed TCP by 53.9%, frame sizes ranging from 1518 to 6203 Bytes and the largest frame size 9014 Bytes, TCP outperformed UDP by up to 43.7% on the smallest frame size in particular however, Linux Fedora on frame sizes 2455 to 6203 Bytes and all of the operating systems on frame sizes 7140 and 8037 Bytes, UDP outperformed TCP by up to 1.6% on frame size 2455 Bytes in particular.

Figure 5-2-21 in chapter five presented a comparison of the performance of the two Internet protocols employed by this study on TCP. According to Figure 5-2-21, it can be concluded that frame sizes ranging from 1518 to 5266 Bytes excluding 4329 Bytes, apart from Microsoft Windows Server 2003, the remaining of the operating systems engaged in this study IPv4 outperformed IPv6 by 2.7% while IPv6 outperformed IPv4 by 9% on Microsoft Windows Server 2003. On frame sizes 6203 and 8037 Bytes, IPv6 outperformed IPv4 by 6.3% however, on frame size 8037 Bytes in particular, IPv4 on Microsoft Windows Server 2003 outperformed IPv6 by 12.4%. On frame size 7140 Bytes, IPv4 outperformed IPv6 by up to a maximum of 15.9% on Linux Fedora however; IPv6 on Linux Ubuntu outperformed IPv4 by 2.9%. On the largest frame size 9014 Bytes, IPv4 on Microsoft Windows Server 2008, Microsoft Windows 7, Linux Ubuntu and OpenSUSE performance was slightly higher than IPv6 however; IPv6 on Microsoft Windows Server 2003 and Linux Fedora outperformed IPv4 by 2.9%.

Also in Chapter five, Figure 5-2-22 presented a comparison of the performance of the two Internet protocols (IPv6 & IPv4) employed on UDP. According to Figure 5-2-22, it can be
concluded that IPv6 on Microsoft Windows Server 2003 outperformed IPv4 by up to 29.2% on all of the frame sizes except for the largest frame sizes where IPv4 outperformed IPv6 by 3.2%. For the Linux operating systems on the smallest frame size IPv6 outperformed IPv4 by 7.2% while on the rest of the frame sizes 3392, 4329, 6203 to 9014 Bytes IPv4 outperformed IPv6 25.8% particularly on Fedora on frame size 8037 Bytes. For Microsoft Windows Server 2008 IPv6 on frame sizes 1518, 2455, 4329 and 6203 to 8037 Bytes outperformed IPv4 by up to 10.8% on the smallest frame size in particular while IPv4 outperformed IPv6 by up to 8.6% on frame size 3392 Bytes. For Microsoft Windows 7, frame sizes 2455, 6203, 8037 and 9014 Bytes IPv6 outperformed IPv4 by up to 0.7% while on the rest of the frame sizes; IPv4 outperformed IPv6 by up to 5.8% on frame size 3392 in particular.

Overall, when TCP and UDP were compared on IPv6 and on IPv4, it was very interesting to find similar results. Excluding frame size 4329 Bytes where throughput values were dropped unsympathetically when TCP was employed, IPv6 on most frame sizes outperformed IPv4 on both TCP and UDP.
6.4 Applications Performance on Jumbo Frames IPv6 and IPv4

This study also had the opportunity to simulate and generate traffic on both IPv6 and IPv4 on the application layer in the experiments. Traffics simulated were DNS, CSa and CSI for games and five different Voice over IP (VoIP) CODECs which was G.711.1, G.711.2, G.723.1, G.729.2 and G.729.3. The metrics were measured in the experiments were Throughput, OWD and Jitter. The CPU utilisations were also measured on both Router 1 and Router 2, added to this were the packets dropped were also recorded.

6.4.1 DNS Performance on Jumbo Frames IPv6 and IPv4

Figure 5-3-4 in chapter five presented a comparison of IPv6 and IPv4 DNS throughput values. According to Figure 5-3-4 it can be concluded that IPv6 and IPv4 on Microsoft Windows Server 2008 and Microsoft Windows 7 yielded similar values while IPv6 outperformed IPv4 on Microsoft Windows Server 2008, Linux Ubuntu and OpenSUSE however, IPv4 on Linux Fedora outperformed IPv6 by 13% and IPv4 on Fedora yielded the highest throughput values overall by 2% more than Microsoft Windows Server 2008 IPv6 and IPv4 which was the second highest.

With regards to the DNS delay on IPv6 and IPv4, Figure 5-3-7 presented a comparison of the DNS delay on both IPv6 and IPv4. Figure 5-3-7 showed that on Microsoft Windows Server 2008 IPv6 generated the highest delay value which was 8.3% higher than IPv4 on Microsoft Windows Server 2008 which was the second highest overall. For the remaining operating systems, IPv4 created more delay than IPv6 with IPv6 on Microsoft Windows Server 2003 being the lowest in terms of delay values by 63.8% lower than IPv6 on Microsoft Windows Server 2008. With regards to jitter values of DNS on IPv6 and IPv4, Figure 5-3-10 showed that IPv6 created more jitter on Microsoft Windows Server 2008 and Linux Fedora while IPv6 and IPv4 generated similar values on Microsoft Windows server 2003 in terms of jitter. However, IPv4 on Microsoft Windows Server 2003, Linux Ubuntu and OpenSUSE yielded more jitter values than IPv6 with IPv4 on OpenSUSE being the highest.

In terms of DNS CPU utilisations on IPv6 and IPv4, Figure 5-3-13 showed that Microsoft Windows Server 2008, Microsoft Windows Server 2003 and Microsoft Windows 7 both IPv6 and IPv4 utilised similar amount of the CPU time on Router 1 however, Microsoft Windows 7 CPU utilisation values was 2.3% higher. IPv6 on Linux Fedora was 4% higher than IPv4 while IPv4 on Linux Ubuntu and OpenSUSE required up to 4% more of the CPU time than IPv6 however, IPv4 on Linux Ubuntu was measured with the highest values in terms of CPU utilisations which was 20% higher than Microsoft Windows Server 2003 which was the lowest. With regards to DNS CPU utilisations on Router 2, Figure 5-3-16 in chapter 5 sections 5-3-10, it can be concluded that Microsoft Windows Server 2008 and Server 2003
IPv6 and IPv4 both yielded similar values while measured to be the lowest values in terms of CPU utilisations. IPv4 on Microsoft Windows 7, Linux Ubuntu and OpenSUSE required more of the CPU resources than IPv6 while IPv6 on Linux Fedora required 2% more of the CPU time than IPv6.

### 6.4.2 Games Performance on Jumbo Frames IPv6 and IPv4

CSa, CSi and Quake3 are the games that were engaged in this study. The goal of the section is to discuss the performance of a network that employed Jumbo frames on IPv6 and IPv4 while engaging with these games. Figure 5-3-19 in chapter 5 sections 5-3-12 presented a comparison of the throughput results on IPv6 and IPv4 on the aforementioned games while engaging them on various platforms. Figure 5-3-19 showed that IPv6 and IPv4 on all of the operating systems engaged in this study yielded similar throughput values on the CSi game however; it was 92.75% lower than the throughput values yielded by Microsoft Windows Server 2008 IPv6 for the game Quake3.

IPv6 outperformed IPv4 on all of the operating systems on the Quake3 game by 5.7% however, all of the operating systems yielded high throughput values on the Quake3 game when compared to CSa and CSi games. CSa game, IPv4 on Microsoft Windows Server 2008 outperformed IPv6 by 43% while IPv6 outperformed IPv4 on Microsoft Windows Server 2008 by 8.5%. IPv4 on Microsoft Windows 7 outperformed IPv6 by 11% while IPv6 on Linux Ubuntu outperformed IPv4 by 8.75%. IPv6 and IPv4 on OpenSUSE yielded similar throughput values while IPv4 outperformed IPv6 on Linux Fedora by 5.25%.

In terms of delay, Figure 5-3-22 concluded that IPv6 on Microsoft Windows server 2008 generated the highest amount of delay which was 16.6% more than IPv4 while IPv6 on Microsoft Windows Server 2003 generated the lowest amount of delay on all of the games involved by 50.4% on the Quake3 game in particular. Figure 5-3-22 also showed that IPv4 on Microsoft Windows Server 2003, Microsoft Windows 7, Linux Fedora and Ubuntu created more delay than IPv6 on the CSa and CSi games however, for OpenSUSE IPv4 generated more delays than IPv6 for the CSi game while IPv6 generated more delay on the CSa and the Quake3 games.

With regards to the jitter results on the three games, Figure 5-3-25 showed that on the Quake3 game, IPv4 generated similar amount of jitter and IPv6 had similar values also on all of the operating systems however, IPv4 generated 57.2% more jitter than IPv6 while IPv6 and IPv4 on the CSi game generated similar amount of jitter on all of the operating systems however it was 28.57% lower when compared to the amount of jitter produced by IPv4 on all of the operating systems on the Quake3 game. In terms of jitter results for the CSa game, IPv6 and IPv4 on Microsoft Windows Server 2003, Linux Fedora, Ubuntu and OpenSUSE
produced similar amount of jitter while IPv6 on Microsoft Windows Server 2008 generated 57.14% more than IPv4 while IPv6 on Microsoft Windows 7 generated 28.57% more jitter than IPv4.

Figure 5-3-28 in Chapter 5 showed a comparison of the three games CPU utilisations on IPv6 and IPv4 on Router 1. Figure 5-3-28 concluded that on the CSa game, IPv4 on Microsoft Windows Server 2008, Microsoft Windows 7 and Linux Ubuntu required more of the CPU time than IPv6 with Microsoft Windows 7 being the highest where IPv4 required 71.62% more than IPv6. IPv6 and IPv4 on Microsoft Windows Server 2003 and OpenSUSE required similar amount of the CPU time however, Microsoft Windows Server 2003 was the lowest in terms of CPU utilisations. For the CSi game, Figure 5-3-28 also concluded that Microsoft Windows Server 2003 was measured with the lowest values in terms of CPU utilisations while IPv6 and IPv4 were measured with similar values on Linux Ubuntu. However, IPv6 on OpenSUSE, Microsoft Windows Server 2008 and Linux Fedora required more of the CPU time than IPv4 while IPv4 on Microsoft Windows 7 utilised more of the CPU resources than IPv6. On the Quake3 game, Figure 5-3-28 showed that IPv4 on Microsoft Windows Server 2008, Microsoft Windows 7 and Linux Ubuntu utilised the CPU resources than IPv6 however, on the other hand, IPv6 on Microsoft Windows Server 2003, Linux Fedora and OpenSUSE required more of the CPU time than IPv4 however, IPv6 on OpenSUSE was measured with the highest values in terms of CPU utilisations while IPv4 on Microsoft Windows Server 2003 was the lowest.

In terms of the CPU utilisations on Router 2, Figure 5-3-31 showed that IPv6 and IPv4 on OpenSUSE required similar amount of the CPU time for game CSa and CSi while IPv6 required 2% more than IPv4 for the Quake3 game. IPv6 on Linux Fedora utilised the CPU resources more than IPv4 on the CSa and CSi games while IPv4 required more than IPv6 on the Quake3 game. IPv4 on Linux Ubuntu utilised the CPU time on all of the games more than IPv6 while IPv6 and IPv4 on Microsoft Windows Server 2008 and Server 2003 were measured with the lowest values in terms of CPU utilisations on all of the games excluding IPv4 on Microsoft Windows Server 2003 on the Quake3 game in which it required 4% more than IPv6. IPv4 on Microsoft Windows 7 on the CSa game was measured to the highest user of the CPU and IPv6 for the Quake3 game.

In terms of the packets dropped for the games, Figure 5-3-34 concluded that, on all of the operating systems for both IPv6 and IPv4, there were no packets dropped recorded for the CSa and CSi games. For the Quake3 game, IPv4 on Microsoft Windows Server 2008, Microsoft Windows 7, Linux Fedora and OpenSUSE dropped up to 4% more packets than
IPv6. However, IPv6 on Microsoft Windows Server 2003 and Linux Ubuntu dropped more packets than IPv4.

6.4.3 VoIP Performance on Jumbo Frames IPv6 and IPv4

There were five VoIP CODECs that were engaged in this study. In this section there is a discussion of the performance of a network that employed Jumbo frames on IPv6 and IPv4 while engaging these VoIP CODECs. Figure 5-3-37 in Chapter 5 sections 5-3-24 presented a comparison of the throughput results on IPv6 and IPv4 on the aforementioned VoIP CODECs while engaging them on various platforms. Figure 5-3-37 showed that on G.711.1 CODEC, IPv6 Microsoft Windows Server 2003, Linux Fedora and Ubuntu outperformed IPv4 by up to 7.2% while IPv4 on Microsoft Windows Server 2008 and Microsoft Windows 7 outperformed IPv6 by up to 11.2% however, on OpenSUSE IPv6 yielded a very low throughput values in which it was 66% lower than the throughput values of IPv4. On the G.711.2 CODEC, IPv6 on Microsoft Windows Server 2008, Linux Ubuntu and OpenSUSE outperformed IPv4 by 8.8% while IPv4 on Microsoft Windows 7, Microsoft Windows Server 2003 and Linux Fedora outperformed IPv6 by 7.6%. All of the operating system yielded similar throughput values on both IPv6 and IPv4 on G.723.1, G.729.2 and G.729.3 CODECs, although it was low but on the G.729.2 CODEC, a 5.6% of throughput values were measured higher when compared to G.723.1 and G.729.3 CODECs.

With regards to the operating systems both IPv6 and IPv4 delays on VoIP CODECs, Figure 5-3-40 in section 5-3-26 showed that IPv6 on Microsoft Windows Server 2008 on all of the CODECs generated more delays than IPv4 while IPv6 on Microsoft Windows Server 2003 generated the lowest values in terms of delay on all of the CODECs. IPv4 on Microsoft Windows 7 and Linux Fedora created more delays than IPv6 while IPv6 on Linux Ubuntu and OpenSUSE created more delays than IPv4 on all of the VoIP CODECs involved. With regards to jitter, according to Figure 5-3-43, excluding OpenSUSE, IPv4 generated more jitter values than IPv6 on the rest of the operating systems while IPv6 and IPv4 measured with similar jitter values on the G.711.1 CODEC. Figure 5-3-43 also showed that on the G.711.2 CODEC, IPv6 on Microsoft Windows Server 2008 2008 generated more jitter than IPv4 while IPv6 and IPv4 on the rest of the operating systems generated similar amount of jitter values. Figure 5-3-43 also showed that for the G.723.1 CODEC, IPv4 on Microsoft Windows Server 2008, Microsoft Windows 7 created more jitter than IPv6 while IPv6 on all of the Linux distributions involved in this study created more jitter than IPv4 however, IPv6 and IPv4 on Microsoft Windows Server 2003 generated similar amount of jitter. On the G.729.2 CODEC, IPv6 on Microsoft Windows 7 generated more jitter than IPv4 while IPv6 and IPv4 generated similar amount of jitter on the rest of the operating systems. On the last CODEC G.729.3, IPv4 on Microsoft Windows Server 2008, Microsoft Windows 7 and Linux Fedora yielded
more jitter values than IPv6 while IPv6 and IPv4 on the rest of the operating systems yielded similar amount of jitter.

With regards to the CPU utilisations of IPv6 and IPv4 on these VoIP CODECs on the first router, Figure 5-3-46 showed that IPv6 and IPv4 on Microsoft Windows Server 2003 on all of the CODECs involved were measured with the lowest values in terms of CPU utilisations however, on the G.711.1 and G.711.2 CODECs IPv6 and IPv4 on Microsoft Windows Server 2008 were also measured with similar values in terms of CPU utilisations to that of Microsoft Windows Server 2003. Not only had that but on the G.723.1 Codec, all of the Microsoft Windows based operating systems involved on both IPv6 and IPv4 were all measured on Router 1 with similar values which were still the lowest in terms of CPU utilisations. IPv4 on Microsoft Windows 7, Linux Ubuntu and OpenSUSE on the G.711.2 Codec required more of the CPU time than IPv6 however, on the same CODEC, IPv6 on Linux Fedora utilised more of the CPU resources than IPv4. On the G.729.2 Codec, IPv6 on Microsoft Windows Server 2008 and Linux Fedora utilised more of the CPU resources than IPv4 while IPv4 on Linux Ubuntu utilised more than IPv6 however, IPv6 and IPv4 on OpenSUSE required similar amount of the CPU time. IPv6 on Microsoft Windows Server 2008 and OpenSUSE on the G.729.3 Codec required more of the CPU time than IPv4 however, IPv4 on Linux Fedora and Ubuntu utilised more of the CPU resources than IPv6 while IPv6 and IPv4 on Microsoft Windows 7 and Microsoft Windows Serve 2003 required similar amount of the CPU time but the still the lowest. On the last CODEC G.711.1, Microsoft Windows Server 2008 and Server 2003 yielded the lowest values in terms of CPU utilisations while IPv6 on Microsoft Windows 7 required more of the CPU time than IPv4 however, IPv4 on Linux Fedora and Ubuntu utilised more of the CPU resources than IPv6 while IPv6 and IPv4 on OpenSUSE yielded similar values in terms of CPU utilisations.

Router 2 on the other hand, Figure 5-3-49 presented a comparison of the CPU utilisations of IPv6 and IPv4 of all the operating systems involved in this study on all of the VoIP CODECs on Router 2. Figure 5-3-49 showed that all of the Microsoft Windows based operating systems were measured on the second router for CODECs G.711.2, G.723.1, G.729.2 and G.729.3 with the lowest and similar values in terms of CPU utilisations for both IPv6 and IPv4. However, on the G.711.1 CODEC, IPv6 and IPv4 on Microsoft Windows Server 2003 required similar amount of the CPU time so as OpenSUSE thus, OpenSUSE yielded similar CPU utilisation values for G.711.1, G.729.2 and G.729.3 CODECs but 7% higher than Microsoft Windows Server 2003.

Figure 5-3-49 also concluded that IPv6 on Linux Fedora required more of the CPU time than IPv4 on the G.711.1, G.711.2 and G.729.2 CODECs than IPv4 however, on the G.723.1 and
G.729.3 CODECs IPv6 and IPv4 on Fedora equally utilised the CPU resources. IPv6 on Linux Ubuntu on the G.711.1 CODEC required more of the CPU time than IPv4 while IPv4 on the G.711.2, G.723.1 and the G.729.3 utilised more CPU resources than IPv6 however. On the G.729.2 CODEC, IPv6 and IPv4 yielded similar values in terms of CPU utilisations.

Figure 5-3-52 in Chapter five sections 5-3-33 presented a comparison of the dropped packets results for both IPv6 and IPv4 on all of the VoIP CODECs engaged in this study. Figure 5-3-52 concluded that all of the operating systems involved in this study dropped no packets on the G.723.1, G.729.2 and the G.729.3 CODECs. For the CODEC G.711.1, IPv4 on Microsoft Windows Server 2008, Microsoft Windows 7, Linux Fedora, Ubuntu and OpenSUSE dropped more packets than IPv6 however, all of the Microsoft Windows based operating systems dropped more packets than the Linux based operating systems with IPv4 on Microsoft Windows 7 dropped the most by 34% more than IPv6 while IPv6 on Microsoft Windows Server 2003 dropped 4% more packets than IPv4. Figure 5-3-52 also concluded that on the G.711.2 CODEC, both IPv6 and IPv4 on all of the Linux based operating systems did not dropped any packets while IPv4 on Microsoft Windows Server 2008 dropped more packets than IPv6 and IPv6 on Microsoft Windows 7 dropped more packets than IPv4 however, IPv6 on Microsoft Windows Server 2003 did not dropped any packets while IPv4 dropped 19% packets.

6.5 Summary of Discussions
In this section, discussions provided earlier in this chapter are summarised. These discussions were based on the findings extracted from the data collected as the results of experiments conducted in the data collection phase of this study. Correspondingly, this section summarises those findings as follows:

- For the TCP analysis, according to a comparison of the IPv6 and IPv4 performance presented in Figure 5-1-3, IPv4 on Microsoft Windows Server 2003 yielded the lowest throughput values of frame sizes ranging from 1518 to 3392 Bytes and frame sizes 5266 and 6203 Bytes while the remaining operating systems on both IPv6 and IPv4 produced similar values. On frame size 4329 Bytes in particular, the throughput values yielded by all of the operating system on both IPv6 and IPv4 dropped by 86.9% completely. On the three larger frame sizes, all of the operating systems on IPv6 yielded similar throughput values but lower than IPv4 on frame size 7140 Bytes, however, in spite of the fact that IPv4 outperformed IPv6 on that particular frame size but the diverse performance of all the operating systems on IPv4.
- Microsoft Windows Server 2008 generated the highest delay on IPv6 and IPv4 on all of the frame sizes however, when comparing them, all of the operating systems on IPv4 generated more delay than IPv6 on all of the frame sizes. It is quite interesting to see
that, when the throughput is pre-eminently low for frame size 4329 Bytes, Figure 5-1-6 showed it to be the highest point for delay so as for jitter as showed in Figure 5-1-9.

- In terms of CPU utilisations, IPv6 on all of the operating systems required more of the CPU time than IPv6 on both Router 1 and Router 2 however, IPv4 on OpenSUSE on frame sizes 3392 and 8037 Bytes unsympathetically utilised the CPU resources more than any of the operating systems.

For the UDP discussions, the summaries are as follows:

- On IPv6 and IPv4 results it showed that, Linux fedora on the three small frame sizes outperformed the rest of the operating system while OpenSUSE yielded the highest throughput values on the larger frame sizes.
- When comparing the throughput results for IPv6 and IPv4, on the smallest frame size 1518 Bytes, IPv6 on Linux Fedora yielded the highest throughput values while IPv4 on Microsoft Windows Server 2008 produced the lowest throughput values.
- On frame sizes 2455 and 3392 Bytes, IPv6 and IPv4 on Linux Fedora yielded the highest throughput values but for the rest of the frame sizes, IPv6 and IPv4 on OpenSUSE slightly outperformed the rest of the operating systems while IPv4 on Microsoft Windows Server 2003 produced the lowest values on the rest of the frame sizes.
- When comparing the delay results for IPv6 and IPv4, the three chosen Microsoft Windows based operating systems generated more delays than the Linux based operating systems.
- With regards to jitter on both IPv6 and IPv4, Microsoft Windows Server 2008 and Microsoft Windows 7 showed a very interesting behaviour, it can be seen that as the frame size increases, the higher the jitter values increases with Microsoft Windows 7 produced more jitter on most of the frame sizes. When comparing IPv6 and IPv4, Microsoft Windows Server 2008 and Microsoft Windows 7 showed the similar pattern with IPv4 yielded slightly higher jitter values than IPv6 while IPv6 and IPv4 generated the lowest jitter values on all of the Linux based operating systems.
- With regards to packets dropped, on IPv6 Linux Fedora and Ubuntu dropped the most packets while Microsoft Windows Server 2008 produced the lowest amount of packets dropped. On IPv4, Microsoft Windows Server 2003 produced the highest amount of packets dropped while Microsoft Windows Server 2008 produced the lowest. When comparing the amount of packets dropped for IPv6 and IPv4, IPv4 on Microsoft Windows Server 2003 dropped the most packets while IPv6 on Microsoft Windows Server 2008 yielded the lowest amount of packets dropped.
- In terms of the CPU utilisations on Router 1, for IPv6 OpenSUSE utilised the CPU resources on most of the frame sizes and Microsoft Windows Server 2008 on some of the frame sizes more than any of the operating systems involved while Microsoft
Windows 7 yielded the lowest percentages on all of the frame sizes. On IPv4, Microsoft Windows 7 still the lowest in terms of CPU utilisation results on all of the frame sizes while OpenSUSE produced the highest the highest percentages on all of the frame sizes. When comparing IPv6 and IPv4 CPU utilisations results, OpenSUSE IPv6 and IPv4 produced the highest values on most of the frame sizes while IPv6 and IPv4 on Microsoft Windows 7 produced the lowest values on all of the frame sizes in terms of CPU utilisations on Router 1.

- For the CPU utilisations on Router 2, on IPv6 Linux Ubuntu on most of the frame sizes produced the highest values while Linux Fedora required more of the CPU time on frame size 5266 Bytes in particular however, OpenSUSE on frame sizes ranging from 1518 to 3392 Bytes, 5266 and 7140 Bytes produced the lowest values and frame size 4329Bytes Microsoft Windows 7 produced the lowest and Microsoft Windows Server 2003 on 8037 and 9014 Bytes produced the lowest values in terms of CPU utilisations. On IPv4, Linux Ubuntu required more of the CPU time than any of the operating systems involved while Microsoft Windows 7 and Microsoft Windows Server 2008 required similar amount of the CPU time but the lowest however, when comparing the CPU utilisations of all the operating systems on IPv6 and IPv4, the IPv6 and IPv4 CPU utilisations on Linux Ubuntu were slightly higher than any other operating systems while IPv6 on OpenSUSE was slightly lower than Microsoft Windows 7 with regards to CPU utilisations on Router 2.

- When comparing the throughput values of the two transmission protocols employed on IPv6, it is quite interesting to see that TCP throughput values were higher than UDP on frame sizes ranging from 1518 to 6203 Bytes excluding frame size 4329 Bytes where all of the operating systems throughput values on TCP dropped significantly but not when UDP is employed. On the larger frame sizes ranging from 7140 to 9014 Bytes, UDP throughput values were higher than TCP throughput values.

- When comparing the throughput values of the two transmission protocols employed on IPv4, TCP also outperformed UDP but only on frame sizes 1518 to 5266 Bytes excluding again 4329 Bytes however, frame sizes ranging from 7140 and 8037 Bytes, UDP outperformed TCP. It is also interesting to see that also on IPv4, the throughput values on all of the operating systems dropped significantly on frame size 4329 Bytes when employing TCP but not when UDP is employed.

- When IPv6 and IPv4 were compared when TCP was employed, it showed that both IPv6 and IPv4 yielded similar throughput values except for IPv4 on Microsoft Windows Server 2003 which yielded the lowest amount of throughput on frame sizes ranging from 1518 to 6203 Bytes. An interesting pattern also illustrated in the results were that both IPv6 and IPv4 on all of the operating systems engaged in this study all dropped in terms of throughput values on frame size 4329 Bytes.
In the comparison of IPv6 and IPv4 on UDP, the results showed that from frame sizes 2455 to 7140 Bytes, IPv4 on Microsoft Windows Server 2003 produced the lowest throughput values while on the smallest frame size IPv6 on Linux Fedora yielded the highest values and on frame size 2455 and 3392 Bytes IPv4 on Fedora produced the highest values. From frame size 4329 to 8037 Bytes excluding IPv4 on Microsoft Windows Server 2003, both IPv6 and IPv4 on all of the operating systems produced similar values with regards to throughput values.

6.5.1 Summary of Applications’ Discussions

In this section the discussions provided earlier in sections 6.4 are summarised. The discussions were based on the findings extracted from the data collected as the results of experiments conducted in the data collection phase of this study. Correspondingly, this section summarised those findings as follows:

- With regards to the DNS throughput, IPv4 on Linux Fedora yielded the highest throughput values. IPv6 and IPv4 on Microsoft Windows Server 2008 produced similar values which also similar to IPv6 on Microsoft Windows Server 2003 which were also the second highest in terms of throughput. IPv6 on Linux Fedora yielded the lowest in terms of throughput values.
- In terms of IPv6 and IPv4 DNS delays, IPv6 on Microsoft Windows Server 2008 generated the highest amount of delays while IPv6 on Microsoft Windows Server 2003 produced the lowest delay values.
- In terms of IPv6 and IPv4 DNS jitter, IPv4 on OpenSUSE created more jitter while IPv6 on Microsoft Windows Server 2003 produced the lowest amount of jitter.
- In terms of IPv6 and IPv4 DNS CPU utilisations on Router 1, IPv4 on Linux Ubuntu required the CPU time the most while IPv6 and IPv4 on Microsoft Windows Server 2003 and IPv6 on Microsoft Windows Server 2008 required similar amount of the CPU time which were also measured to be the lowest in terms of CPU utilisations on Router 1. It was also interesting to see that on both IPv6 and IPv4 on Router 1, all of the chosen Linux based operating systems utilised more of the CPU resources than all of the Microsoft Windows based operating systems.
- CPU utilisations on Router 2, IPv6 and IPv4 on Microsoft Windows Server 2008 and Server 2003 both yielded the lowest values in terms of CPU utilisations while IPv4 on Linux Ubuntu was measured with the highest values. However, IPv4 on Microsoft Windows 7, Linux Ubuntu and OpenSUSE were all yielded higher CPU utilisation values than IPv6 while IPv6 on Linux Fedora utilised more of the CPU resources than IPv4.
- To summarise the discussions on the throughput results of the three games that were engaged in this study, IPv6 on all of the chosen operating systems yielded the highest
throughput values which were slightly higher than IPv4 on the Quake3 game. IPv6 and IPv4 yielded similar throughput values on the CSi game however, it was the lowest.

- IPv6 on Microsoft Windows Server 2008 generated the highest delay values on all of the games which were higher than IPv4 while IPv6 on Microsoft Windows Server 2003 generated the lowest delay values on all of the games which were also lower than IPv4.

- With regards to jitter results of IPv6 and IPv4 on the games, IPv6 on Microsoft Windows Server 2008 generated more jitter than IPv4 on the CSa game while IPv4 on Microsoft Windows Server 2008 and Microsoft Windows 7 generated similar but the lowest amount of jitter. IPv6 on OpenSUSE generated the highest jitter values on the CSi a games while IPv6 and IPv4 on all of the operating systems generated similar amounts with IPv4 on OpenSUSE which were the lowest. IPv4 on all of the operating systems generated similar amounts of jitter which were higher than IPv4 in which all of the operating systems yielded similar jitter values.

- With regards to the CPU utilisations of IPv6 and IPv4 on all of the games on Router 1, IPv4 on Microsoft Windows 7 utilised the CPU resources the most on the CSa game with IPv6 and IPv4 on Microsoft Windows Server 2008 required similar amount of the CPU time but the lowest. On the CSi game, IPv4 on Microsoft Windows Server 2008 and both IPv6 and IPv4 on Microsoft Windows Server 2003 produced similar values in terms of CPU utilisations which were the lowest. However, IPv6 on OpenSUSE on the CSi game yielded the highest values in terms of CPU utilisations. On the Quake3 game, IPv6 on OpenSUSE yielded the highest CPU utilisation values while IPv4 on Microsoft Windows Server 2003 produced the lowest CPU utilisations values.

- With regards to the CPU utilisations of IPv6 and IPv4 on all of the games on Router 2, on the CSa game, IPv4 on Microsoft Windows 7 utilised the CPU resources the most while IPv4 on Microsoft Windows Server 2003 yielded the lowest CPU utilisation values. On the CSi game, IPv4 on Linux Ubuntu produced the highest values in terms of CPU utilisations while IPv6 and IPv4 on Microsoft Windows Server 2008 and Microsoft Windows Server 2003 yielded similar values but the lowest with regards to CPU utilisations. On the Quake3 game, IPv6 on Microsoft Windows 7 produced the highest CPU utilisation values however; it was the highest for the Quake3 game and all of the games while IPv6 on Microsoft Windows Server 2003 produced the lowest CPU usage values on the Quake3 game.

- In terms on IPv6 and IPv4 packets dropped on all of the games, both IPv6 and IPv4 on all of the operating systems on both CSa and CSi games were recorded with no packets dropped however, on the Quake3 game, IPv4 on Microsoft Windows Server 2008 dropped the most packets while IPv4 on Microsoft Windows Server 2008 produced the lowest packets dropped values.
• In terms of throughput values of IPv6 and IPv4 on all of the VoIP CODECs, IPv6 on Microsoft Windows Server 2003 and Linux Ubuntu produced the highest throughput values which were slightly higher than IPv4 on Microsoft Windows 7 while IPv6 on OpenSUSE yielded the lowest throughput values on the G.711.1 CODECs. On the G.711.2 CODEC, IPv6 on OpenSUSE yielded the highest while IPv6 on Linux Fedora yielded the lowest however, IPv6 and IPv4 on all of the operating systems yielded similar throughput values on the G.723.1 CODEC however it was the lowest when compared to the other VoIP CODECs involved in this study.

• In terms of delay, on all of the VoIP CODECs, IPv6 on Microsoft Windows Server 2008 generated the highest amount of delay than any of the other operating systems which were higher than IPv4 however, IPv4 on Microsoft Windows Server 2008 on all of the VoIP CODECs were also high. IPv6 on Microsoft Windows Server 2003 was also measured with the lowest values in terms of delay on all of the CODECs.

• With regards to jitter, on the G.711.1 CODEC, IPv4 on Microsoft Windows Server 2008 and Linux Ubuntu were measured with similar but the jitter values while IPv6 on Microsoft Windows Server 2008, Microsoft Windows Server 2003, Linux Fedora and Ubuntu were measured with similar but the lowest jitter values. IPv6 on Microsoft Windows Server 2008 was measured on the G.711.2 CODEC with the highest jitter values while IPv4 on Microsoft Windows Server 2008 was measured to have similar jitter values with IPv6 and IPv4 on the rest of the operating systems which were the lowest. IPv6 on Microsoft Windows Server 2008 had the lowest jitter values on the G.723.1 CODEC while IPv6 on Linux Fedora was measured with the highest jitter values. IPv6 on Microsoft Windows 7 was measured on the G.729.2 CODEC with the highest jitter values while IPv4 on Microsoft Windows 7 was measured to have similar jitter values with IPv6 and IPv4 on the rest of the operating systems which were the lowest. IPv4 on Microsoft Windows Server 2008, Microsoft Windows 7 and Linux Fedora were measured on the G.729.3 CODEC with the highest jitter values while IPv4 on Microsoft Windows Server 2008, Microsoft Windows 7 and Linux Fedora were measured to have similar jitter values with IPv6 and IPv4 on the rest of the operating systems which were the lowest.

• In terms of CPU utilisations on Router 1, IPv6 and IPv4 on OpenSUSE were measured with the highest CPU utilisation values on the G.711.1 CODEC while IPv6 and IPv4 on Microsoft Windows Server 2003 were measured with the Lowest. On the G.711.2 CODEC, IPv4 on OpenSUSE was measured with the highest while IPv6 and IPv4 on Microsoft Windows Server 2008 and Server 2003 were similarly utilised the CPU resources which were the lowest. On the G.723.1 CODEC, IPv4 on OpenSUSE was measured with the highest while IPv6 and IPv4 on Microsoft Windows Server 2008 and Server 2003 were similarly utilised the CPU resources which were the lowest. On the G.729.2 CODEC, IPv6 and IPv4 on OpenSUSE were measured with the highest while
IPv6 and IPv4 on Microsoft Windows Server 2008 and Microsoft Windows 7 were similarly utilised the CPU resources which were the lowest. On the G.729.3, IPv6 on OpenSUSE was measured with the highest while IPv6 and IPv4 on Microsoft Windows Server 2008 and Microsoft Windows 7 were measured with similar CPU utilisation values which were the lowest.

- In terms of CPU utilisations on Router 2, IPv6 on Linux Ubuntu was measured with the highest CPU utilisation values on the G.711.1 CODEC while IPv6 and IPv4 on Microsoft Windows Server 2003 were measured with the Lowest. On the G.711.2 CODEC, IPv4 on Linux Ubuntu was measured with the highest while IPv6 and IPv4 on Microsoft Windows Server 2008, Server 2003 and Microsoft Windows 7 were similarly utilised the CPU resources which were the lowest. On the G.723.1, IPv4 on Linux Ubuntu was measured with the highest while IPv6 and IPv4 on Microsoft Windows Server 2008, Server 2003 and Microsoft Windows 7 were similarly utilised the CPU resources which were the lowest. On the G.729.2 CODEC, IPv6 and IPv4 on Linux Ubuntu were measured with the highest while IPv6 and IPv4 on Microsoft Windows Server 2008, Microsoft Windows 7 and Microsoft Windows Server 2003 were similarly utilised the CPU resources Which were the lowest. On the G.729.3 CODEC, IPv4 on Ubuntu was measured with the highest while IPv6 and IPv4 on Microsoft Windows Server 2008, Microsoft Windows 7 and Microsoft Windows Server 2003 were measured with similar but the lowest CPU utilisation values.

- With regards to the packets dropped values of IPv6 and IPv4 on all of the VoIP CODECs, there were no packets dropped recorded for all of the chosen operating systems on both IPv6 and IPv4 for the VoIP CODECs G.723.1, G.729.2 and G.729.3 however, on the G.711.2 CODEC, there were no packets dropped recorded for all of the Linux based operating system while IPv4 on Microsoft Windows Server 2003 was recorded with the highest packets dropped values. On the G.711.1 CODEC, IPv4 on Microsoft Windows 7 was recorded with the highest packets dropped values while IPv6 on Linux Fedora yielded the lowest amount of packets dropped.
7.0 Conclusions

This study was conducted based on the network performance of Jumbo frames. Performance was also evaluated on the two main Internet protocols IPv6 and IPv4 when employed by various operating systems. As mentioned in the previous section, results extracted from the data collected from the experiments conducted were based on five main performance metrics namely throughput, jitter, delay, CPU utilisations and the packets dropped rate on IPv6 and IPv4. These metrics were measured on both the two main transmission protocols TCP and UDP and on DNS and five VoIP CODECs. There were six operating systems engaged in this study, three were chosen from the Microsoft Windows operating systems ranged were Microsoft Windows Server 2008, Microsoft Windows Server 2003 and Microsoft Windows 7 Professional and three from the Linux distributions were Linux Fedora, Ubuntu and OpenSUSE. According to the literature analysis in Chapter Two, this area had not been studied or explored prior to this study. After having completed gathering data via conducting experiments and analysing the collected data, this study can be summarised and concluded as follows:

- IPv6 on TCP, Microsoft Windows Server 2008 on 1518, 6203 and 7140 Bytes showed its best performance while Microsoft Windows 7 showed its best performance on 2455 and 3392 Bytes. Out of the Linux based operating systems, OpenSUSE outperformed the rest on 5266, 8037 and 9014 Bytes. All of the operating systems’ performance dropped off unusually at the 4329 Bytes frame size.

- IPv4 on TCP, OpenSUSE showed its best performance on 1518, 3392, 5266, 6203 bytes while Microsoft Windows 7 showed its best performance on 2455, 8037 and 9014 Bytes. An Unusual outcome was that on the 4329 Bytes frame size, performance dropped in all of the operating systems’ performance.

- Comparing IPv6 and IPv4 performance on TCP, IPv4 performances were slightly better than IPv6.

- For IPv6 and IPv4 on UDP, the smaller frame sizes Linux Fedora yielded the best performance while OpenSUSE was measured to be the best for the larger frame sizes.

- According to the comparison of IPv6 and IPv4 on UDP, IPv6 Fedora yielded the best on the smallest frame size while IPv4 was the best on the 2455 and 3392 Bytes frame sizes. On the larger frame sizes, OpenSUSE IPv6 and IPv4 were the best.

- According to the comparison results of TCP and UDP traffics on IPv6, TCP outperformed UDP on frame sizes 1518, 2455, 3392, 5266 Bytes and the highest frame sizes 9014 Bytes however, UDP yielded the highest throughput values on the rest of the frame sizes. Moreover, the same results were found when TCP and UDP traffics were compared on IPv4.
According to the comparison results of IPv6 and IPv4 traffics on TCP, IPv4 performance on OpenSUSE was found to be slightly better than IPv6 performance. It was also interesting to find that the similar results found when IPv6 and IPv4 were compared on UDP.

For the study conducted on the applications level, which involved DNS, Games and VoIP, results can be summarised and concluded as follows:

- IPv6 on DNS, Microsoft Windows Server 2008 and Microsoft Windows Server 2003 yielded similar performance, which was the best and on IPv4, Linux Fedora, which was the highest.
- In a comparison of IPv6 and IPv4 on DNS, IPv4 on Linux Fedora yielded the highest performance while IPv6 and IPv4 yielded similar performance on Microsoft Windows Server 2008 which was also similar to IPv6 on Microsoft Windows Server 2003 and they all had the second highest value.
- For IPv6 on the Games, IPv6 yielded its best performance on the Quake3 game in that all of the operating systems yielded similar performance however; Microsoft Windows Server 2008 performance was slightly higher. Not only that but, a similar outcome was found when games results were compared on IPv4 but Ubuntu and OpenSUSE were slightly higher.
- According to the results obtained when comparing IPv6 and IPv4 on games, IPv6 on all of the operating systems outperformed IPv4 and the highest was on the Quake3 games for all of the operating systems.
- On the G.711.1 CODEC, Microsoft Windows Server 2003 and Linux Ubuntu yielded similar throughput values which were the highest while OpenSUSE produced the lowest which was 73.6% lower.
- On the G.711.2 CODEC, OpenSUSE yielded the highest however, on the G.723.1, G.729.2 and G.729.3 CODECs, throughput values were very low similar to OpenSUSE on the G.711.1 CODEC which was 73.6% lower the throughput values produced by Microsoft Windows Server 2003 and Linux Ubuntu.
- According to the comparison conducted on throughput values of IPv6 and IPv4, for the G.711.1 CODEC, IPv6 outperformed IPv4 on all of the operating systems involved except for Microsoft Windows 7 where IPv4 outperformed IPv6.
Chapter 3 section 3.1 outlined a number of hypotheses for this study. Experiments were conducted in order to collect data, and this data was analysed, all of which helped in agreeing to the following conclusions:

**The main hypothesis:**

"Performance of networks implemented with Jumbo frames gives different performance metrics values depending on transmission protocol, Internet protocol version utilised and the operating system platform employed"

- As mentioned earlier in this document, six operating systems were employed in this study; two transmission protocols TCP and UDP and two Internet protocols IPv6 and IPv4 were involved in the experiments. The findings of this study concluded that this hypothesis is true, the performance of Jumbo frames did perform differently depending on the operating systems implemented on and the transmission protocol and the Internet protocol utilised.

**Hypothesis 1: IPv6 network using Jumbo frames give different network performance values depending on the distribution of Linux operating system implemented.**

- The findings of this study concluded that this hypothesis is true. IPv6 on Linux Fedora on TCP did degrade at a different rate which was 4% higher throughput than Ubuntu.
- With regards to DNS, the findings of this study concluded that this hypothesis is also true as Linux Ubuntu outperformed Fedora by 8.6%.
- With regards to Games, the findings of this study concluded that this hypothesis is true on the CSa game as Linux Ubuntu outperformed Fedora however, on the CSi and the Quake3 games, the findings of this study concluded that this hypothesis is false as Ubuntu and Fedora produced similar performance values.
- With regards to VoIP, the findings of this study concluded that this hypothesis is true on the VoIP CODECs G.711.1 and G.711.2 as Linux Ubuntu outperformed Fedora however, on the G.723.1, G.729.2 and the G.729.3 CODECs, the findings of this study concluded that this hypothesis is FALSE as Ubuntu and Fedora produced similar throughput values.

**Hypothesis 2: IPv6 network using Jumbo frames give different network performance values depending if the operating system is that from Microsoft or a Linux distribution.**

- The findings of this study concluded that this hypothesis is true. On both TCP and UDP, Microsoft Windows operating systems outperformed Ubuntu however; as the frame sizes increased Ubuntu outperformed the Microsoft Windows operating systems on UDP only.
• With regards to DNS, the findings of this study concluded that this hypothesis is true as Linux Ubuntu outperformed Microsoft Windows 7 by 5.8% while the two Microsoft Windows Server operating systems outperformed Linux Ubuntu by 1.9%.

• With regards to Games, the findings of this study concluded that this hypothesis is true on the CSa game as Linux Ubuntu outperformed all the Microsoft Windows based operating systems however, on the CSi and the Quake3 games, the findings of this study concluded that this hypothesis is false as Ubuntu yielded similar throughput values to that of the Microsoft Windows operating systems.

• With regards to VoIP, the findings of this study concluded that this hypothesis is true as the Microsoft Windows operating systems did degrade at different rates to that of Linux Ubuntu.

**Hypothesis 3:** IPv4 network using Jumbo frames give different network performance values depending on the distribution of Linux operating system implemented on. Linux Ubuntu degrades at a different rate to that of a network using Jumbo frames on Linux Fedora;

• The findings of this study concluded that this hypothesis is not true. IPv4 on Linux Fedora with TCP produced similar throughput values to that of Ubuntu for most of the frame sizes. On UDP, it proved the hypothesis to be true as Linux Fedora degraded at a different rate to that of Linux Ubuntu.

• With regards to DNS, the findings of this study concluded that this hypothesis is true as Linux Fedora outperformed Ubuntu by 8.5% IPv4.

• With regards to Games, the findings of this study concluded that this hypothesis is true as Linux Ubuntu outperformed Fedora on the CSa game while Linux Fedora outperformed Ubuntu on the Quake3 game however, on the CSi game, the findings of this study concluded that this hypothesis is false as Ubuntu and Fedora produced similar performance values.

• With regards to VoIP, the findings of this study concluded that this hypothesis is true on as Linux Fedora outperformed Ubuntu however, on the G.729.2 and the G.729.3 CODECs, the findings of this study also concluded that this hypothesis is false as Linux Ubuntu outperformed Fedora.

**Hypothesis 4:** IPv4 network using Jumbo frames give different network performance values depending if the operating system is that from Microsoft or a Linux distribution;

• The findings concluded that this hypothesis is true as Linux Ubuntu did degrade at a different rate to that of all of the Microsoft Windows operating systems employed. Linux Ubuntu outperformed Microsoft Windows Server 2003 on all frame sizes by up to 6.5% on most of the frame sizes. On UDP, the findings concluded that this hypothesis is also
true as Linux Ubuntu outperformed Microsoft Windows Server 2003 by up to 25.6% on all frame sizes.

- With regards to DNS, the findings of this study concluded that this hypothesis is true as Linux Ubuntu degrades at different rates to that of Microsoft Windows operating systems.
- With regards to Games, the findings of this study concluded that this hypothesis is false as Ubuntu yielded similar throughput values to that of the three Microsoft Windows based operating systems.
- With regards to VoIP, the findings of this study concluded that this hypothesis is true on the VoIP CODECs G.711.1, G.711.2 and the G.729.3 as all of the Microsoft Windows based operating systems outperformed Linux Ubuntu while on the G.723.1 and the G.729.2 CODECs, the findings of this study concluded that this hypothesis is false as Ubuntu on the G.723.1 and G.729.2 CODECs produced similar throughput values to that of the three Microsoft Windows based operating systems engaged in this study.

**Hypothesis 5:** IPv6 network using Jumbo frames on Microsoft Windows Server operating systems give different network performance to that when implemented on Linux Fedora.

- The findings of this study concluded that this hypothesis is true. On TCP, on smaller frame sizes, the two Microsoft Windows Server operating systems outperformed Linux Fedora. On the larger frame sizes, Linux Fedora outperformed Microsoft Windows Server 2003 by 4.2% while Microsoft Windows Server 2008 outperformed Fedora by 4.6%. On UDP, Linux Fedora outperformed the two Microsoft Windows Server operating systems while the two Microsoft Windows Server operating systems on larger frame sizes outperformed Linux Fedora by up to 25.9% on frame size 8037 Bytes in particular.
- With regards to DNS, the findings of this study concluded that this hypothesis is true as the two Microsoft Windows Server operating systems outperformed Linux Fedora by 10.6%.
- With regards to Games, the findings of this study concluded that this hypothesis is true on the CSa and the Quake3 games as Linux Fedora degrades at different rates to that of the Microsoft Windows server operating systems however, on the CSI game, the findings of this study concluded that this hypothesis is false as Linux Fedora yielded similar throughput values to that of the two Microsoft Windows Server operating systems.
- With regards to VoIP, the findings of this study concluded that this hypothesis is true on the VoIP CODECs G.711.1, G.711.2 as Microsoft Windows Server 2003 outperformed Fedora while on the G.723.1, G.729.2 and the G.729.3 CODECs, the findings of this study concluded that this hypothesis is false as Linux Fedora produced similar throughput values to that of the two Microsoft Windows Server operating systems.
Hypothesis 6: IPv4 network using Jumbo frames on Microsoft Windows Server operating systems give different network performance to that when implemented on Linux Fedora.

- The findings of this study concluded that this hypothesis is true. On both TCP and UDP on IPv4, on the smaller frame sizes, Linux Fedora outperformed Microsoft Windows Server 2003 while Microsoft Windows Server 2008 outperformed Fedora. On frame size 7140 Bytes, Fedora outperformed the two Microsoft Windows Server operating systems and on frame size 8037 Bytes Microsoft Windows Server 2003 outperformed Fedora and Microsoft Windows Server 2008.
- With regards to DNS, the findings of this study concluded that this hypothesis is true as Linux Fedora outperformed the two Microsoft Windows Server operating systems on a network employing IPv4.
- With regards to Games, the findings of this study concluded that this hypothesis is true as the operating systems degrades at a different rates to each other however, on the CSi game, the findings of this study concluded that this hypothesis is false as Fedora yielded similar throughput values to that of the two Microsoft Windows Server operating systems.
- With regards to VoIP, the findings of this study concluded that this hypothesis is true on the VoIP CODECs G.711.1, G.711.2, G.729.2 and the G.729.3 CODECs. On the G.723.1 CODEC, the findings of this study concluded that this hypothesis is false as Fedora produced similar throughput values to that of the two Microsoft Windows Server operating systems engaged in this study.
7.1 Further Study

This research presented a comparison study conducted based on the network performance of Jumbo frames. There were five main performance metrics extracted from the data collected namely throughput, jitter, delay, CPU utilisations and the packets dropped rate on IPv6 and IPv4. These metrics were measured on both the two main transmission protocols TCP and UDP and on DNS and five VoIP CODECs. There were six operating systems engaged in this study, three were chosen from the Microsoft Windows operating systems ranged were Microsoft Windows Server 2008, Microsoft Windows Server 2003 and Microsoft Windows 7 professional and three from the Linux distributions were Linux Fedora, Ubuntu and OpenSUSE.

The findings and outcome of this study can be a valuable tool for network and Information Technology professionals as a reference guide in making well informed decisions when it comes to engaging Jumbo frames in a network together with any of the operating systems involved in this study. This performance comparison study can be extended by:

- Conducting a thorough research into why throughput values of frame sizes ranging from 4096 to 4465 Bytes dropped but were high in jitter and delay
- Using the same set up but limit the MTU to 1518 and compare the performance with the data collected from this study
- Conduct the same study on hardware routers
- Conduct the same study but on wireless.

As mentioned earlier in this document, Jumbo frames is still an emerging technology therefore, further studies in this arena is required to evaluate the performance of Jumbo frames.
References:


Conference on Network Applications Protocols and Services (NETAPPs), (pp. 90-94). Kedah: IEEE.


