

UNITEC New Zealand

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

**Performance evaluation of IP version 4 and IP
version 6 transition mechanisms on various
operating systems**

Name: Sotharith Tauch

Abstract

Internet Protocol version 6 is the Next Generation Internet Protocol developed by Internet Engineering Task Force to substitute the current Internet Protocol version 4. The reason of this substitution is the exhaustion of IPv4 address space. However, it is not possible to migrate from IPv4 to IPv6 in a short period due to the size and complexity of the Internet infrastructure. Thus, IPv4 will coexist with IPv6 for a long time before the entire Internet infrastructure can fully migrate to IPv6. IETF Next Generation Transition Working Group (NGtrans) developed IPv4/IPv6 transition mechanisms that help IPv4 and IPv6 coexist on the Internet during the migration period.

The purpose of this research is to evaluate performance of two tunnelling mechanisms (Configured Tunnel and 6to4 tunnelling mechanisms) operate on four selected operating systems (Windows Server 2003, Windows Server 2008, Ubuntu 9.10, and Fedora Core 11). This performance measurement research examined on two types of transmission protocols namely UDP (User Datagram Protocol) and TCP (Transmission Control Protocol). The result of this research focused on four metrics such as throughput, delay, jitter, and CPU utilization. The experiments conducted using different payload sizes, ranging from 64 bytes to 1536 bytes.

Results of this experimental research indicated that, Configured Tunnel and 6to4 perform differently on Windows Server 2003, Windows Server 2008, Ubuntu 9.10, and Fedora 11. By using TCP as transport protocol, Configured Tunnel on Fedora 11 produced the highest throughput. However, it also produced a very high delay as compared to Ubuntu 9.10, Windows Server 2003, and Windows Server 2008. On the other hand, after measuring UDP traffic, the results indicated that 6to4 on Ubuntu 9.10 produced the highest throughput with the lowest delay, which designate as the best choice for video and voice traffics.

Acknowledgements

I would like to acknowledge for all help and support that I received from postgraduate programmes director Donald Joyce. I would like to acknowledge a special thanks to my supervisors Shaneel Narayan and Hira Sathu for their time, ideas, guidance and support, throughout this research study.

My family is a big part of this achievement. I would like to thank you to all my family members who have given me courage and support throughout the duration of my Master of Computing study and particularly during the period of this research.

Without people mentioned above, I would not have completed my thesis and Master of Computing from Unitec.

Table of Contents

ABSTRACT	I
ACKNOWLEDGEMENTS	II
LIST OF TABLES	VI
LIST OF FIGURES	VII
1 CHAPTER 1: INTRODUCTION.....	1
1.1 STRUCTURE OF THE REPORT	2
2 CHAPTER 2: LITERATURE REVIEWS.....	3
2.1 INTERNET PROTOCOL VERSION 4 (IPv4).....	3
2.1.1 <i>Consequences of the limited IPv4 address space</i>	4
2.2 INTERNET PROTOCOL VERSION 6 (IPv6).....	5
2.3 COMPARISON OF IPv4 AND IPv6 FEATURES	7
2.4 TRANSITION MECHANISMS.....	9
2.4.1 <i>Dual Stack</i>	9
2.4.2 <i>Configured Tunnel</i>	10
2.4.3 <i>6over4 Tunnelling Mechanism</i>	11
2.4.4 <i>6to4 Tunnelling Mechanism</i>	11
2.4.5 <i>Intra-Site Automatic Tunnel Addressing Protocol</i>	12
2.4.6 <i>Network Address Translation-Protocol Translation</i>	13
2.5 PERFORMANCE METRICS	14
2.5.1 <i>Throughput</i>	14
2.5.2 <i>Delay</i>	14
2.5.3 <i>Jitter</i>	14
2.5.4 <i>CPU Utilization</i>	15
2.6 EVALUATION OF PERFORMANCE MEASUREMENT TOOLS	15
2.6.1 <i>iPerf</i>	15
2.6.2 <i>Netperf</i>	16
2.6.3 <i>IP Traffic</i>	17
2.6.4 <i>Distributed Internet Traffic Generator (D-ITG)</i>	18
2.6.5 <i>Selection of Tool</i>	18
2.7 RELATED STUDIES	19
2.7.1 <i>Study 1</i>	19
2.7.2 <i>Study 2</i>	22
2.7.3 <i>Study 3</i>	24
2.7.4 <i>Study 4</i>	29
2.7.5 <i>Study 5</i>	31
2.8 IDENTIFIED GAPS	34
2.9 LITERATURE MAP.....	36
2.10 CHAPTER SUMMARY.....	37

3	CHAPTER 3: METHODOLOGY	38
3.1	HYPOTHESIS	38
3.2	METHOD USED FOR STUDY	38
3.3	DATA COLLECTION.....	39
3.4	LITERATURE REVIEW PROCESS.....	40
3.5	EXPERIMENTAL DATA GATHERING PROCESS.....	40
3.6	CHAPTER SUMMARY.....	42
4	CHAPTER 4: EXPERIMENTAL DESIGN.....	43
4.1	HARDWARE SPECIFICATIONS	43
4.2	SOFTWARE SPECIFICATION.....	44
4.3	NETWORK DESIGN.....	44
4.4	NETWORK SETUP AND CONFIGURATION.....	45
4.4.1	<i>Configured Tunnel mechanism on Windows Server 2003.....</i>	<i>45</i>
4.4.2	<i>6to4 mechanism on Windows Server 2003.....</i>	<i>47</i>
4.4.3	<i>Configured Tunnel mechanism on Windows Server 2008.....</i>	<i>48</i>
4.4.4	<i>6to4 mechanism on Windows Server 2008.....</i>	<i>49</i>
4.4.5	<i>Configured Tunnel mechanism on Ubuntu 9.10.....</i>	<i>50</i>
4.4.6	<i>6to4 mechanism on Ubuntu 9.10</i>	<i>51</i>
4.4.7	<i>Configured Tunnel mechanism on Fedora Core 11</i>	<i>52</i>
4.4.8	<i>6to4 mechanism on Fedora Core 11</i>	<i>53</i>
4.5	CHAPTER SUMMARY.....	54
5	CHAPTER 5: DATA ANALYSIS.....	55
5.1	TCP DATA ANALYSIS.....	55
5.1.1	<i>TCP Throughput.....</i>	<i>55</i>
5.1.2	<i>TCP Jitter.....</i>	<i>65</i>
5.1.3	<i>TCP Delay.....</i>	<i>66</i>
5.1.4	<i>TCP - Router1 CPU Utilization</i>	<i>67</i>
5.1.5	<i>TCP - Router2 CPU Utilization</i>	<i>69</i>
5.2	UDP DATA ANALYSIS	70
5.2.1	<i>UDP Throughput.....</i>	<i>70</i>
5.2.2	<i>UDP Jitter.....</i>	<i>76</i>
5.2.3	<i>UDP Delay.....</i>	<i>78</i>
5.2.4	<i>UDP Router1 CPU Utilization</i>	<i>79</i>
5.2.5	<i>UDP Router2 CPU Utilization</i>	<i>80</i>
5.3	COMPARISON OF TCP AND UDP	81
5.3.1	<i>UDP and TCP Throughput</i>	<i>81</i>
5.3.2	<i>UDP and TCP Jitter.....</i>	<i>83</i>
5.3.3	<i>UDP and TCP Delay.....</i>	<i>84</i>
5.3.4	<i>UDP and TCP Router 1 CPU Utilisation</i>	<i>85</i>
5.3.5	<i>UDP and TCP Router 2 CPU Utilisation</i>	<i>86</i>
5.4	SUMMARY.....	86
6	CHAPTER 6: DISCUSSION.....	87

6.1	DISCUSSION OF FINDINGS.....	87
6.1.1	<i>TCP Performance</i>	87
6.1.2	<i>UDP Performance</i>	89
6.1.3	<i>UDP and TCP Comparison</i>	90
6.1.4	<i>Summary of Findings</i>	90
6.2	FUTURE STUDY.....	92
7	CHAPTER 7: CONCLUSION.....	93
	APPENDICES	95
	APPENDIX A: TCP THROUGHPUT RESULTS	95
	APPENDIX B: TCP JITTER RESULTS.....	96
	APPENDIX C: TCP DELAY RESULTS.....	97
	APPENDIX D: ROUTER1 - TCP CPU UTILISATION RESULTS.....	98
	APPENDIX E: ROUTER2 - TCP CPU UTILISATION RESULTS.....	99
	APPENDIX F: UDP THROUGHPUT RESULTS.....	100
	APPENDIX G: UDP JITTER RESULTS.....	101
	APPENDIX H: UDP DELAY RESULTS.....	102
	APPENDIX I: ROUTER1 - UDP CPU UTILISATION RESULTS.....	103
	APPENDIX J: ROUTER2 - UDP CPU UTILISATION RESULTS.....	104
	REFERENCES.....	105

List of Tables

Table 2-1: Classes of IPv4.....	3
Table 2-2: Internet host count history.....	4
Table 2-3: Comparison of IPv4 and IPv6 features.....	8
Table 2-4: Hardware specifications.....	20
Table 2-5: The differences between BDMS and DSTM.....	23
Table 2-6: Comparisons of tunnel-based mechanisms.....	24
Table 2-7: TCP connection time.....	30
Table 2-8: Web client/Server simulation tests.....	31
Table 2-9: Parameters used in simulation cases.....	32
Table 2-10: Summary of related work.....	34
Table 4-1: Hardware specification.....	43

List of Figures

Figure 2-1: Graph of the evolution of the Internet hosts	5
Figure 2-2: IPv4 and IPv6 headers (Sailan, Hassan, and Patel, 2009)	7
Figure 2-3: Dual IP Stacks	9
Figure 2-4: Configured Tunnel Network Infrastructure	10
Figure 2-5: 6to4 address architecture.....	11
Figure 2-6: 6to4 Network Infrastructure	12
Figure 2-7: NAT-PT Network Infrastructure	13
Figure 2-8: IPerf components.....	16
Figure 2-9: ISATAP test-bed configuration (Visoottiviseth & Bureenok, 2008)	20
Figure 2-10: Configured Tunnel test-bed (Chen et al., 2004)	25
Figure 2-11: 6to4 Tunnel test-bed	26
Figure 2-12: Tunnel Broker test-bed (Chen et al., 2004)	26
Figure 2-13: Perfmon CPU performance monitoring tool.....	28
Figure 2-14: Network Setup (Raicu & Zeadally, 2003)	29
Figure 2-15: Network Model.....	32
Figure 3-1: Sample D-ITG total result.....	41
Figure 3-2: Sample Line Chart.....	41
Figure 4-1: Network design.....	44
Figure 5-1: Configured Tunnel and 6to4 on Windows Server 2008 – TCP Throughput	55
Figure 5-2: Configured Tunnel and 6to4 on Windows Server 2003 – TCP Throughput	57
Figure 5-3: Configured Tunnel and 6to4 on Ubuntu 9.10 – TCP Throughput	58
Figure 5-4: Configured Tunnel and 6to4 on Fedora 11 – TCP Throughput	60
Figure 5-5: TCP Throughput (Mbps)	61
Figure 5-6: TCP Throughput from packet size 64 bytes to 128 bytes	62
Figure 5-7: TCP Throughput from packet size 128 bytes to 768 bytes.....	62
Figure 5-8: TCP Throughput from packet size 768 bytes to 1536 bytes.....	64
Figure 5-9: TCP Jitter	65
Figure 5-10: TCP Delay	67
Figure 5-11: Router 1 - CPU Utilisation of TCP performance	68
Figure 5-12: Router 2 - CPU Utilisation of TCP performance	69
Figure 5-13: Configured Tunnel and 6to4 on Windows Server 2008 – UDP Throughput	71
Figure 5-14: Configured Tunnel and 6to4 on Windows Server 2003 - UDP Throughput.....	72
Figure 5-15: Configured Tunnel and 6to4 on Ubuntu - UDP Throughput	73
Figure 5-16: Configured Tunnel and 6to4 on Fedora 11 – UDP Throughput.....	74
Figure 5-17: UDP Throughput Result	75
Figure 5-18: UDP Jitter Result	77
Figure 5-19: UDP Average Delay Result	78
Figure 5-20: Router 1 – CPU Utilisation of TCP performance.....	79
Figure 5-21: Router 2 – CPU Utilisation of TCP performance.....	80
Figure 5-22: UDP and TCP Throughput Result.....	81
Figure 5-23: UDP and TCP Jitter	83
Figure 5-24: UDP and TCP Delay	84
Figure 5-25: TCP and UDP Router 1 CPU Utilisation.....	85
Figure 5-26: TCP and UDP Router 2 CPU Utilisation.....	86

List of Abbreviation

ALG	Application Layer Gateway
ATM	Asynchronous Transfer Mode
BDMS	Bi-Directional Mapping System
BSD	(Tools)
CLI	Command Line Interface
CPU	Central Processing Unit
D-ITG	Distributed Internet Traffic Generator
DLPI	(Tools)
DSTM	Dual Stack Tunnelling Mechanism
EED	End-to-End Delay
IEEE	Institute of Electrical and Electronic Engineer
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISATAP	Intra-Site Automatic Tunnel Addressing Protocol
ISDN	Integrated Services Digital Network
LAN	Local Area Network
Mbps	Megabit per second
MLD	Multicast listener Discovery
ms	Millisecond
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NAT-PT	Network Address Translation – Protocol Translator
NGtrans	Next Generation Transition Working Group
NIC	Network Interface Card
pps	Packet per second
RFC	Request For Comment
RTT	Round Trip Time
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WAN	Wide Area Network
WLAN	Wireless Local Area Network

1 Chapter 1: Introduction

This research focuses on the differences in network performance of various IP version 4 and IP version 6 transition mechanisms used over various Operating Systems. TCP/IP is a protocol suite that allow the Internet operate across geographical areas. IP is one of the protocols within TCP/IP protocol suite and this protocol was begun with version 4 which is known as Internet Protocol version 4 (IPv4). Internet Protocol is the standard protocol being used on the Internet which allows computers to be able to communicate in order to exchange information such as data, voice (VoIP), and video (Video conference). Internet Protocol version 4 (IPv4) is the current internet protocol that is widely used across the Internet, but in the near future, there exist issues like insufficient public Internet Protocol version 4 address space that does not allow the growth of the Internet. Nowadays, most of mobile devices are required to have an IP address to connect to the Internet which leads to high consumption of IP address. Internet Engineer Task Force has considered this issue and proposed a new version of Internet Protocol namely Internet Protocol Version 6 (IPv6).

Internet Protocol version 6 (IPv6) is the solution to the massive growth of the Internet due to the size of the address spaces. IPv6 addressing contains 128 bits binary value that provide 2^{128} addresses. In the near future the current Internet Protocol version 4 (IPv4) will slowly migrate to Internet Protocol version 6 (IPv6). Sailan, Hassan, and Patel (2009) state that “Currently IPv6 network penetration is still low but it is expected to grow, while IPv4 address pool is projected by Regional Internet Registry to be exhausted by the end of 2011”. During the migration period there will be compatibility and interoperability issues relating to Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6) because Internet Protocol version 6 (IPv6) is not backward compatible with Internet Protocol version 4 (IPv4). According to Govil, Govil, Kaur, and Kaur (2008):

The transition between IPv4 internet and IPv6 will be a long process as they are two completely separate protocols and it is impossible to switch the entire internet over to IPv6 over night. IPv6 is not backward compatible with IPv4 and IPv4 hosts and routers will not be able to deal directly with IPv6 traffic and vice-versa. As IPv4 and IPv6 will co-exist for a long time, this requires the transition and inter-operation mechanisms.

Migrating from IPv4 to IPv6 is a complicated task that cannot be done overnight. The size and complexity of the Internet cause this migration task to become enormously difficult and

time consuming. Next Generation Transition (NGtrans) proposed three main transition mechanisms that included dual stack, tunnelling, and translation (Waddington & Chang, 2002). These solution allow Internet Protocol version 4 (IPv4)to be able to coexist with Internet Protocol version 6 (IPv6) during the migration period.

The main target of this research is to study the performance of different transition mechanisms when implemented on various operating systems such as Windows server 2003, Windows server 2008, Linux Ubuntu 9.10, and Linux Fedora Core 11. The result of this research will discuss later in this report. Next section will be presenting the structure of this report.

1.1 Structure of the Report

There are total of seven chapters included in this report. Chapter one contains the introduction which briefly describes the overview of Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6) and then leads to the transition mechanisms that overcome the issue of interoperation between IPv4 and IPv6 during the transition period. Chapter two presents the literature review, which contains information regarding IPv4, IPv6, transition mechanisms in detail and related studies conducted by different researchers.

Chapter three covers the research hypothesis, methodology used in this research, and the method of data collection. Chapter four covers details of experimental design covering the specification of hardware and software used in experiments, network diagram used to simulate the design of the network environment, and detail of network configuration. Chapter fivecovers the analysis of data gathered from the experiment. These data are presented in line charts with the discussion of each chart. Chapter six presents in depth discussion of the findings from this research. Chapter seven concludes this research. Next chapter will be introducing literature reviews.

2 Chapter 2: Literature Reviews

This chapter will be looking at IPv4 and IPv6 in detail in order to identify the problems, opportunities and the differences between the two versions of Internet Protocol. This chapter will be also looking at the coexistence of IPv4 and IPv6 by analysing previous researches and finding gaps in the literature.

2.1 Internet Protocol Version 4 (IPv4)

IPv4 is the protocol used extensively on the Internet. All communication across the Internet currently relies on IPv4 protocol. In order to understand this protocol in more detail, first we need to look at the address scheme. IPv4 addressing contains four octets and each octet represents 8 bits of a binary number. The entire address space of IPv4 contains 32 bits of binary number, which mean IPv4 has 2^{32} addresses that are equivalent to 4,294,967,296 different addresses. According to Cisco (2007-2009), IPv4 contains four classes of address, which shows in Table 2-1 below.

Class	High Order Bits	Start	End
Class A	0	0.0.0.0	127.255.255.255
Class B	10	128.0.0.0	191.255.255.255
Class C	110	192.0.0.0	223.255.255.255
Multicast	1110	224.0.0.0	239.255.255.255

Table 2-1: Classes of IPv4

According to Jivesh, Govil, and Govil (2007) IPv4 address is written in dot decimal notation and it contains three types of address, which include unicast, broadcast, and multicast address.

2.1.1 Consequences of the limited IPv4 address space

As discussed above, IPv4 contains 4,294,967,296 addresses. The number of hosts on the Internet increase dramatically every year, which has led to the exhaustion of IPv4 address space. Table 2-2 below was taken from Internet Systems Consortium (2001-2009), shows the increasing number of hosts on the Internet each year starting from year 1981 to year 2009.

Date	Hosts	Date	Hosts	Date	Hosts
08/1981	213	07/1992	992,000	01/2000	72,398,092
05/1982	235	10/1992	1,136,000	07/2000	93,047,785
08/1983	562	01/1993	1,313,000	01/2001	109,574,429
10/1984	1,024	04/1993	1,486,000	07/2001	125,888,197
10/1985	1,961	07/1993	1,776,000	01/2002	147,344,723
02/1986	2,308	10/1993	2,056,000	07/2002	162,128,493
11/1986	5,089	01/1994	2,217,000	01/2003	171,638,297
12/1987	28,174	07/1994	3,212,000	01/2004	233,101,481
07/1988	33,000	10/1994	3,864,000	07/2004	285,139,107
10/1988	56,000	01/1995	4,852,000	01/2005	317,646,084
01/1989	80,000	07/1995	6,642,000	07/2005	353,284,187
07/1989	130,000	01/1996	9,472,000	01/2006	394,991,609
10/1989	159,000	07/1996	12,881,000	07/2006	439,286,364
10/1990	313,000	01/1997	16,146,000	01/2007	433,193,199
01/1991	376,000	07/1997	19,540,000	07/2007	489,774,269
07/1991	535,000	01/1998	29,670,000	01/2008	541,677,360
10/1991	617,000	07/1998	36,739,000	07/2008	570,937,778
01/1992	727,000	01/1999	43,230,000	01/2009	625,226,456
04/1992	890,000	07/1999	56,218,000		

Table 2-2: Internet host count history

The table above shows that from 1981 to 1991 the numbers of hosts increased from 213 to 617,000 hosts, which mean within ten years there was an increase of 616,787 hosts. From 1991 to 2001, the numbers of host increased from 617,000 to 109,574,429 hosts, which

mean there were 109,574,429 hosts increased. From 2001 to 2009, the numbers of host increased from 109,574,429 to 625,226,456 hosts, which mean there was an exponential growth of 515,651,027 hosts. Figure 2-1 below shows the exponential growth of the Internet hosts which is according to the information from Table 2-2 above.

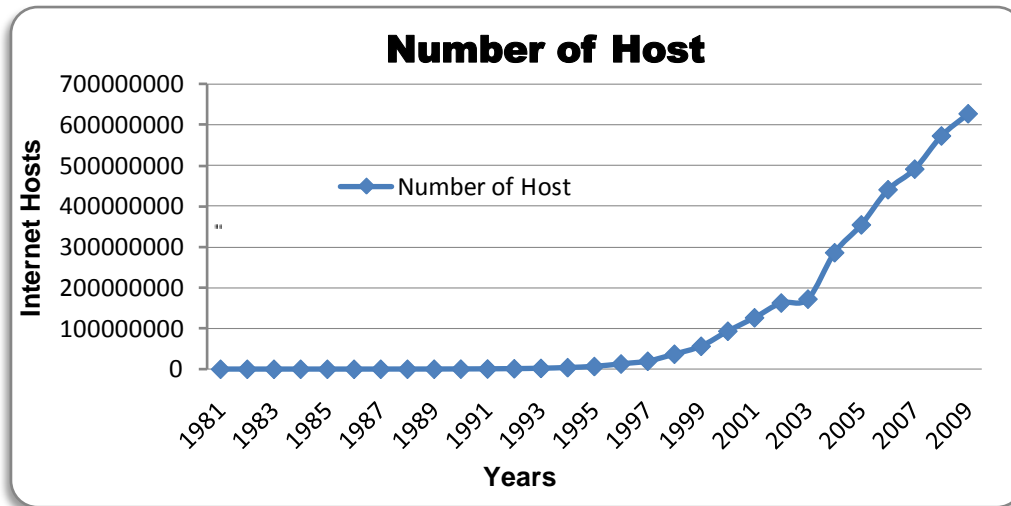


Figure 2-1: Graph of the evolution of the Internet hosts

Year after year, the numbers of hosts on the Internet keep increasing significantly due to the rapid grow of technology and number of people in large population countries began to access the Internet thus high demand of IP4 is bound to cause shortage of IPv4 address space (Grosse & Lakshman, 2003). Due to this reason, IETF proposed a solution to overcome the exhaustion of IPv4 address, which known as Internet Protocol Next Generation (IPng) or Internet Protocol version 6 (IPv6). Clear evidence from the above graph shows that, from the year 1997 to 2009 the growth of the Internet host has been asymptotic. Using the above graph it will not be very long for the existing IPv4 address space to be exhausted. Next section, discussed IPv6 the next generation Internet protocol that not only overcome address issues but also enhances other features.

2.2 Internet Protocol Version 6 (IPv6)

Internet Protocol version 6 (IPv6) is the new version of the Internet Protocol which was designed to overcome the shortcoming of IPv4. Green, Fiuczynski, & Jankiewicz (2006)

stated that *“IPv6 was designed to incorporate all of the patches, changes, and best practices developed from over twenty years of IPv4 Internet engineering into a new next-generation protocol to support the expansive growth of Internet communications and applications”*. The development of IPv6 is not just resolving the address space but also provide better performance and improvement over IPv4 (Wang, Ye, & Li, 2005).It has been almost two decades that IETF NGtrans had proposed IPv6. According to Hiromi and Yoshifuji (2006) *“IPv6 had been proposed at IETF as the next generation of Internet Protocol at early in the 1990s and it is now ready for practical use after trial phase”*. Both IPv4 and IPv6 have different addressing format, as IPv4 addressing format is written in decimal notation and IPv6 addressing format is written in hexadecimal notation(Govil& Govil, 2007).IPv6supports unicast, anycast, and multicast address. On the other hand, IPv4 support unicast, anycast, and broadcast address.

Broadcast address of IPv4 is not available in IPv6 as stated in RFC2372 (1998) that *“The function of broadcast addresses in IPv6 is being superseded by multicast addresses”*. IPv6 contains number of advantages over IPv4.IPv6 is the solution to the exhaustion of IPv4 address. Eventually, IPv4 public addresses will be exhausted due to the limitation of the available address space. As a result, IETF developed IPv6 is the solution to this problem due to its larger address space. In addition to this scalability, IPv6 allows the Internet to grow efficiently, the detail of which have been mentioned in this document. Prior to the introduction of IPv6, Network Address Translation (NAT)and Classless Interdomain Routing (CIDR) were introduced as a temporary solution to the shortage of IPv4 address that allows all the hosts within an intranet site to be able to use the same private IP address range as other intranet sites around the globe.

Currently, address space use in private network and address space use in public network are disjointed which means that the communication between private network and public network is not possible without the use of Network Address Translation (NAT). After migrating to IPv6, disjointed address space will be resolved. At current stage home and enterprise networks are using Internet gateway devices such as ADSL modem or router to connect to the Internet. Each of these devices require having a public IPv4 address that is dynamically or statically allocated by the ISP while private IPv4 addresses are assigned to host devices. When IPv6 is fully established, both home and enterprise networks will be using global IPv6 address, which currently known in IPv4as public IP address. IPv6 has more efficient forwarding mechanism than IPv4 due to the 40 bytes fixed header size that allows routers to make faster decisions in forwarding IPv6 packets(Davies, 2008a).

There are number of advantages that IPv6 has over IPv4. Next section will discuss in detail the differences between these two protocols.

2.3 Comparison of IPv4 and IPv6 features

IPv6 packet header has fewer fields when compare to IPv4 header. IPv4 contains fourteen header fields while IPv6 has eight header fields. However, the size of IPv6 header is double the size of IPv4 header, which means the difference between these two protocols' headers is 20 bytes. This is due to the length of source and destination IPv6 address in IPv6 header fields (Davies, 2008a). There are changes in IPv6 header as compared to IPv4 header:

- The **Header Length** field in IPv4 header is not present in IPv6.
- **Type of Service** field in IPv4 header changed to **Traffic Class** and **Flow Label** field in IPv6.
- **Source address** and **destination address** of IPv4 contains 32 bit long for each field whereas IPv6 contains 128 bit long for each field.
- **Time to Live** field in IPv4 header changed to **Hop Limit** field in IPv6.
- **Protocol** field in IPv4 header changed to **Next Header** field in IPv6.
- IPv6 header does not contain **Options** and **Padding** fields.

Figure 2-2 below shows the differences between IPv4 header and IPv6 header.

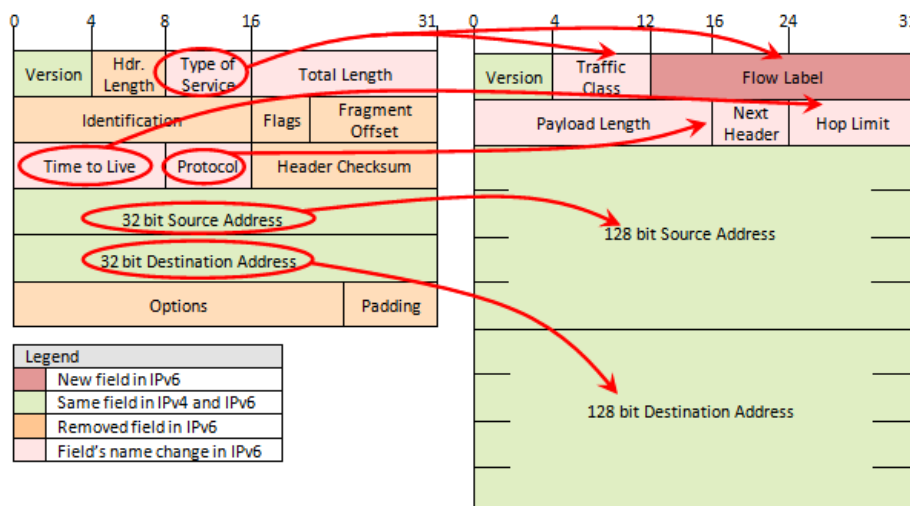


Figure 2-2: IPv4 and IPv6 headers (Sailan, Hassan, and Patel, 2009)

Apart from IP header, IPv6 contain other features that are different from IPv4. Sailan, Hassan, and Patel (2009) stated that the differences between IPv4 and IPv6 features are as follow:

Features	IPv4	IPv6
Address	32 bits	128 bits
Checksum in header	Included	No Checksum
Header includes options	Required	Moved to IPv6 extension headers
Quality of Services	Differentiated Services	Use traffic classes and flow labels
Fragmentation	Done by routers and source node	Only by the source node
IP configuration	Manually or DHCP	Auto-configuration or DHCP
IPsec support	Optional	Required
Transmission type	Unicast, Multicast, and broadcast	Uses unicast, multicast, and anycast (Broadcast dropped)
Address Resolution Protocol (ARP)	Use to resolve an IPv4 address	Replaced with Multicast listener Discovery (MLD)
Domain name service (DNS)	Use host address (A) resource records	Use host address (AAAA) resource records
Mobility	Use Mobile IPv4 (MIPv4)	MIPv6 with faster handover, routing and hierarchical mobility

Table 2-3: Comparison of IPv4 and IPv6 features

According to Table 2-3 above, IPv6 introduces enhanced features over IPv4. The address space in IPv6 is larger than IPv4, which helps solving the shortage of IPv4 address. This is the main feature that IETF introduced in IPv6. IPv6 contains built-in support for Quality of Service, which is an ideal solution for voice and video traffics.

IPv6 supports address auto-configuration and DHCPv6 that include stateless and stateful configuration. IPv6 designed to support mobility as it provides faster handover and routing. All these enhanced features, which make IPv6 as a successor over IPv4. However, IPv6 is not backward compatible with IPv4, which creates difficulties while both protocols coexist during transition period. Transition mechanisms are the tools that help resolving coexistence and transition issues. Section below describes different available transition tools in detail.

2.4 Transition Mechanisms

The design of IPv6 shows that this new version of internet protocol was not designed to be backward compatible with IPv4, which mean IPv4 host is only capable of sending IPv4 packets to other IPv4 hosts, and the same applies to IPv6 host, which is only capable of sending IPv6 packets to other IPv6 hosts. Interoperation is a major issue when both protocols coexist on the Internet. To overcome the coexisting and incompatibility issue, Internet Engineering Task Force NGtrans designed and developed IPv4/IPv6 transition mechanisms to enable transition period to progress without any major issue. IPv4/IPv6 transition mechanisms allow IPv4 and IPv6 to coexist on the Internet. The coexistence of these two internet protocols can last for many years. Before conducting performance measurement on transition mechanisms, it is important to understand the theory behind each transition mechanism comprehensively. The following sections describe each transition mechanism in detail.

2.4.1 Dual Stack

Dual Stack is straightforward and simple configuration transition mechanism. Dual Stack requires operating systems to support both IPv4 and IPv6 which means both IPv4 and IPv6 are enabled on a single network interface card. A device with both IPv4 and IPv6 enabled known as IPv6/IPv4 node, which has the ability to send IPv4 or IPv6 packets to IPv4-only or IPv6-only node and receive IPv4 or IPv6 packets from IPv4-only or IPv6-only node. According to RFC2893 (2000) “IPv6/IPv4 nodes can directly interoperate with IPv4 nodes using IPv4 packets, and also directly interoperate with IPv6 nodes using IPv6 packets.” The following diagram shows the architecture of Dual IP Stacks, which was adapted from Mark & Miller (2000a):

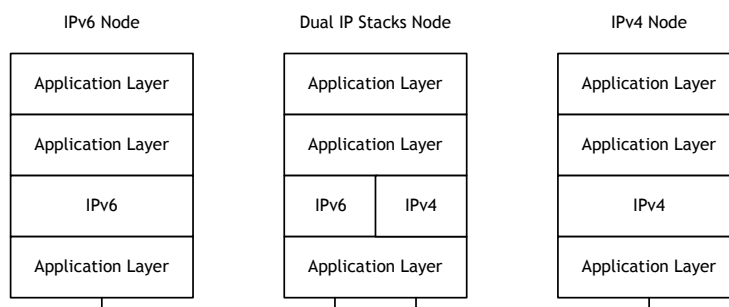


Figure 2-3: Dual IP Stacks

In order to communicate with both IPv4 node and IPv6 node, Dual IP Stacks node requires to have both IPv4 and IPv6 addresses to manually assign or assign by Dynamic Host Configuration Protocol (DHCP). Additionally, Dual IP Stacks node must support the record of DNS for IPv4 and DNS for IPv6 (Mark & Miller, 2000a). Configured Tunnel will discuss next.

2.4.2 Configured Tunnel

Configured Tunnel is a manual configured tunnelling mechanism, which enables two or more IPv6 networks to communicate across IPv4 routing infrastructure through a tunnel. According to Mark & Miller (2000a), “*Configured Tunnelling is defined as IPv6-over-IPv4 tunnelling where the IPv4 tunnel endpoint address is determined by configuration information on the encapsulating node. The tunnel can be either unidirectional or bidirectional. Bidirectional Configured Tunnels behave as virtual point-to-point link.*” The following Figure 2-4 shows the implementation of Configured Tunnel network infrastructure.

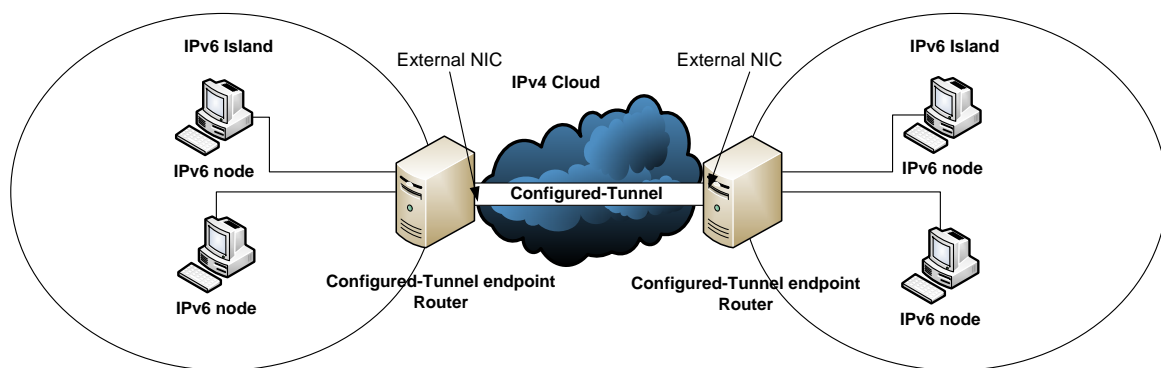


Figure 2-4: Configured Tunnel Network Infrastructure

This tunnelling mechanism requires configuring each tunnel endpoint routers manually in order to deliver IPv6 across IPv4 infrastructure. Each tunnel endpoint router contains two network interface cards with the internal network interface cards configured with IPv6 address and external network interface card configured with IPv4 address. IPv6 address is configured on the tunnel endpoint interface. Next section will discuss 6over4 tunnelling mechanisms.

2.4.3 6over4 Tunnelling Mechanism

6over4 is an automatic tunnelling mechanism that allows interconnection between isolated IPv6 Islands within IPv4 Ocean. 6over4 requires dual stack enabled on the external interface network card with a globally routable IPv4 address in order to connect to the IPv4 internet (Blanchet & Parent, 2000).6to4 tunnelling mechanism will discuss below.

2.4.4 6to4 Tunnelling Mechanism

According to Davies (2008a) “6to4 is an address assignment and router-to-router, host-to-router, and router-to-host automatic tunnel technology that is used to provide unicast IPv6 connectivity between IPv6 sites and hosts across the IPv4 Internet”. Figure 2-5 below show the architecture of 6to4 address adopted from Davies (2008a).

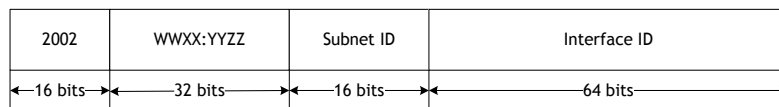


Figure 2-5: 6to4 address architecture

As IPv6 packet arrives at 6to4 router, the encapsulation process is initiated by putting IPv6 packet in IPv4 packet in order to transmit across IPv4 Internet infrastructure. Source and destination IPv4 address is specified with IPv4 header and the body of IPv4 packet contain IPv6 header and payload as stated in RFC3056 (2001).6to4 packet is travelling across 6to4 tunnelling established by 6to4 routers which also known as tunnelling endpoints. As the encapsulated packet arrives at the destination tunnelling end-point, 6to4 router performs de-encapsulation process by removing IPv4 header and forward IPv6 packet through to IPv6 node. Figure 2-6 below shows the implementation of 6to4 network infrastructure:

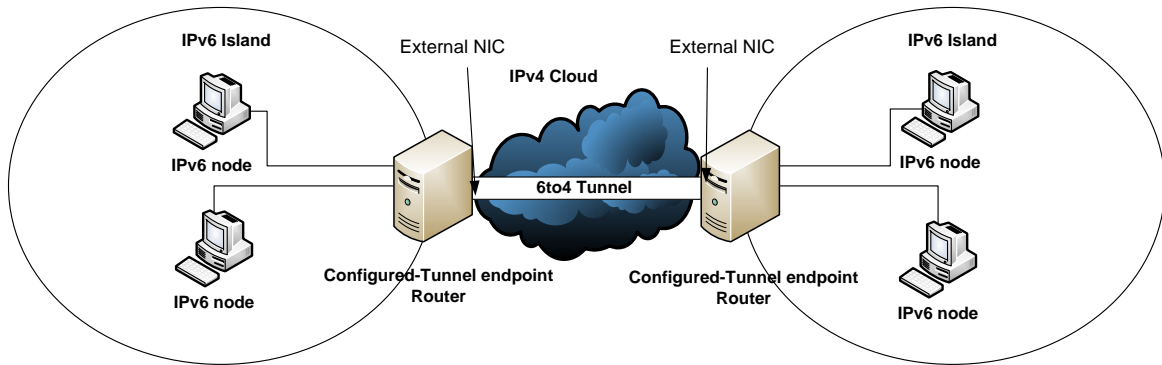


Figure 2-6: 6to4 Network Infrastructure

6to4 has four components that have different functionality. Those four components are 6to4 host, 6to4 router, 6to4 host/router, and 6to4 relay. 6to4 host is a client computer, which does not have ability to perform 6to4 tunnelling across IPv4 Internet. 6to4 router has ability to perform 6to4 tunnelling across the Internet and forwarding 6to4 packet from 6to4 host in a site to another 6to4 host in another site across the Internet. 6to4 host/router has the ability perform tunnelling with 6to4 host/routers, 6to4 routers, and 6to4 relay but it does not have functionality to forward packet. Intra-Site Automatic Tunnel Addressing Protocol will discuss in next section.

2.4.5 Intra-Site Automatic Tunnel Addressing Protocol

Davies (2008a) defines ISATAP as below:

ISATAP is an address assignment and host-to-host, host-to-router, and router-to-host automatic tunnelling technology that provide unicast IPv6 connectivity between IPv6/IPv4 hosts across an IPv4 intranet. ISATAP hosts do not require any manual configuration and they can create ISATAP addresses using standard IPv6 address auto-configuration mechanisms.

ISATAP address will automatically assign to the ISATAP interface. ISATAP does not support router-to-router, which is the reason that this transition mechanism was not selected for this experimental research study. The drawback of ISATAP is the ability to implement across the Internet. ISATAP was not designed for the Internet users, but it was designed for the Intranet users (Hong, Ko, & Ryu, 2006). The discussion of Network Address Translation-Protocol Translation will discuss in next section.

2.4.6 Network Address Translation-Protocol Translation

NAT-PT is a translation mechanism, which functions as a translator for both IPv4 packet and IPv6 packets. The concept of NAT-PT is to translate IPv6 address to IPv4 address and vice versa (RFC 2766, 2000). The implementation of NAT-PT allows IPv4 network and IPv6 network to be able to communicate with one another via just a single NAT-PT server. NAT-PT is one of the ideal solutions, which helps IPv4 and IPv6 to coexist on the internet. Both IPv4 host and IPv6 host do not require having dual stack mode enable. However, each IPv4 and IPv6 network must have its own DNS server. Figure 2-7 below shows the implementation of NAT-PT network infrastructure.

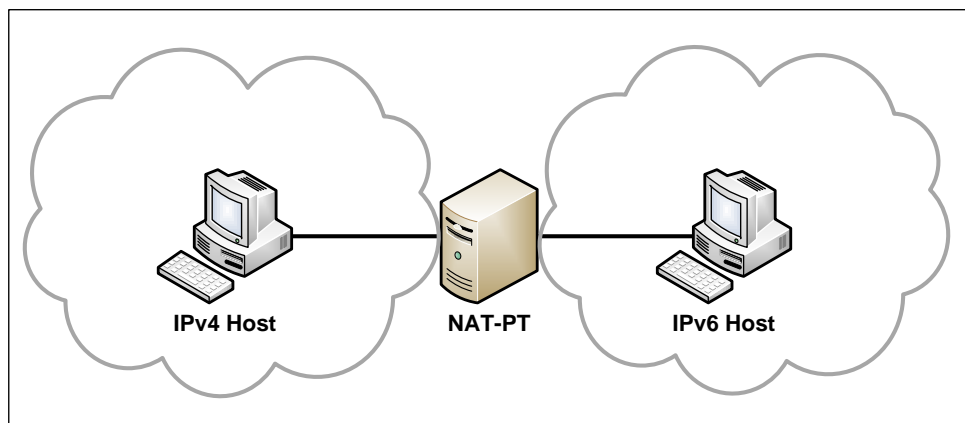


Figure 2-7: NAT-PT Network Infrastructure

The operation of packet translation in NAT-PT is to convert IPv4 packet to IPv6 packet. According to Lee, Shin, and Kim (2004) “*The source address of IPv6 header is replaced by an IPv6 address from IPv4 address pool and then the 96 bits prefix of destination IPv6 address is removed and last 4 bytes is used as IPv4 destination address*”. NAT-PT functions as a translator server that holds global routable IPv4 address pool. The IPv4 address pool is used to dynamically allocate to the IPv6 node before the translation process is initiated across NAT-PT node. As soon as the translation process ends, IPv4 address assignment will end (Atwood, Das, Haddad, 2010). In general, NAT-PT is the IPv4 and IPv6 packet translator that sits in between IPv4 and IPv6 network and it translates IPv4 packet to IPv6 packet and vice-versa. The following section will discuss on different performance metrics used in this research.

2.5 Performance Metrics

This section will be discussing the performance measurement metrics used in the performance testing of transition mechanisms. These metrics are throughput, jitter, delay, and CPU Utilisation.

2.5.1 Throughput

According to previous studies conducted by different researchers, throughput is one of the most common metrics used in the study of network performance evaluation. It helps to understand the amount of data travel across a network connection or between two network hosts. According to Blum (2003), "*The throughput of a network represents the amount of network bandwidth available for a network application at any given moment, across the network links*". There are factors that can affect throughput performance such as the limitation of hardware processing power and network congestion or bottleneck due to the design of network topology. Megabits per second (Mbps) is the unit used in throughput measurement.

2.5.2 Delay

According to Deveriya (2006) states that "*Latency or delay, is the amount of time it takes a packet to traverse from source to destination*". Before measuring delay in an experiment, time synchronisation between sender and receiver is necessary. Make sure that sender and receiver nodes have exactly the same time settings.

2.5.3 Jitter

According to Cisco Systems (2001) "*Jitter is the inter-packet delay variance; that is, the difference between inter-packet arrival and departure. Jitter is an important QoS metric for voice and video applications.*" when multiple packets send across the network, the differences in time that each packet arriving at the destination is known as jitter.

2.5.4 CPU Utilization

CPU utilisation is the percentage amount of computer's CPU resource taken during the processing of an application or a task. Higher percentage of CPU Utilisation shows that the computer optimally used CPU resources. This is not the case, which we should always be concerned about because if a computer shows higher CPU utilisation but it produces higher throughput, means the computer uses best possible resources in order to produce best throughput result. However, if the computer produces lower throughput, which is a point of concern, that hardware is not efficiently using the CPU resources. In this research study, all hardware requires to have identical specification in order to provide consistency between each node.

2.6 Evaluation of Performance Measurement Tools

There are varieties of performance tools available for measuring performance of various networks. Selecting the right tool for the experiment is critical because different tools have different functionalities. This section presents the review and discussion of four main network measurement tools. Below are the four main network measurement tools considered for the network performance measurement in this research study:

- iPerf
- Netperf
- IP Traffic
- D-ITG

2.6.1 iPerf

According to SourceForge (2009) iPerf is a network measurement tool developed by Distributed Applications Support Team (DAST). iPerf is open source software that can be used on both Windows and non-Windows operating systems. As stated by SourceForge (2009), iPerf is capable of evaluating UDP and TCP performance. iPerf is a measurement tool, which is capable of testing bandwidth, delay, jitter, and datagram loss.

iPerf is a command line tool, which means it requires command line interface such as command prompt tool for Microsoft operating systems and terminal tool for Linux. Two

components are required for iPerf to be able to execute. Those components are iPerf client and iPerf server. These two components are included in a single software package but different command is required to execute each component individually. Figure 2-8 below was adapted from openmaniak.com (2010) which shows the two components of iPerf.

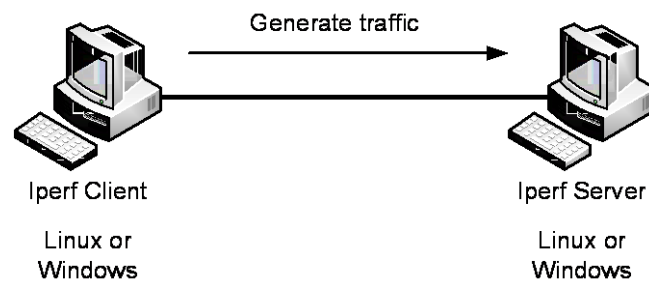


Figure 2-8: IPerf components

iPerf client and iPerf server can either have the same or different operating system installed on the hardware. As per Schroder (2007) statement, “*iPerf can run over the Internet as well as over LAN. It is invaluable for seeing what is happening over a WAN link, whether it is a nice expensive dedicated link or an OpenVPN tunnel over the Internet. The best way is to have iPerf on border route*”. So iPerf is not just a measurement tool that can only use in the experiment test-bed but it can also use to test connection between two sites across WAN connection.

To connect iPerf client to iPerf server use the following command:

```
#iperf -c [iperf server IP address]
```

To connect execute iPerf server use the following command:

```
#iperf -s
```

According to openmaniak.com (2010), TCP and UDP port 5001 is the default port number that iPerf client uses to connect IPerf server.

2.6.2 Netperf

Netperf is another performance measurement tool that has the ability to measure the performance of different type of network. Netperf can be used on various platforms such as Windows, Linux, and Unix (Jones, 2006). According to Jones (2006):

Netperf is a benchmark tool that can be used to measure the performance of many different types of networking. It provides tests for both unidirectional throughput, and end-to-end latency. The environments currently measurable by Netperf include:

- *TCP and UDP via BSD Sockets for both IPv4 and IPv6.*
- *DLPI.*
- *UNIX Domain Sockets.*
- *SCTP for both IPv4 and IPv6.*

Netperf is a command line standard tool that has two elements which is known as Netclient and NetServer. As of June 2009, the latest version of Netperf is 2.4.5. Netperf is open source software that can be downloaded from <http://www.netperf.org/netperf/DownloadNetperf.html>.

2.6.3 IP Traffic

According to ZTI Telecom (2007), IP Traffic is commercial software developed by ZTI-Telecom in France. It is a data generation tool for IP networks. Data flows use TCP (Transmission Control Protocol), UDP (User Datagram Protocol) or ICMP (Internet Control Message Protocol) protocols.

This tool is an IP software-testing tool using the Microsoft Windows TCP/IP stack (Winsock2 interface). So it is independent of any transmission or telecom link and can use any transmission link managed by the Windows operating system: LAN (Ethernet, Token-ring, Hyper LAN, etc.), WLAN, WAN (modem, ISDN, ATM, satellite link, etc.), remote access, mobile or cellular networks.

IP Traffic is a graphic interface benchmark tool; it operates only on Microsoft platform such as Windows 98, Windows XP, Windows 2003, and newer version of Microsoft Windows operating system. Unlike the other three tools above, this performance tool is a commercial software; hence a licence is required to run this software after the 15-day-trial period expires. Like other performance tools, IP Traffic requires two separate parts: Traffic Generator and Traffic Answering. IP Traffic is capable of collecting the following statistics for each connection:

- Sent and received throughput
- Sent and received data volume
- Send and received packets
- Sequence number error
- Min, Max and Mean RTT
- Jitter

2.6.4 Distributed Internet Traffic Generator (D-ITG)

D-ITG is known as Distributed Internet Traffic Generator which was developed by Università degli Studi di Napoli “Federico II”. D-ITG is a command line tool that can run on Windows and non-Windows operating system and supports both IPv4 and IPv6. According to D-ITG (2008)

D-ITG 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, etc.). D-ITG supports both IPv4 and IPv6 traffic generation. It is capable of generating traffic at network, transport, and application layer.

The performance measurement can be done by using two components of D-ITG which include ITG-Send and ITG-Receive. D-ITG is capable of producing traffic at packet level for both IDT (Inter Departure Time) and PS (Packet Size). D-ITG can be used to measure throughput, packet loss, delay, and jitter analysis across heterogeneous network such as wired network, wireless network, GPRS, and Bluetooth (Avallone, Botta, Dainotti, Donato, & Pescap, 2004). Components of D-ITG are ITGSend, ITGRecv, ITGLog, ITGManager, and ITGDec.

2.6.5 Selection of Tool

After evaluating all possible tools, D-ITG has been chosen as the performance measurement tool of this research study. D-ITG tool supports all Windows and Linux environment. Most importantly, D-ITG supports IPv6 protocol and also capable of generating different type of results such as throughput, jitter, delay, and packet drop. D-ITG is a free network

performance measurement tool which has been used in many network performance evaluation studies. D-ITG fulfils all the requirements for this research, which is the reason why this tool was selected.

2.7 Related Studies

There are number of studies related to IPv4 and IPv6 transition mechanisms have been studied in the past. This section covers review of studies relating to the performance evaluation of various transition mechanisms, which will be using as part of secondary resources in data gathering. The following are the five studies:

- Study 1: Performance Comparison of ISATAP Implementations on FreeBSD, RedHat, and Windows 2003.
- Study 2: Evaluating BDMS and DSTM transition mechanisms.
- Study 3: Performance Investigation of IPv4/IPv6 transition mechanisms.
- Study 4: Evaluation IPv4 to IPv6 transition mechanisms.
- Study 5: Performance Evaluation of IPv4/IPv6 deployment over dedicated data links.

2.7.1 Study 1

Performance Comparison of ISATAP Implementations on FreeBSD, RedHat, and Windows 2003

Visoottiviseth and Bureenok have conducted their research on the performance evaluation of Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) comparing to IPv4 and IPv6 protocol based on three different operating systems that include FreeBSD, RedHat 5.0, and Windows 2003 Server. Three experimental test-beds were setup to fulfil the goal of this research. These three setups included native IPv4 network, native IPv6 network, and ISATAP network.

Figure 2-9 below shows the ISATAP experimental design and setup of this research. The network setup involved two computers that function as sender and receiver and these computers were installed with Windows Server 2003 operating system throughout the entire experiment. One computer namely “Client 1” is located in the IPv4 network and configured

as ISATAP client and another computer namely “Client 2” is an IPv6 computer, which is located in the IPv6 network, which also configured as a client. An ISATAP router is used in between the two computers that provide transition between IPv4 network and IPv6 network. The operating system on ISATAP router is varied from Windows Server 2003 to FreeBSD and finally to RedHat Linux accordingly. According to Visoottiviseth & Bureenok (2008), ISATAP router perform three tasks in the transition process“(1) *advertising address prefixes to ISATAP hosts, (2) forwarding packets between ISATAP hosts in IPv4-only network and hosts in hosts in IPv6-only network, and (3) acting as IPv6 default router for ISATAP hosts*”.

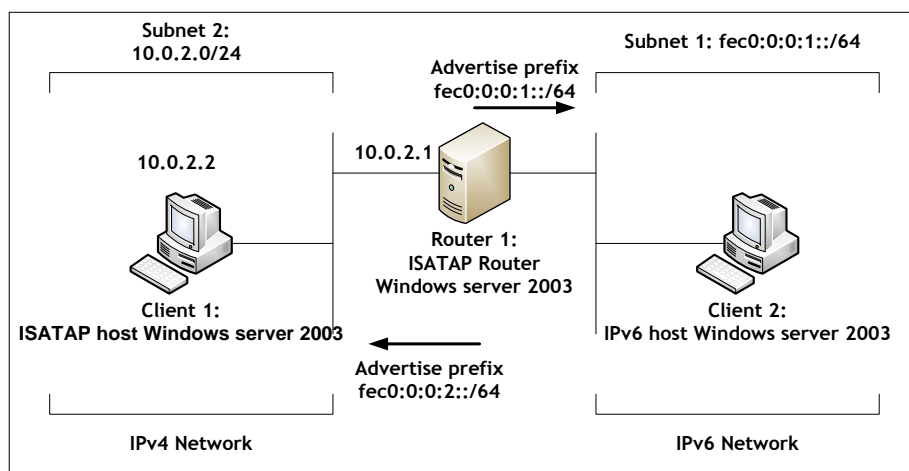


Figure 2-9: ISATAP test-bed configuration (Visoottiviseth & Bureenok, 2008)

Table 2-4 below shows specification of hardware used in experimental research. All three hardware contained different processor and memory specifications.

Router	Processor: Intel Pentium 4 (1.6GHz) Memory: 512 MB
Sender	Processor: Intel Pentium 4 (2.4 GHz) Memory: 128 MB
Receiver	Processor: Intel Pentium 4 (2.4 GHz) Memory: 256 MB

Table 2-4: Hardware specifications

To obtain the result of this research, Visoottiviseth and Bureenok used iPerf measurement tool throughout the experiment. The reason that iPerf had been selected was due to the fact that iPerf compatible with Windows and non-Windows operating systems and support both TCP and UDP traffic. Each test ran 10 times with different payload sizes. The performance metrics used in the experiment are throughput, packet loss rate, and maximum number of packets per second at which the router starts to drop packets. There were 15 different buffer sizes used which included 32, 64, 128, 256, 384, 512, 640, 768, 896, 1024, 1152, 1280, 1408, 1420 and 1536 bytes.

2.7.1.1 TCP Performance Result

The result of TCP throughput shows that IPv4 has the highest throughput due to the size of packet header which is smaller than ISATAP header that encapsulate IPv6 header with IPv4 header before the packet is sent through IPv4 network. Among these three protocols, ISATAP has the highest overhead, which places it at the lowest performance. The comparison of throughput on all three operating systems showed that RedHat 9.0 had the highest throughput and Windows Server 2003 has the lowest throughput when the buffer size reaches 896 bytes. However, with throughput below 896 bytes Windows Server 2003 gained the highest throughput performance.

2.7.1.2 UDP Performance Result

In UDP performance measurement, the payload size was changed from 32 bytes to 64, 128, 256, 512, 1024, and 1432 bytes. The result of UDP throughput shows that native IPv4 has the highest throughput followed by native IPv6 and ISATAP has the lowest throughput. The performance ISATAP on the three operating systems showed that RedHat 9.0 has the highest throughput followed by FreeBSD 5.3 and Windows Server 2003 has the lowest throughput. In term of packet loss results, FreeBSD 5.3 and RedHat 9.0 begin to lose packets when the payload size increases to 20000 pps and Windows Server 2003 began to lose packets at 35000 pps. Overall, the packet loss rate of Windows Server 2003 is larger than RedHat 9.0 and FreeBSD 5.3.

2.7.1.3 Discussion

The group of researchers recommended that RedHat 9.0 is the ideal operating system for the implementation of ISATAP because it shows that it produces the highest performance compared to FreeBSD 5.3 and Windows Server 2003. There is no recommendation for any future study in this paper but further study of the same ISATAP protocol can be conducted on the newer version of Microsoft Windows operating systems and Linux operating systems because the newer operating systems usually provide enhancement in performance and bug fixes as applicable.

2.7.2 Study 2

Evaluating BDMS and DSTM transition mechanisms

AlJa'afreh, Mellor, and Awan (2008) conducted research on Bi Directional Mapping System (BDMS) and Dual Stack Transition Mechanism (DSTM). The purpose of this research was to identify the differences and evaluate the performance of BDMS and DSTM by implementing these mechanisms on OMNet++ simulation software. BDMS is the mechanism that performs conversion whereas DSTM is the mechanism that performs transition between IPv4 and IPv6. AlJa'afreh et al. (2008) presented Table 2-5 below, which is shown the main differences between BDMS and DSTM.

<i>BDMS</i>	<i>DSTM</i>
<i>Uses two types of global addresses that are assigned by the DNS Server for each communication session.</i>	<i>Each communication session is required to have the global IPv4 that assign by a pool of IPv4 addresses.</i>
<i>Two IP addresses (IPv4 and IPv6 addresses) are assumed to be globally unique.</i>	<i>Only IPv4 addresses are assumed to be globally unique.</i>
<i>Tunnelling is not required</i>	<i>Tunnelling is required</i>
<i>Only translation from IPv4 to IPv6 and vice versa is needed.</i>	<i>Encapsulation and de-capsulation IP packet</i>

<i>Applicable for bi-directional communication between IPv6 only nodes and IPv4 only nodes.</i>	<i>Not applicable to the IPv6-only nodes that want to communicate with IP4-only nodes, or vice versa</i>
<i>Does not require upgrading or additional software to be used at the end user nodes.</i>	<i>Requires modifications or extra software to support the dual stack on the end user nodes and IPv6 native network</i>
<i>Less cost</i>	<i>High cost due to upgrading the required DSTM nodes as well as all the edge nodes.</i>

Table 2-5: The differences between BDMS and DSTM

2.7.2.1 Performance Result

The performance metrics that the researchers used in their study are round trip time (RTT), End-to-End Delay (EED), and Throughput. According to RTT result shows that BDMS performs better than DSTM with the packet sizes of less than 400 bytes. The reason is due to high overhead that caused by the encapsulation process in DSTM.

The result of End-to-End Delay shows that BDMS gained better performance than DSTM due to high overhead that caused by the encapsulation process in DSTM. The throughput result of both mechanisms shows that BDMS performs better than DSTM with the packet size of less than or equal to 256 bytes but when the packet size is getting larger, the performance of both mechanisms is very much similar.

2.7.2.2 Discussion

Overall both BDMS and DSTM have similar performance with large packet size but BDMS performs better with small packet size (less than or equal to 256 bytes). This is because DSTM has higher overheads than BDMS due to the encapsulation process in DSTM. According to Alja'afreh et al. (2008), BDMS is better than DSTM in terms of cost, applicability, and modification on end user nodes and other nodes on the network. There is no recommendation for any future study in this paper but further study of BDMS and DSTM can be conducted on the different operating systems and physical hardware that could provide results that are more realistic.

2.7.3 Study 3

Performance Investigation of IPv4/IPv6 transition mechanisms

Chen, Chang, and Lin conducted the performance evaluation of different tunnelling transition mechanisms on Windows Server 2003 operating system for their research. Data gathering focused on four parameters such as latency, throughput, CPU utilization, and packet loss rate for both TCP and UDP transmissions. The test was conducted using three tunnelling mechanisms that included Configured Tunnel, 6to4, and Tunnel broker. Three different test-beds were setup for these three tunnelling mechanisms. Chen et al. (2004) presented Table 2-6 below, which shows the comparisons between three tunnelling mechanisms:

<i>Tunnelling Mechanisms</i>	<i>Advantages</i>	<i>Limitations</i>	<i>Requirements</i>
<i>Configured Tunnel</i>	<i>Stable and secure links for regular communication</i>	<i>Management overhead No independently managed NAT</i>	<i>ISP registered IPv6 address; Dual Stack routers</i>
<i>6to4 tunnel</i>	<i>Connection of multiple remote IPv6 domains</i>	<i>Number of tunnels supported by the 6to4 router</i>	<i>IPv6 prefix (2002::/16); Dual Stack router</i>
<i>Tunnel broker</i>	<i>Standalone isolated IPv6 end system</i>	<i>Potential security implications</i>	<i>It must know how to create and set a script</i>

Table 2-6: Comparisons of tunnel-based mechanisms

Table 2-6 above shows that 6to4 is an ideal transition mechanism for the implementation during the period of migrating from IPv4 to IPv6.

2.7.3.1 Configured TunnelTest-bed

According to Chen et al. (2004), in Configured Tunnel test-bed, Dual Stack gateway is configured as a tunnel endpoint on one end and Dual Stack router is configured as another tunnel end-point at the other end. The tunnel is established across IPv4 network infrastructure and the encapsulation and de-capsulation will be performed at Dual Stack gateway and Dual Stack router. Configured Tunnel requires manual configuration, which accomplished by using command line interface (CLI).Figure 2-10 below shows the design of the network setup for Configured Tunnel test-bed:

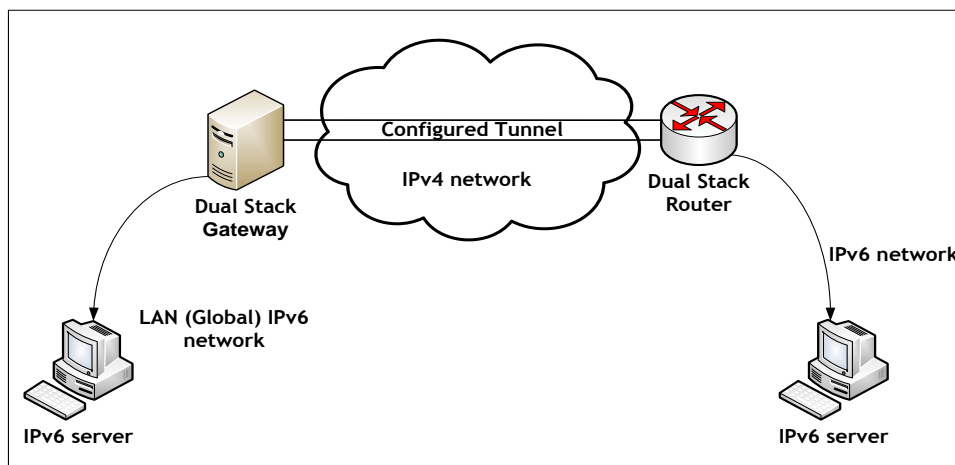


Figure 2-10: Configured Tunneltest-bed(Chen et al., 2004)

According to Figure 2-7 above, the connection between the two tunnel end points is IPv4 network infrastructure or the IPv4 Internet. In between IPv6 server and Dual Stack Gateway or Routers is IPv6 network infrastructure. The IPv6 packets sent from one IPv6 server to another IPv6 server will be encapsulated with IPv4 header and de-capsulated the header by the Dual Stack Gateway or Router

2.7.3.2 6to4 Tunnel Test-bed

In 6to4 tunnel test-bed, 6to4 gateway was configured on one router and the other router is configured as 6to4 relay router. These two routers are functioning as tunnel endpoint that establishes a tunnel across IPv4 network infrastructure say the Internet. The IPv6 packet that is sent from IPv6 node will be encapsulated in IPv4 packet by 6to4 gateway and forwarded to 6to4 relay router through 6to4 tunnel that is established across the IPv4

network. When the packet arrives at 6to4 relay router, the packet will be de-capsulated by 6to4 relay router and forwarded to the global IPv6 network(Chen et al., 2004). Figure 2-11 below shows the network design and setup by Chen et al. (2004) for 6to4 test-bed:

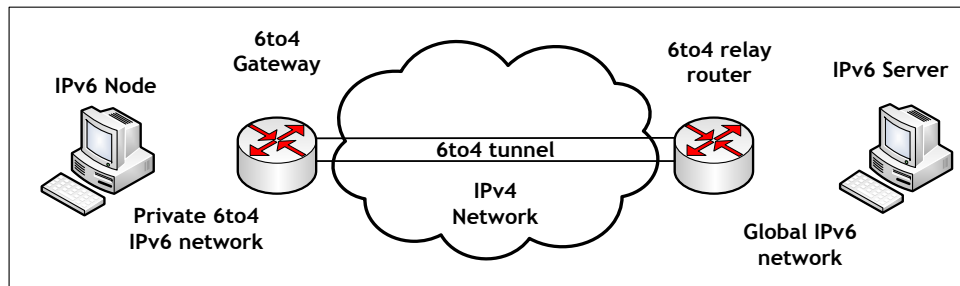


Figure 2-11: 6to4 Tunnel test-bed

2.7.3.3 Tunnel Broker Test-bed

To be able to connect to the tunnel broker Server, tunnel broker client required to have a dual stack configured. In order to establish tunnel between Tunnel Broker client and IPv6 network, Tunnel Broker client must connect to Tunnel Broker Server by using DNS service in Tunnel Broker Server. Figure 2-12 below show Tunnel Broker test-bed.

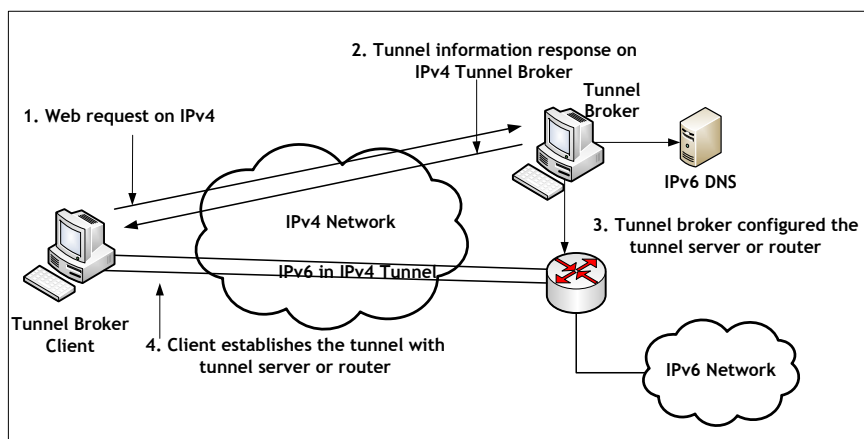


Figure 2-12: Tunnel Broker test-bed(Chen et al., 2004)

2.7.3.4 Latency Analysis

Latency test was conducted by sending six different packet payload sizes that included 64, 128, 256, 512, 768, and 1024 bytes from client to Server and Server back to client in 10000

iteration cycles. The result of latency shows that 6to4 has the lowest latency and Tunnel Broker has the highest latency, which implies that 6to4 performs better than Tunnel Broker transition mechanism.

2.7.3.5 Throughput Analysis

According to Chen et al. (2008) “*The packet size used were ranged from 128 to 1024 bytes. The tests were limited to datagram of 1440 bytes to prevent a potential undocumented fragmentation problem in the IPv6 protocol stack*”. The formula used by Chen et al. (2008) to calculate throughput is $\text{Throughput} = \text{Data packet size} / \text{Latency}$. The summary of Throughput analysis shows that 6to4 transition mechanism achieved better performance than Tunnel Broker transition mechanism.

2.7.3.6 CPU Utilization

The CPU utilization was captured on the edge router that functions as sender. CPU utilization was captured by using the built-in Windows Server 2003 task manager tool. The results show that 6to4 transition mechanism consumed higher CPU processing power due to the process of sent and received packets, which required encapsulation and de-capsulation processes.

2.7.3.7 Loss Rate Analysis

In term of loss rate, the result shows that the loss rates increased when the packet size increases. According to Chen et al. (2008)

When the packet size is 64 bytes, the lost rates of the 6to4 mechanism, configured tunnel, and tunnel broker are 1.0%, 1.5%, and 1.6 % respectively. When the size of the packet is increased to 1024 bytes, these loss rates become 4.2%, 5.8%, and 6.8%”.

This analysis shows that 6to4 mechanism had the lowest loss rate followed by Configured tunnel and tunnel broker respectively.

2.7.3.8 Discussion

There was confusion in term of operating system used in this study. The researchers did not state which operating systems were used for the testing at the beginning of the report except in the CPU utilization section there is a statement saying that “*the CPU utilization was captured by using Windows Server 2003 Task Manger’s performance monitoring tool*” Chen et al. (2008).

As regards the tools, Perfmon can achieve a more accurate result than Task Manager tool because Perfmon produces clear and accurate result of CPU performance. Figure 2.13 below show the screenshot of Perfmon tool, which clearly state the minimum, maximum, and average of the processor percentage.

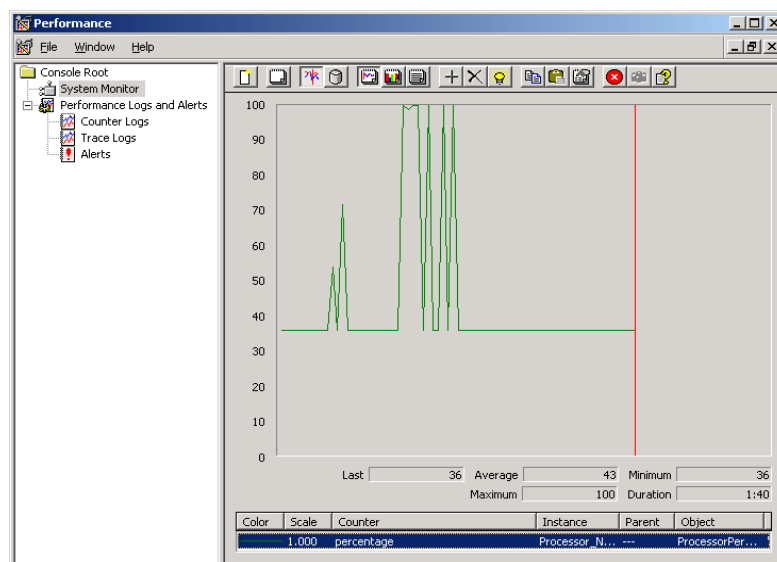


Figure 2-13: Perfmon CPU performance monitoring tool

At the conclusion of this study, recommendation of any further study has not been state by the researchers. However, further study of these transition mechanisms can be conducted on various operating systems using different measurement tools. Thereby, further establishing the reliability of these findings.

2.7.4 Study 4

Evaluation IPv4 to IPv6 transition mechanisms

Raicu & Zeadally conducted a research study on the performance evaluation of 6over4 and IPv6 in IPv4 tunnelling on windows 2000 platform. The parameters used in this study are throughput, latency, CPU utilization, and TCP connection time. Test-beds were implemented using Ericsson AXI462 and IBM 2216 Nways Multiaccess connector Model 400 routers to connect to a workstation on each separate network.

The following was the specification for both workstations:

- Processor: Intel Pentium III 500MHz
- Memory: 256 MB
- HDD: 30GB
- NIC: 10/100

Windows 2000 professional was installed on both workstations and these workstations were configured to support IPv6 protocol stack in order to operate in dual stack mode with IPv4 standard protocol stack. The performance measurement tool was not mentioned but Raicu and Zeadally(2003) state that the software tools were written in C++ programming. The performance measurement was conducted in three different tests. Figure 2-14 below shows three different network setup:

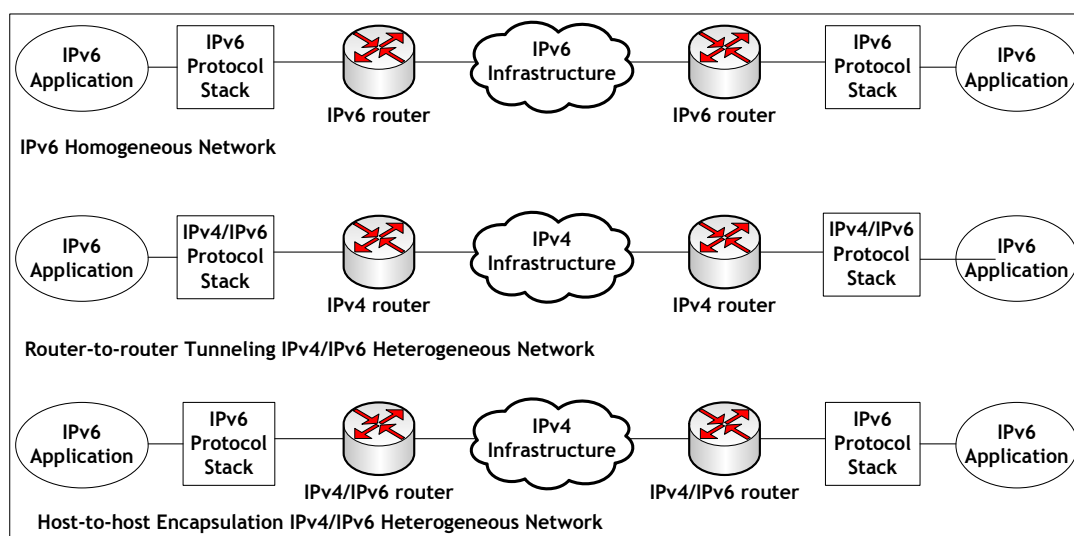


Figure 2-14: Network Setup (Raicu & Zeadally, 2003)

2.7.4.1 **Throughput Result**

The result of throughput comparison between IPv6 network, Router-to-Router tunnelling, and Host-to-Host encapsulation shows that Router-to-Router tunnelling has very small overhead on IPv6 protocol stack and host-to-host encapsulation performed better than IPv6 (Raicu & Zeadally, 2003).

2.7.4.2 **Latency Result**

Latency is the time taken to send a packet from sender to receiver. The packet size used is ranging from 64 bytes to 64 Kbytes. The result shows that with the packet size of 64 Kbytes, Router-to-Router has the highest latency of 42ms, as compare to 40ms on IPv6 and 30ms for host-to-host encapsulation (Raicu & Zeadally, 2003).

2.7.4.3 **TCP Connection Time**

Among the three experiments, host-to-host encapsulation had the fastest connection time and Router-to-Router had the worst connection time. This measurement is important for the applications that are based on TCP connection (Raicu & Zeadally, 2003). Table 2-7 below shows the result of TCP connection time:

<i>IP version</i>	<i>Connection Time (microsecond)</i>
<i>IPv6</i>	2959.13
<i>Host-to-Host encapsulation</i>	2784.42
<i>Router-to-Router</i>	3261.58

Table 2-7: TCP connection time

2.7.4.4 Web Client/Server Simulation

Raicu & Zeadally (2003) presented the following Table 2-8, which shows the number of connections for web client/Server simulation:

<i>IP version</i>	<i>Number of connections</i>
<i>IPv6</i>	79
<i>Host-to-Host encapsulation</i>	80
<i>Router-to-Router</i>	76

Table 2-8: Web client/Server simulation tests

2.7.4.5 Discussion

The researchers will continue their research with different types of transition mechanisms. Windows 2000 used in this research is a bit out of date as many organizations are now mainly deploy windows 2003 and windows XP and some organizations are gradually migrating from windows 2003 and windows XP to windows 2008 and windows vista. However, the result of this research still can be used to compare with newer windows operating systems in order to identify which of these operating systems perform better when using IP version 4 and IP version 6 transition mechanisms.

2.7.5 Study 5

Performance Evaluation of IPv4/IPv6 deployment over dedicated data links

Sanguankotchakorn and Somrobru evaluated the performance of Dual Stack Transition Mechanism (DSTM) on four different types of network traffic that include VoIP IPv4, HTTP IPv4, FTP IPv6 and MPEG-4 IPv6 for both video and audio. The performance evaluation criteria used in this performance evaluation are bandwidth, throughput, the percentage of dropped packets, and end-to-end delay. The experiment of this research conducted using network simulation tool, which known as ns-2. Figure 2-12 below shows the network model used to evaluate the network performance.

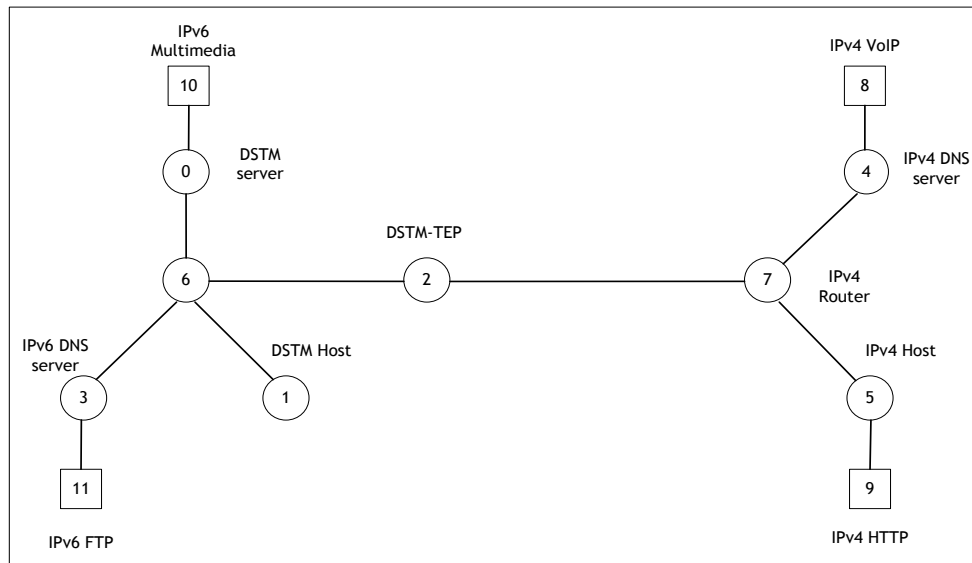


Figure 2-15: Network Model

In this model, four hosts namely host 8, 9, 10, and 11 are being used to generate four different types of traffics. The research team had split the simulation into four cases. Sanguankotchakorn & Somrobru (2005) presented the following Table 2-9, which shows the parameters used in all simulation cases.

Case	VoIP IPv4 (Bytes)	HTTP IPv4 (Bytes)	FTP IPv6 (Bytes)	MPEG-4 (Video)	MPEG-4 (Audio) (Bytes)
1	200 ~ 500, increment step 100	100	1000	Rate Factor = 1	100
2	100	300 ~ 600, increment step 100	1000	Rate Factor = 1	100
3	100	200	1200 ~ 1500, increment step 100	Rate Factor = 1	100
4	100	200	1000	Rate Factor = 3 ~ 6 with step 1	300 ~ 600, increment step 100

Table 2-9: Parameters used in simulation cases

2.7.5.1 Performance Result

The result of Case 1 showed that throughput, bandwidth, and the percentage of dropped packet of all traffic except FTP IPv6 are affected by the large packet size of VoIP IPv4 (400 ~ 500 bytes) because most of the traffic are dropped due to congestion.

The result of Case 2 shows that the packet size of the HTTP IPv4 affects throughput, bandwidth, and the percentage of Dropped Packet. When the packet size of HTTP IPv4 increases, the percentage of dropped packet is also increased for every other traffic except FTP IPv6.

The result of Case 3 shows that the packet size of FTP IPv6 affects throughput, bandwidth, and the percentage of dropped packet of all traffic.

The result of Case 4 shows that the traffic of MPEG-4 IPv6 traffic affects throughput, bandwidth, and the percentage dropped packet of all other traffic (Sanguankotchakorn & Somrobru, 2005).

2.7.5.2 Discussion

The result of this research shows that IPv6 performs better than IPv4. According to Sanguankotchakorn & Somrobru(2005), when IPv6 traffic session increases, the bandwidth of IPv6 session also increases by decrementing the bandwidth of IPv4 session. However, when the traffic of IPv4 session increases, the bandwidth for IPv4 session does not increase.

2.8 Identified Gaps

Table 2-10 below shows the summary of the transition mechanism and operating systems used in previous studies.

Author	Year	Transition Mechanisms Study	Operating Systems
Visoottiviseth, V. and Bureenuk, N	2008	ISATAP	Windows Server 2003, FreeBSD 5.3, and Red Hat 5.0
AlJa'afreh, R., Mellor, J., and Awan, I.	2008	Bi-Directional Mapping System (BDMS) and Dual Stack Transition Mechanism (DSTM)	OMNet++ Simulation Platform
Sanguankotchakorn, T. and Somrobru, M.	2005	Dual Stack Transition Mechanism	Simulation Tool ns-2
Chen, J., Chang, Y., and Lin, C.	2004	Tunnel Broker, Configured Tunnel and 6to4 Tunnel Mechanisms	Windows Server 2003
Raicu, I. and Zeadally, S.	2003	Host-to-host encapsulation and router-to-router tunnelling	Windows Server 2000 Professional

Table 2-10: Summary of related work

Table 2-10 above shows that, operating systems used in the above experimental researches, are Windows Server 2000 and 2003, FreeBSD 5.3, RedHat 5.0, OMNet++ simulation platform, and simulation tool ns2. The first gap identified herein is the version of Microsoft Windows operating systems used in previous study. Currently, there are a number of new releases of Microsoft operating systems and Linux operating systems versions, which could provide improvement over the previous versions. Those improvements could be bug fixes and additional functionalities onto the new versions. In this case, Windows Server 2008, Ubuntu 9.10, and Fedora core 11 can be selected for this current research. Configured Tunnel and 6to4 are the two transition mechanisms selected for this research.

The second gap identified in previous study is the study of Configured Tunnel and 6to4 transition mechanisms on a single operating system (Windows Server 2003). Chen, Chang, & Lin (2004) studied the performance of Configured Tunnel and 6to4 based on Windows

Server 2003 operating system. The difference is that, this study will conduct on multiple operating systems, which are Windows Server 2003 SP2, Windows Server 2008, Ubuntu 9.10, and Fedora 11. The reason that Windows Sever 2003 was chosen for this study is due to the fact that, this study will be conducted on Windows Server 2003 with service pack 2, which is an updated version of the original Windows Server 2003 used in previous study. Next section will presents the literature map of the literature study.

2.9 Literature Map

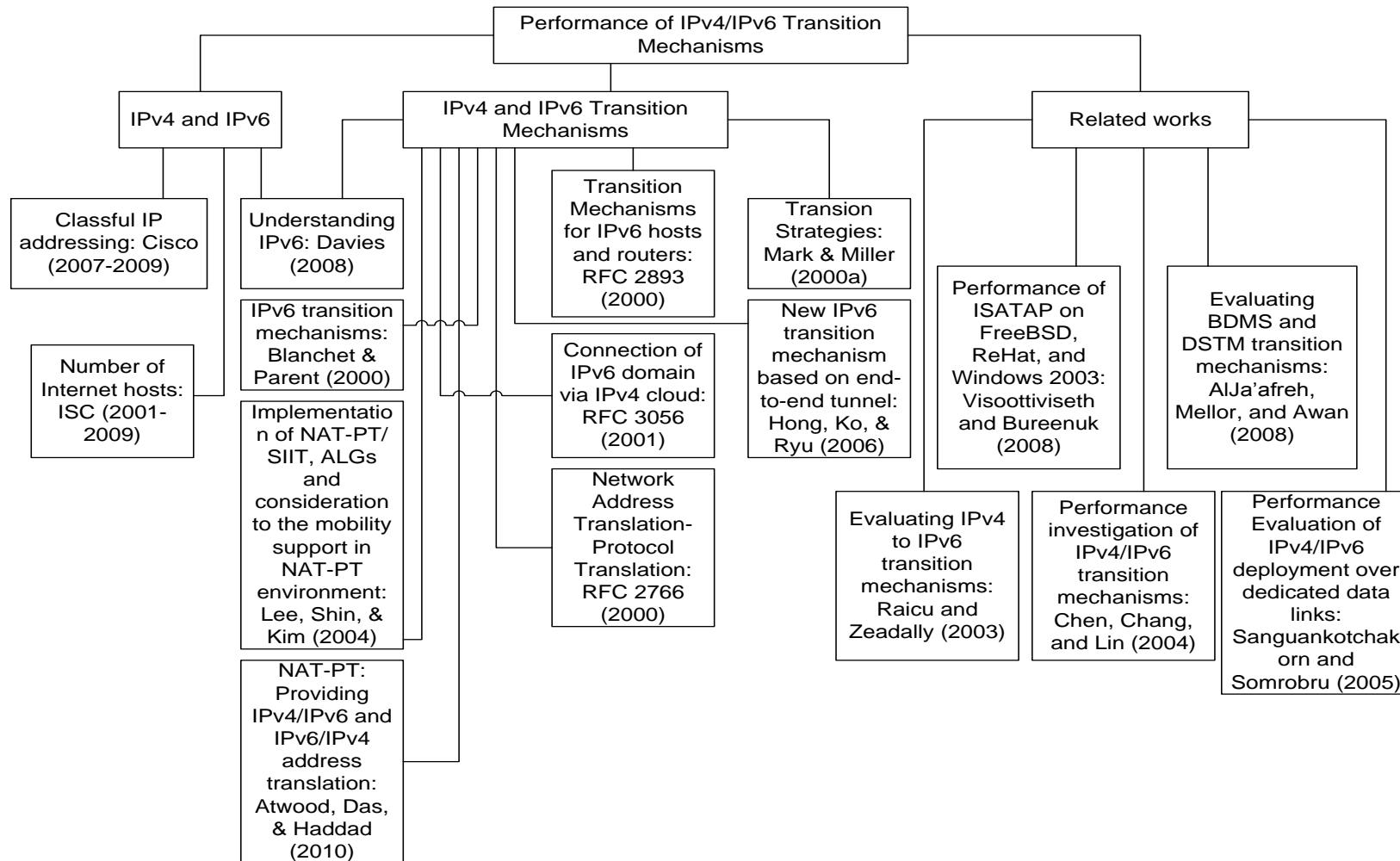


Figure 2-16: Literature map

2.10 Chapter Summary

The above chapter covered the review of Internet Protocol version 4 (IPv4), Internet Protocol version 6 (IPv6), transition mechanisms, performance measurement tools, Performance metrics, related studies, and identified gaps. From the investigation of previous study, there are two gaps found and those gaps are old version of operating systems used and different transition mechanisms used on different operating systems. Next chapter will discuss the research methodology adopted for this study.

3 Chapter 3: Methodology

This chapter will cover the research hypothesis, method of study, and data collection method used in this research.

3.1 Hypothesis

The following are the research hypotheses, which provide a clear scope and understanding of the elements present in this research. Hypothesis is the key driver for this research that forms the experiment and critical data analysis in order to produce a result of the research.

- ❖ There are performance differences between different IP version 4 and IP version 6 transition mechanisms.
- ❖ Different IP version 4 and IP version 6 transition mechanisms perform differently on various Operating Systems.
- ❖ There are factors that cause performance differences between IP version 4 and IP version 6 transition mechanisms.

Hypotheses have been clearly defined in this section and next section will cover the methodology used in this research.

3.2 Method Used for Study

Quantitative method was adopted for this study of network performance measurement. This method mainly concentrates on measurement and statistical data for the objectives that the research focused on. Thomas (2003) states that “*Quantitative methods focus attention on measurements and amounts (more and less, larger and smaller, often and seldom, similar and different) of the characteristics displayed by the people and events that the researchers studies*”. Data gathered in this research is quantitative data, which collected from the experiment conducted in the networking laboratory environment. The findings of this research are the outcomes of the evaluation of data collected from the experiment.

According to Thomas (2003), there are four types of quantitative studies, which include telephone survey, experiment, co-relational study, and quantitative content analysis.

This research will only focus on experimental quantitative research; due to the primary data is totally dependence on the experimental results. The outcome of this research is to find out the performance differences between Configured Tunnel and 6to4 transition mechanisms for the four operating systems that include Windows Server 2003, Windows Server 2008, Linux Ubuntu 9.10, and Linux Fedora Core 11. Next section will be introducing the data collection method.

3.3 Data Collection

Data collection is the process of relevant gathering information by using different type of techniques. According to Creswell (1994), *“data collection steps involve setting the boundaries for the study; collecting information through observations, interviews, documents, and visual materials; and establishing the protocol for recording information”*.

Firstly, the boundaries of this research is to study the performance of Configured Tunnel and 6to4 transition mechanisms on four operating systems that help understanding and adopting a suitable IPv4/IPv6 transition mechanisms for implementation during the migration period from IPv4 to IPv6.

Secondly, the method of collecting information in this research based upon literature review and experimental results. The first stage of data collection process is to search for relevant literature to support the justification of this research and to provide a clear understanding of the previous studies that conducted by different researchers. Searching for literature is mainly based on books, journals, conference proceedings, reports, Request For Comment (RFC) and other credible sources from the Internet. The second data collection process is the experimental design, implementation, and measurement conducted at Unitec project laboratory. In this process, data will obtain by using a suitable performance measurement tool (D-ITG). The experimental process will performs on Microsoft Windows Server and Linux operating systems. Four operating systems selected for the experiment are Windows Server 2003, Windows Server 2008, Linux Ubuntu 9.10, and Linux Fedora Core 11. The result from the experiment help answering the research hypotheses as mentioned in section 3.1 above.

Finally, the results gathered from experiment will enter into Microsoft Excel spreadsheet. Line graphs can be drawn according to the Excel spreadsheet and those graphs can be used as the sources of data analysis, which will produce the outcome of the result. The description of literature gathering process of this research will discuss in the next section.

3.4 Literature Review Process

Literature gathering is the initial stage of this research. It provides the knowledge base and information needed for the study. Related studies provide an idea of how previous researchers conducted their studies and help identifying the direction of future research studies. All literature was gathered from variety of credible and relevant sources such as books, journals, conference proceedings, and online resources. The following are credible sources of literature gathering:

- Books: Unitec Library
- Online Database: IEEE, ACM, Library search engine
- Web search engine: Google, ietf.org

All sources mentioned above are credible and relevant for research study. After literature has been gathered and analysed, the experimental setup and data gathering process will take place in the experimental laboratory. Next section is the discussion of the experimental data gathering process.

3.5 Experimental Data Gathering Process

The primary data source of this research is the experimental result. D-ITG is the selected tool for this performance measurement study. Figure 3-1 below shows the sample log file produced by D-ITG sender and receiver and use D-ITG decoder to decode each log file after the completion of each experiment.

***** TOTAL RESULTS *****		
Number of flows	=	10
Total time	=	10.004000 s
Total packets	=	76437
Minimum delay	=	7.709000 s
Maximum delay	=	7.738000 s
Average delay	=	7.721542 s
Average jitter	=	0.000679 s
Delay standard deviation	=	0.006075 s
Bytes received	=	107623296
Average bitrate	=	86064.211116 Kbit/s
Average packet rate	=	7640.643743 pkt/s
Packets dropped	=	2071 (2.64 %)
Error lines	=	0

Figure 3-1: Sample D-ITG total result

The above figure shows the total result of 10 runs per each packet size. Relevant data used in this research are average delay, average jitter, and average bit rate that circled in red. The results obtained from D-ITG will convert into Microsoft Excel spreadsheets and produce line charts for the comparison and data analysis stage. Figure 3-2 below shows the sample line chart produced by using Microsoft Excel spreadsheet:

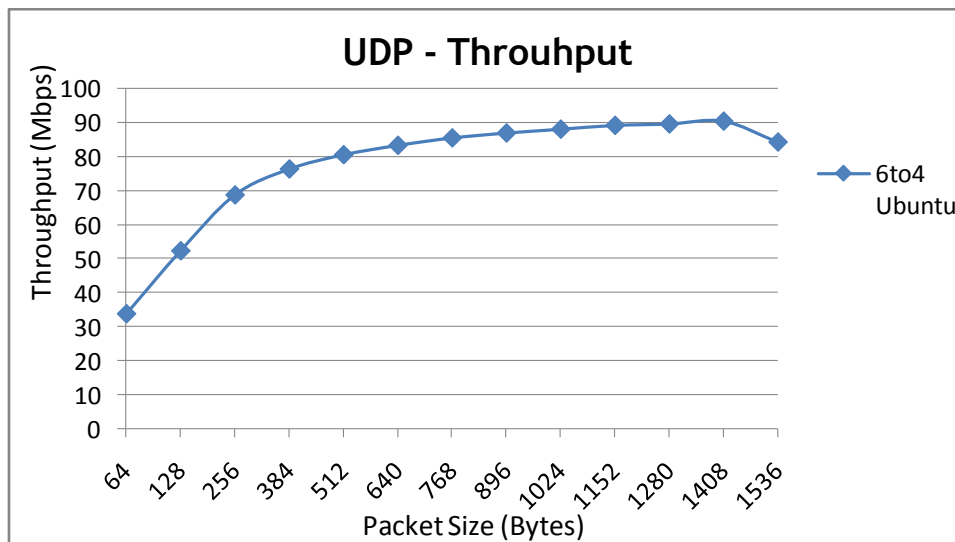


Figure 3-2: Sample Line Chart

From this figure, x-axis show variation in packet sizes measured in bytes and y-axis is the value of throughputs, measured in Megabit per second (Mbps).

3.6 Chapter Summary

This chapter covers the research hypothesis, method of study, and data gathering process of this research. Hypothesis helps to identify the scope of the research. The method of this study is an experimental quantitative approach. Experiment is the primary data source and the result of the experiment will presents in Microsoft Excel spreadsheet. Line charts will produces from Microsoft Excel spreadsheet, which benefits data analysis. Next chapter introduces the design of the experiment.

4 Chapter 4: Experimental Design

This chapter describes the details and procedures of the experiment, which involve the design of network infrastructure and performance measurement. The aim of this experimental research is to focus on the evaluation of different IPv4 and IPv6 transition mechanisms running over various operating systems that include Windows Server 2003, Windows Server 2008, Linux Fedora Core 11, and Linux Ubuntu 9.10.

4.1 Hardware Specifications

To produce a consistent and accurate network performance measurement of 6to4 and Configured Tunnel transition mechanisms over four operating systems, four computers with identical hardware were used in the experiment. Table 4-1 below shows hardware specification for all four computers.

Hardware	Specification
CPU	Intel Core 2 Duo E6300 @ 1.86 GHz
Memory	2GB
Onboard Network Interface Card	Broadcom NetXtreme Gigabit Ethernet
PCI Network Card	Intel 100s Fast Ethernet

Table 4-1: Hardware specification

Due to limitation of hardware resources, each computer was not able to have either two Gigabit NIC card or two Fast Ethernet NIC card. To minimize network bandwidth to 100Mbps, a five ports Fast Ethernet switch used for interconnection between sender computer and the router. Crossover Ethernet cables used for the connection between router and router, and router and receiver.

4.2 Software Specification

Four operating systems selected for the experimental research study in which two are Microsoft operating systems and the other two are Linux open source operating systems:

- Windows Server 2003 Enterprise SP2
- Windows Server 2008 Enterprise SP1
- Ubuntu 9.10
- Fedora Core 11

4.3 Network Design

The experiment involved four computers, which create a network infrastructure that contains two client nodes and two router nodes. This infrastructure is a simulation of two IPv6 LANs interconnected via IPv4 network infrastructure as a simulation of the Internet. Network diagram of the experimental design presents in figure 4-1 below:

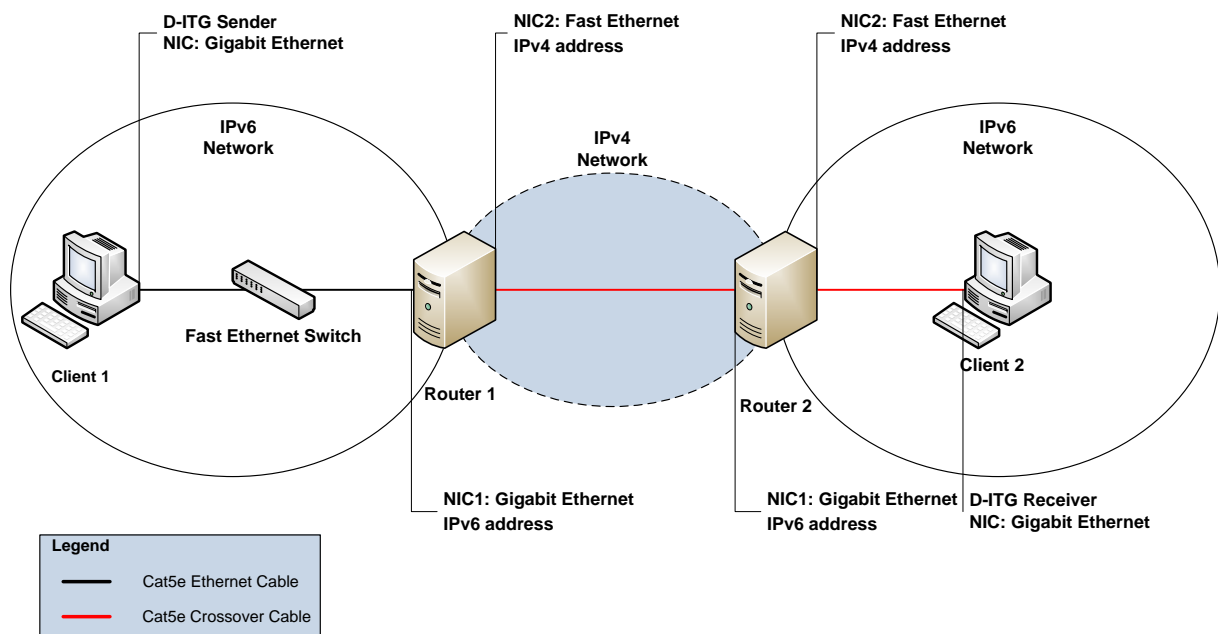


Figure 4-1: Network design

In view of the limitation of hardware resources, there is mixture of Gigabit and Fast Ethernet network cards on the computers used. The overall experimental design involved use of

Internet Protocol version 4 (IPv4), Internet Protocol Version 6 (IPv6) and IPv4/IPv6 transition mechanisms. Both client 1 and client 2 were loaded with Microsoft Windows Vista SP1 for every test-bed. Client 1 configured as the D-ITG packet sender and Client 2 configured as D-ITG packet receiver. To minimise the bandwidth to 100Mbps, a fast Ethernet switch used for interconnection between Client 1 (sender) and Router1 internal NIC card. Crossover cables used to connect Router1 and Router2 and between Router 2 and Client 2. Section below shows the configuration of each test-bed.

4.4 Network Setup and Configuration

There were eight experimental test-bed conducted in this research. Each operating system contains two test-beds. The first experiment is Configured Tunnel test-bed and the second experiment 6to4 test-bed. Windows Vista SP1 installed on a sender host and a receiver host for all eight experiments. However, the operating systems and transition mechanisms on both routers varied according to the experimental designed mentioned above. One computer uses as timeserver, which allows three other computers to synchronise time. Detail of the experimental setup and configuration will present in the following section.

4.4.1 Configured Tunnel mechanism on Windows Server 2003

This section presents the experimental setup and configuration of Configured Tunnel on Windows Server 2003. In this experiment, Windows Vista SP1 was installed on both sender and receiver hosts. In between the two hosts, Windows Server 2003 SP2 was installed on two computers and these two computers were configured as tunnel endpoints for each IPv6 network infrastructure. In addition, these two tunnel endpoints were also configured as software router, to forward packet across from one tunnelling endpoint to another tunnelling endpoint.

Batch script was used to simplify the configuration. In Windows Server 2003, Batch script below used in setting up Configured Tunnel on Router 1:

Batch script for Router 1

- Install IPv6
ipv6 install
- create tunnel name myTunnel
netsh int ipv6 add v6v4tunnel "myTunnel" 10.1.1.1 10.2.1.1
- add IPv6 address to tunnel interface of Router 1
netsh int ipv6 add address "myTunnel" 2001:210:10:3::1
- Configure static routing
netsh int ipv6 add route ::/0 "myTunnel" 2001:210:10:3::2
netsh int ipv6 add route 2001:210:10:1::/64 "Private"
- Set packet forwarding
netsh int ipv6 set int "myTunnel" forwarding=enable
netsh int ipv6 set int "Private" forwarding=enable advertise=enable

Batch script below used in setting up Configured Tunnel on Router 2:

Batch script for Router 2

- Install IPv6
ipv6 install
- create tunnel name myTunnel
netsh int ipv6 add v6v4tunnel "myTunnel" 10.2.1.1 10.1.1.1
- add IPv6 address to tunnel interface of Router 2
netsh int ipv6 add address "myTunnel" 2001:210:10:3::2
- Configure static routing
netsh int ipv6 add route ::/0 "myTunnel" 2001:210:10:3::1
netsh int ipv6 add route 2001:210:10:2::/64 "Private"
- Set packet forwarding
netsh int ipv6 set int "myTunnel" forwarding=enable
netsh int ipv6 set int "Private" forwarding=enable advertise=enable

4.4.2 6to4 mechanism on Windows Server 2003

This section presents the experimental setup and configuration of 6to4 on Windows Server 2003. Batch script for 6to4 configuration was used in order to simplify the configuration. Batch script below used in setting up 6to4 on Router 1:

Batch script for Router 1

- Install IPv6
ipv6 install
- Set packet forwarding
netsh int ipv6 set int "Private" forwarding=enabled
- Enable 6to4
netsh int ipv6 6to4 set state enabled
- Set packet forwarding on 6to4 endpoint interface
netsh int ipv6 set int "6to4 Tunneling Pseudo-Interface" forwarding=enabled
- Add IPv6 address to the Private network interface
netsh int ipv6 add address "Private" 2002:9d3c:101:1::2
- Configure static routing
netsh int ipv6 add route 2002:9d3c:101:1::/64 "Private"
netsh int ipv6 set int "Private" forwarding=enabled advertise=enabled
netsh int ipv6 add route 2002:9d3c:1:472a::/64 "Private"

Batch script below used in setting up 6to4 on Router 2:

Batch script for Router 2

- Install IPv6
ipv6 install
- Set packet forwarding
netsh int ipv6 set int "Private" forwarding=enabled
- Enable 6to4
netsh int ipv6 6to4 set state enabled
- Set packet forwarding on 6to4 endpoint interface

- `# netsh int ipv6 set int "6to4 Tunneling Pseudo-Interface" forwarding=enabled`
- Add IPv6 address to the Private network interface
 - `# netsh int ipv6 add address "Private" 2002:9d3c:1:1::2`
- Configure static routing
 - `# netsh int ipv6 add route 2002:9d3c:1:1::/64 "Private"`
 - `# netsh int ipv6 set int "Private" forwarding=enabled advertise=enabled`
 - `# netsh int ipv6 add route 2002:9d3c:101:472a::/64 "Private"`

4.4.3 Configured Tunnel mechanism on Windows Server 2008

This section presents the experimental setup and configuration of Configured Tunnel on Windows Server 2008. Batch script below used in setting up Configured Tunnel on Router 1:

Batch script for Router 1

- create tunnel name myTunnel
 - `# netsh int ipv6 add v6v4tunnel "myTunnel" 10.1.1.1 10.2.1.1`
- add IPv6 address to tunnel interface of Router 1
 - `# netsh int ipv6 add address "myTunnel" 2001:210:10:3::1`
- Configure static routing
 - `# netsh int ipv6 add route ::/0 "myTunnel" 2001:210:10:3::2`
- netsh int ipv6 add route 2001:210:10:1::/64 "Private" Set packet forwarding
 - `# netsh int ipv6 set int "myTunnel" forwarding=enable`
 - `# netsh int ipv6 set int "Private" forwarding=enable advertise=enable`

Batch script below used in setting up Configured Tunnel on Router 2:

Batch script for Router 2

- create tunnel name myTunnel
 - `# netsh int ipv6 add v6v4tunnel "myTunnel" 10.2.1.1 10.1.1.1`
- add IPv6 address to tunnel interface of Router 2
 - `# netsh int ipv6 add address "myTunnel" 2001:210:10:3::2`
- Configure static routing

```
# netsh int ipv6 add route ::0 "myTunnel" 2001:210:10:3::1
```

```
# netsh int ipv6 add route 2001:210:10:2::/64 "Private"
```

- Set packet forwarding

```
# netsh int ipv6 set int "myTunnel" forwarding=enable
```

```
# netsh int ipv6 set int "Private" forwarding=enable advertise=enable
```

4.4.4 6to4 mechanism on Windows Server 2008

This section presents the experimental setup and configuration of 6to4 on Windows Server 2008. Batch script below used in setting up 6to4 tunnel on Router 1:

Batch script for Router 1

- Set packet forwarding

```
# netsh int ipv6 set int "Private" forwarding=enabled
```

- Enable 6to4

```
# netsh int ipv6 6to4 set state enabled
```

- Set packet forwarding on 6to4 endpoint interface

```
# netsh int ipv6 set int "6to4 Tunneling Pseudo-Interface" forwarding=enabled
```

- Add IPv6 address to the Private network interface

```
# netsh int ipv6 add address "Private" 2002:9d3c:101:1::2
```

- Configure static routing

```
# netsh int ipv6 add route 2002:9d3c:101:1::/64 "Private"
```

```
# netsh int ipv6 set int "Private" forwarding=enabled advertise=enabled
```

```
# netsh int ipv6 add route 2002:9d3c:1:472a::/64 "Private"
```

Batch script below used in setting up 6to4 on Router 2:

Batch script for Router 2

- Set packet forwarding

```
# netsh int ipv6 set int "Private" forwarding=enabled
```

- Enable 6to4

- `# netsh int ipv6 6to4 set state enabled`
- Set packet forwarding on 6to4 endpoint interface
 - `# netsh int ipv6 set int "6to4 Tunneling Pseudo-Interface" forwarding=enabled`
- Add IPv6 address to the Private network interface
 - `# netsh int ipv6 add address "Private" 2002:9d3c:1:1::2`
- Configure static routing
 - `# netsh int ipv6 add route 2002:9d3c:1:1::/64 "Private"`
 - `# netsh int ipv6 set int "Private" forwarding=enabled advertise=enabled`
 - `# netsh int ipv6 add route 2002:9d3c:101:472a::/64 "Private"`

4.4.5 Configured Tunnel mechanism on Ubuntu 9.10

This section presents the experimental setup and configuration of Configured Tunnel on Ubuntu 9.10. The following commands used in setting up Configured Tunnel on Router 1:

Command for Router 1

- Set IPv6 packet forwarding
 - `# sysctl -w net.ipv6.conf.default.forwarding=1`
- Create a sit interface for setting up IPv6-in-IPv4 tunnel
 - `# ip tunnel add sit1 remote 10.2.1.1 local 10.1.1.1`
- Start the interface after configured
 - `# ip link set sit1 up`
- Add IPv6 address to sit1 interface
 - `# ip add address 2001:210:10:3::1/64 dev sit1`
- Configure static routing
 - `# ip -6 route add 2001::/16 dev sit1`
 - `# ip -6 route add ::/0 via 2001:210:10:1::2 dev sit1 metric 1`

The following command used in setting up Configured Tunnel on Router 2:

Command for Router 2

- Set IPv6 packet forwarding
sysctl -w net.ipv6.conf.default.forwarding=1
- Create a sit interface for setting up IPv6-in-IPv4 tunnel
ip tunnel add sit1 remote 10.1.1.1 local 10.2.1.1
- Start the interface after configured
ip link set sit1 up
- Add IPv6 address to sit1 interface
ip add address 2001:210:10:3::1/64 dev sit1
- Configure static routing
ip -6 route add 2002::/16 dev tun6to4
ip -6 route add ::/0 via 2002:0a01:0101:2::2 dev tun6to4 metric 1

4.4.6 6to4 mechanism on Ubuntu 9.10

This section presents the experimental setup and configuration of 6to4 on Ubuntu 9.10.

The following command used in setting up 6to4 tunnel on Router 1:

Command for Router 1

- Set IPv6 packet forwarding
sysctl -w net.ipv6.conf.default.forwarding=1
- Add 6to4 tunnel endpoint
ip tunnel add tun6to4 mode sit remote any local 10.1.1.1
ip link set dev tun6to4 mtu 1472 up
- Add IPv6 address to the tunnel interface
ip -6 addr add dev tun6to4 2002:0a01:0101::1/64
- Configure static routing
ip -6 route add 2002::/16 dev tun6to4
ip -6 route add ::/0 via 2002:0a01:0101:1::2 dev tun6to4 metric 1

The following command used in setting up 6to4 tunnel on Router 2:

Command for Router 2

- Set IPv6 packet forwarding
sysctl -w net.ipv6.conf.default.forwarding=1
- Add 6to4 tunnel endpoint
ip tunnel add tun6to4 mode sit remote any local 10.2.1.1
ip link set dev tun6to4 mtu 1472 up
- Add IPv6 address to the tunnel interface
ip -6 addr add dev tun6to4 2002:0a01:0201::2/64
- Configure static routing
ip -6 route add 2002::/16 dev tun6to4
ip -6 route add ::/0 via 2002:0a01:0201:1::2 dev tun6to4 metric 1

4.4.7 Configured Tunnel mechanism on Fedora Core 11

This section presents the experimental setup and configuration of Configured Tunnel on Fedora 11. The following command used in setting up Configured Tunnel on Router 1:

Command for Router 1

- Set IPv6 packet forwarding
sysctl -w net.ipv6.conf.default.forwarding=1
- Create a sit interface for setting up IPv6-in-IPv4 tunnel
ip tunnel add sit1 remote 10.2.1.1 local 10.1.1.1
- Start the interface after configured
ip link set sit1 up
- Add IPv6 address to sit1 interface
ip add address 2001:210:10:3::1/64 dev sit1
- Configure static routing
ip -6 route add 2001::/16 dev sit1
ip -6 route add ::/0 via 2001:210:10:1::2 dev sit1 metric 1

The following command used in setting up Configured Tunnel on Router 2:

Command for Router 2

- Set IPv6 packet forwarding
sysctl -w net.ipv6.conf.default.forwarding=1
- Create a sit interface for setting up IPv6-in-IPv4 tunnel
ip tunnel add sit1 remote 10.1.1.1 local 10.2.1.1
- Start the interface after configured
ip link set sit1 up
- Add IPv6 address to sit1 interface
ip add address 2001:210:10:3::1/64 dev sit1
- Configure static routing
ip -6 route add 2002::/16 dev tun6to4
ip -6 route add ::/0 via 2002:0a01:0101:2::2 dev tun6to4 metric 1

4.4.8 6to4 mechanism on Fedora Core 11

This section presents the experimental setup and configuration of 6to4 on Fedora 11. The following command used in setting up 6to4 on Router 1:

Command for Router 1

- Set IPv6 packet forwarding
sysctl -w net.ipv6.conf.default.forwarding=1
- Add 6to4 tunnel endpoint
ip tunnel add tun6to4 mode sit remote any local 10.1.1.1
ip link set dev tun6to4 mtu 1472 up
- Add IPv6 address to the tunnel interface
ip -6 addr add dev tun6to4 2002:0a01:0101::1/64
- Configure static routing
ip -6 route add 2002::/16 dev tun6to4
ip -6 route add ::/0 via 2002:0a01:0101:1::2 dev tun6to4 metric 1

The following command used in setting up 6to4 tunnel on Router 2:

Command for Router 2

- Set IPv6 packet forwarding
`# sysctl -w net.ipv6.conf.default.forwarding=1`
- Add 6to4 tunnel endpoint
`# ip tunnel add tun6to4 mode sit remote any local 10.2.1.1`
`# ip link set dev tun6to4 mtu 1472 up`
- Add IPv6 address to the tunnel interface
`# ip -6 addr add dev tun6to4 2002:0a01:0201::2/64`
- Configure static routing
`# ip -6 route add 2002::/16 dev tun6to4`
`# ip -6 route add ::/0 via 2002:0a01:0201:1::2 dev tun6to4 metric 1`

4.5 Chapter Summary

This chapter covered the experimental design of the simulation network environment for this study. The network design included hardware specification, software specification, network diagram, and network configuration. Next chapter presents the analysis of the results from the experiment.

5 Chapter 5: Data Analysis

5.1 TCP Data Analysis

TCP data analysis includes TCP results of Configured Tunnel and 6to4 transition mechanisms over Windows Server 2003, Windows Server 2008, Ubuntu 9.10, and Fedora 11. This section covers the analysis of Configured Tunnel and 6to4 transition mechanisms on all four operating systems under consideration for features of throughput, jitter, delay, and CPU Utilisation.

5.1.1 TCP Throughput

This section presents the line charts and the analysis of TCP results for both Configured Tunnel and 6to4. The values of TCP throughput present in Appendix A.

5.1.1.1 Windows Server 2008 TCP Throughput Results

Figure 5-1 below shows line chart of TCP throughput for Configured Tunnel and 6to4 on Windows Server 2008 operating system with different packet sizes.

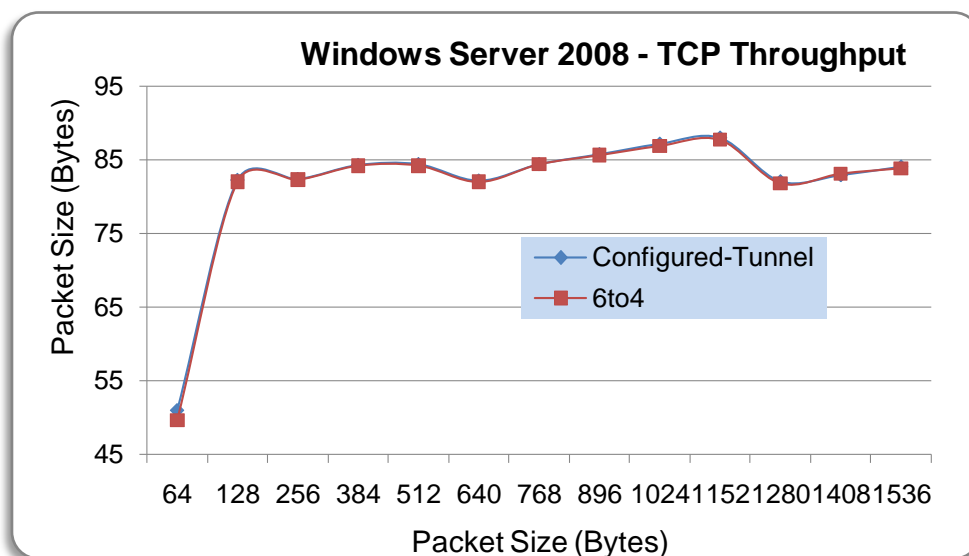


Figure 5-1: Configured Tunnel and 6to4 on Windows Server 2008 – TCP Throughput

- At the smallest packet size of 64 bytes, Configured Tunnel produced higher throughput than 6to4. The difference between these throughputs is 1.5%.
- From packet sizes of 128 bytes to 1280 bytes, Configured Tunnel produced 0.2% higher throughput than 6to4.
- At packet size of 1408 bytes, 6to4 produced 0.2% higher throughput than Configured Tunnel.
- At packet size of 1536 bytes, Configured Tunnel produced 0.2% higher in throughput than 6to4.
- From the packet sizes of 64 bytes to 1152 bytes, there is a significant increase of 38% in throughput values for both Configured Tunnel and 6to4.
- There is a 6% drop in throughput when both tunnelling mechanisms approached the packet size of 1280 bytes. However, as the packet sizes increase to 1536 bytes, there is an increase of 3% in throughput.
- Both Configured Tunnel and 6to4 achieved the highest throughput in between 87 Mbps and 89 Mbps at the packet size of 1152 bytes.
- According to line chart in Figure 5-1 above, Configured Tunnel and 6to4 show the same pattern.
- Overall, there is no significant different in throughput between Configured Tunnel and 6to4 transition mechanisms. However, Configured Tunnel produced slightly higher throughput than 6to4 on most packet sizes.

Next section presents and discusses the results of Configured Tunnel and 6to4 TCP throughput on Windows Server 2003.

5.1.1.2 Windows Server 2003 TCP Throughput Results

Figure 5-2 below shows line chart of TCP throughput for Configured Tunnel and 6to4 on Windows Server 2003 operating system with different packet sizes.

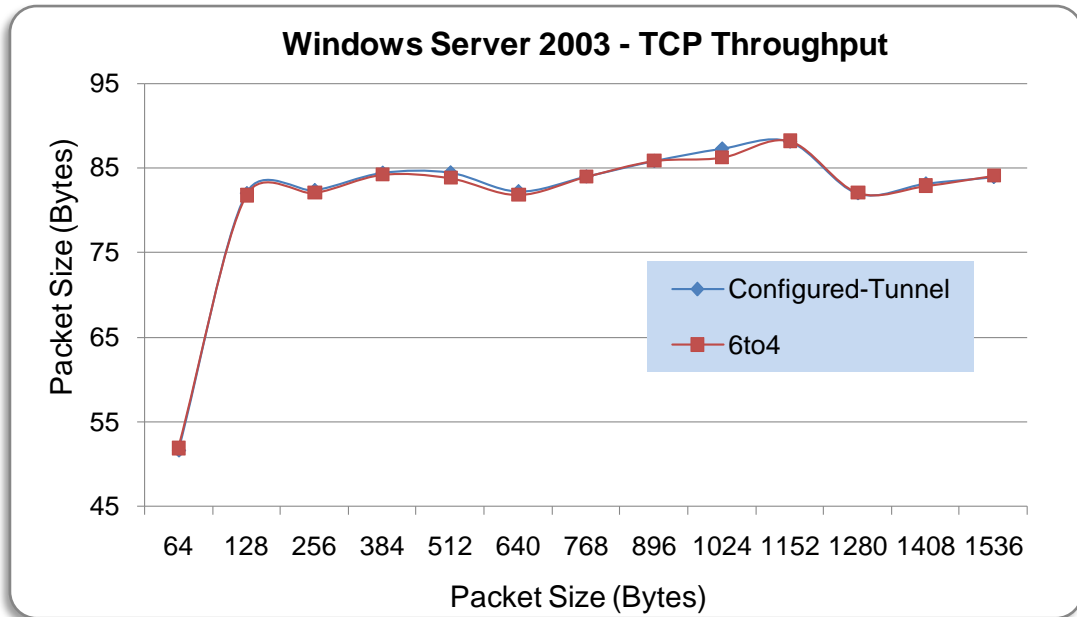


Figure 5-2: Configured Tunnel and 6to4 on Windows Server 2003 – TCP Throughput

- According to line chart in Figure 5-2 above, 6to4 produced 0.3% higher throughput than Configured Tunnel at the packet size of 64 bytes.
- As the packet sizes increase from 128 bytes to 640 bytes, Configured Tunnel produced slightly higher throughput than 6to4 with the differences of 0.2%.
- From packet sizes of 768 bytes to 896 bytes, 6to4 produced 0.5% higher throughput than Configured Tunnel.
- At packet size of 1024 bytes, Configured Tunnel produced 1% higher in throughput than 6to4.
- From packet sizes of 1152 bytes to 1280 bytes, 6to4 produced 0.2% higher in throughput than Configured Tunnel.
- At packet size of 1408 bytes, Configured Tunnel produced 0.23% higher in throughput than 6to4.
- At packet size of 1536 bytes, 6to4 produced 0.2% higher in throughput than Configured Tunnel.
- From the packet sizes of 64 bytes to 1152 bytes, there is a significant increase of 36% in throughput values for both Configured Tunnel and 6to4.
- There is a 6% drop in throughput when both tunnelling mechanisms approached the packet size of 1536 bytes.
- Both Configured Tunnel and 6to4 achieved the highest throughput in between 87 Mbps to 89 Mbps at the packet size of 1152 bytes.

- According to line chart in Figure 5-2 above, Configured Tunnel and 6to4 show the same pattern.
- Overall, there is no significant different in throughput between Configured Tunnel and 6to4. However, Configured Tunnel produced slightly higher throughput than 6to4 on most packet sizes.

Next section presents and discusses the results of Configured Tunnel and 6to4 TCP throughput on Ubuntu 9.10.

5.1.1.3 *Ubuntu 9.10 TCP Throughput Results*

Figure 5-3 below shows line chart of TCP throughput for Configured Tunnel and 6to4 on Ubuntu 9.10 operating system with different packet sizes.

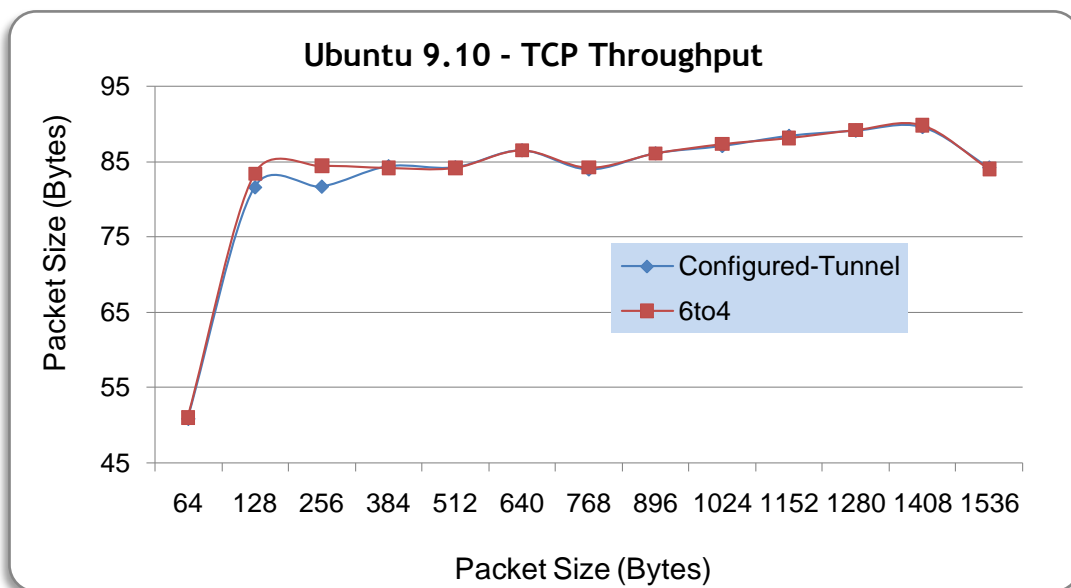


Figure 5-3: Configured Tunnel and 6to4 on Ubuntu 9.10 – TCP Throughput

- On packet sizes 128 bytes and 256 bytes, 6to4 produced 2% higher in throughput than Configured Tunnel.
- At packet size of 512 bytes, Configured Tunnel produced 0.1% higher in throughput than 6to4.

- From packet sizes of 640 bytes to 1152 bytes, 6to4 produced 0.2% higher in throughput than Configured Tunnel.
- From packet sizes of 1280 bytes to 1408 bytes, Configured Tunnel produced 0.3% higher in throughput than 6to4.
- At packet size of 1536 bytes, 6to4 produced 0.2% higher throughput than Configured Tunnel.
- From the packet sizes of 64 bytes to 1408 bytes, there is a significant increase of 40% throughput values for both Configured Tunnel and 6to4.
- There is a 6% drop in throughput when both tunnelling mechanisms approached the packet size of 1536 bytes.
- Both Configured Tunnel and 6to4 achieved the highest throughput in between 88 Mbps to 90 Mbps at the packet size of 1408 bytes.
- According to line chart in Figure 5-3 above, Configured Tunnel and 6to4 show the same pattern.
- Overall, there is no significant different in TCP throughput between Configured Tunnel and 6to4. However, 6to4 produced slightly higher throughput than Configured Tunnel on most packet sizes.

Next section presents the results of Configured Tunnel and 6to4 TCP throughput on Fedora 11.

5.1.1.4 *Fedora 11 TCP Throughput Results*

Figure 5-4 below shows line chart of TCP throughput for Configured Tunnel and 6to4 on Fedora 11 operating system with different packet sizes.

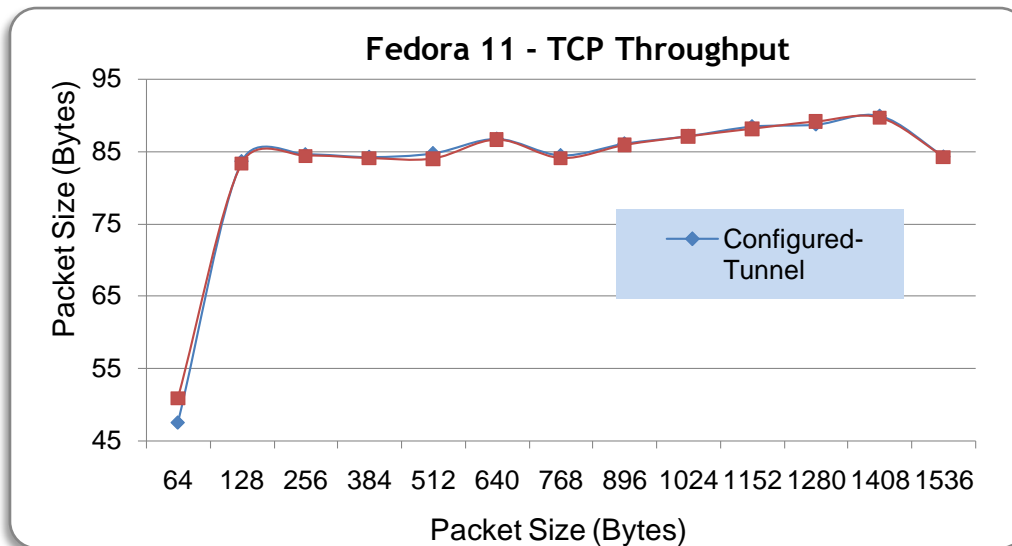


Figure 5-4: Configured Tunnel and 6to4 on Fedora 11 – TCP Throughput

- As shown in Figure 5-4 above, 6to4 produced 3% higher throughput value than Configured Tunnel.
- From packet sizes of 128 bytes to 1152 bytes, Configured Tunnel produced 0.2% higher throughput than 6to4.
- At packet size of 1280 bytes, 6to4 produced 0.02% higher throughput than Configured Tunnel.
- At packet size of 1408 bytes, Configured Tunnel produced 0.2 higher throughput than 6to4.
- At packet size of 1536 bytes, 6to4 produced 0.02% higher throughput than Configured Tunnel.
- From the packet sizes of 64 bytes to 1408 bytes, there is a significant increase of 40% in throughput values for both Configured Tunnel and 6to4.
- There is a 6% drop in throughput when both tunnelling mechanisms approached the packet size of 1536 bytes.
- Both Configured Tunnel and 6to4 achieved the highest throughput in between 88 Mbps to 90 Mbps at the packet size of 1408 bytes.
- According to line chart in Figure 5-3 above, Configured Tunnel and 6to4 show the same pattern.
- Overall, there is no significant different in TCP throughput between Configured Tunnel and 6to4. However, Configured Tunnel produced slightly higher throughput than 6to4 on most packet sizes.

Next section presents the comparison results of Configured Tunnel and 6to4 on all four operating systems.

5.1.1.5 Compare Four Operating systems TCP Throughput Result

Figure 5-5 below shows TCP throughput between Configured Tunnel and Fedora on all four operating systems with different packet sizes.

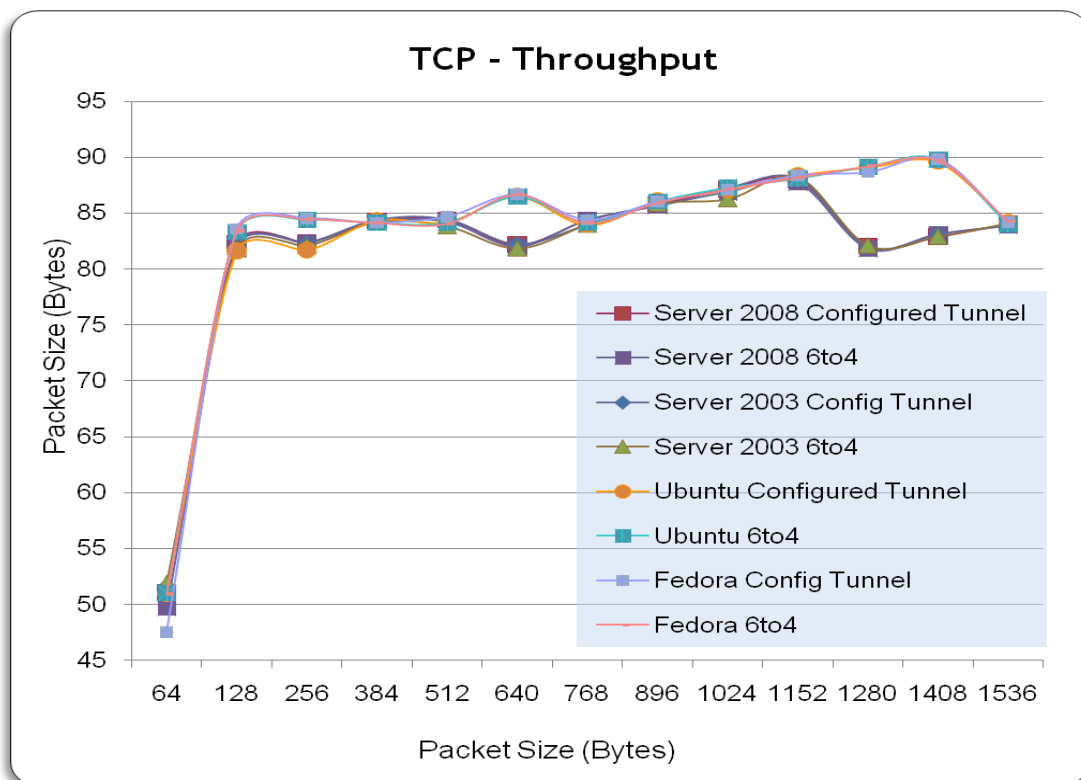


Figure 5-5: TCP Throughput (Mbps)

- From the smallest packet size of 64 bytes to a large packet size of 1152 bytes, there is a significant increase of 38% in throughput values for both Configured Tunnel and 6to4.

Figure 5-6 below shows the line chart from packet size of 64 bytes to 128 bytes.

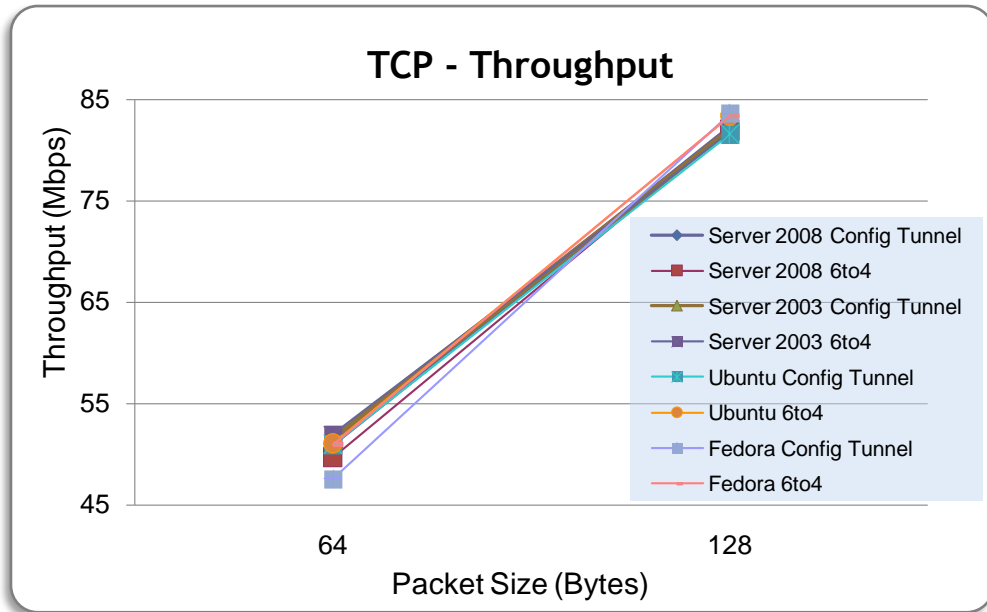


Figure 5-6: TCP Throughput from packet size 64 bytes to 128 bytes

- For the smallest packet size of 64 bytes, Configured Tunnel on Fedora produced the lowest throughput. On the other hand, 6to4 on Windows Server 2003 produced the highest throughput with an average difference of 4.5%.
- At packet size of 128 bytes, Configured Tunnel on Ubuntu produced the lowest throughput and Configured Tunnel on Fedora produced the highest throughput with the difference of 2%.

Figure 5-7 below shows the line chart from packet size of 128 bytes to 768 bytes.

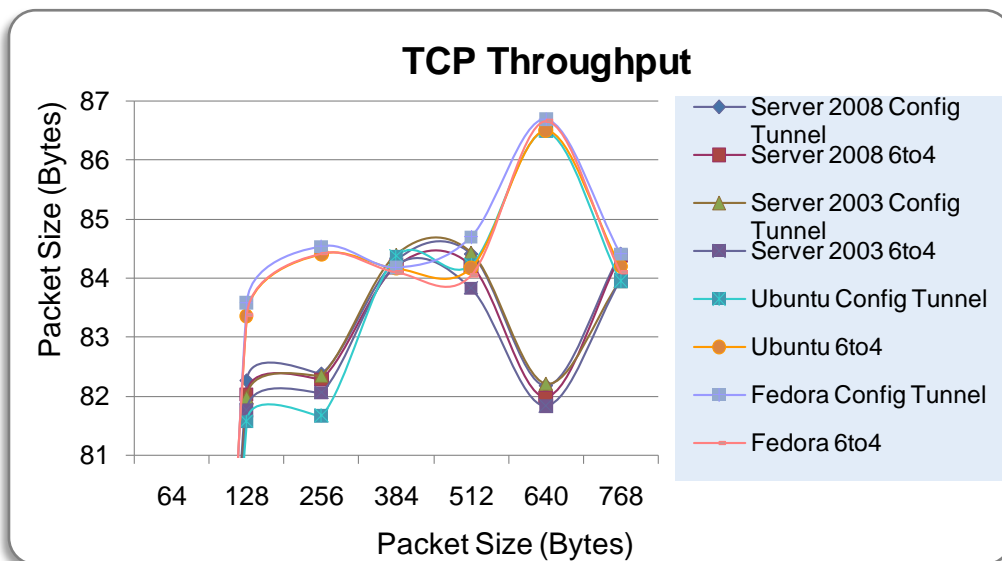


Figure 5-7: TCP Throughput from packet size 128 bytes to 768 bytes

- From packet sizes of 128 bytes to 256 bytes, there is a 1% increase in throughput for 6to4 on both Linux operating systems and Configured Tunnel on Fedora. For every other performance, TCP throughput is stabilised. Within this packet sizes range, 6to4 on both Linux and Configured Tunnel on Fedora have similar throughput and higher than any other TCP throughput. On the other hand, Configured Tunnel on Ubuntu produced the lowest throughput.
- As packet sizes increase from 254 bytes to 384 bytes, 6to4 on both Linux and Configured Tunnel on Fedora show a slight drop of 0.3% in throughput values. On the other hand, every other TCP performance shows a significant increase of 2% in throughput values. According to Figure 5-7, both tunnelling mechanisms on all four operating systems produced similar throughput at the packet size of 384 bytes.
- At packet size of 512 bytes, Configured Tunnel on Fedora began to increase in TCP throughput value and 6to4 on Windows Server 2003 began to drop in TCP throughput value. At this instant, Configured Tunnel on Fedora produced the highest throughput and 6to4 on Windows Server 2003 produced the lowest throughput with the difference of less than 1%. TCP throughput everywhere else stabilise at packet size of 512 bytes.
- As the packet sizes increase from 512 bytes to 768 bytes, a pattern shows that there is an increase of 2.5% in throughput values for both tunnelling mechanisms of both Linux operating systems at the packet size of 640 bytes and a drop of 2.5% in throughput at the packet size 768 bytes. On the other hand, there is a dropped of 2.5% in throughput values for both tunnelling mechanisms on both Windows operating systems at the packet size of 640 bytes and an increase of 2.5% at the packet size of 768 bytes. At the packet size of 768 bytes, both tunnelling mechanisms on all four operating systems produced similar TCP throughput.
- At the packet size of 640 bytes, both tunnelling mechanisms on both Linux operating systems outperform both Windows operating systems by 4.5% in throughput values.

Figure 5-8 below shows the line chart from packet size of 768 bytes to 1536 bytes.

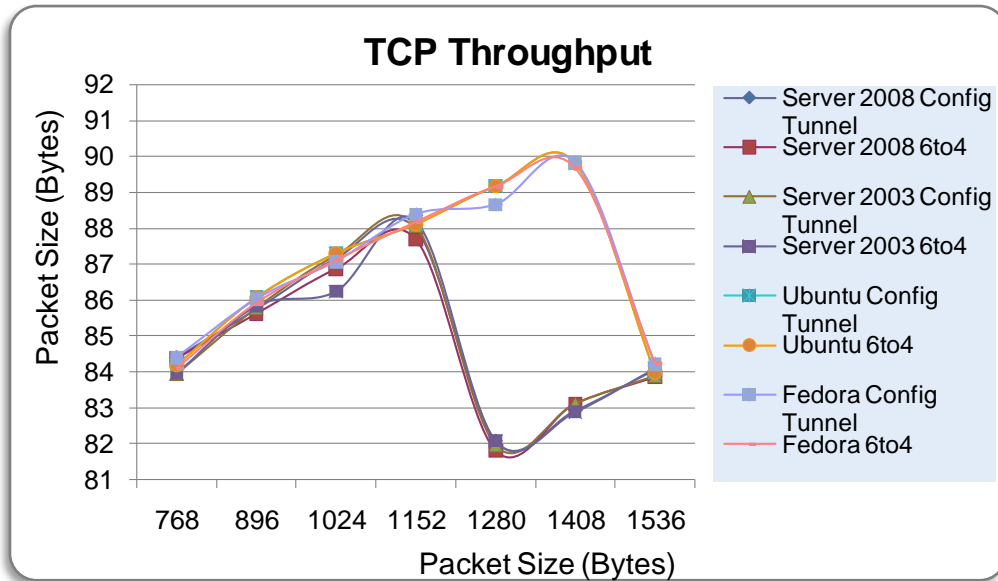


Figure 5-8: TCP Throughput from packet size 768 bytes to 1536 bytes

- As the packet sizes increase from 768 bytes to 1152 bytes, there is a significant increase of 4% in throughput values for both tunnelling mechanisms on all four operating systems. In between these packet sizes, both tunnelling mechanisms have similar performance on all four operating systems except at packet size of 1024 bytes, 6to4 on Windows Server 2003 has slightly lower throughput than the rest of TCP throughput with the difference of less than 1%.
- From packet sizes of 1152 bytes to 1280 bytes, both tunnelling mechanisms on both Windows operating systems show a significant decline in TCP throughput with a dropped average of 6%. On the other hand, both transition mechanisms on both Linux operating systems show a slight incline in TCP throughput with an average of less than 1%. Both transition mechanisms on Linux operating systems outperform both Windows operating systems with a significant different of 7%.
- However, at packet size of 1152 bytes, Configured Tunnel on Fedora produced slightly higher throughput than the rest of TCP throughput and 6to4 on Windows Server 2008 produced slightly lower TCP throughput than the rest of TCP throughput.
- From packet sizes of 1280 bytes to 1408 bytes, there is an incline of 1% in throughput values for both transition mechanisms on all four operating systems. However, both Linux operating systems still outperform both Windows operating systems with a difference of 7%.

- From packet size of 1408 bytes to 1536 bytes, both tunnelling mechanisms on both Linux operating systems show a significant decline in TCP throughput with an average of slightly less than 6%. On the other hand, both tunnelling mechanisms on both Windows operating systems show an incline in TCP throughput with an average of 1%.
- At packet size 1536 bytes, both tunnelling mechanisms produced similar throughput on all four operating systems.

Overall, both transition mechanisms on both Linux operating systems outperform both transition mechanisms of both Windows operating systems for most packet sizes, except at packet size of 64 bytes and 384 bytes. However, for most packet sizes, Configured Tunnel produced slightly higher throughput than 6to4 on Fedora and both tunnelling mechanisms on Ubuntu.

Next section presents the results of Configured Tunnel and 6to4 TCP jitter on all four operating systems.

5.1.2 TCP Jitter

Figure 5-9 below shows TCP jitter of Configured Tunnel and 6to4 transition mechanisms on four operating systems with packet sizes range from 64 bytes to 1536 bytes. The values of TCP jitter present in Appendix B.

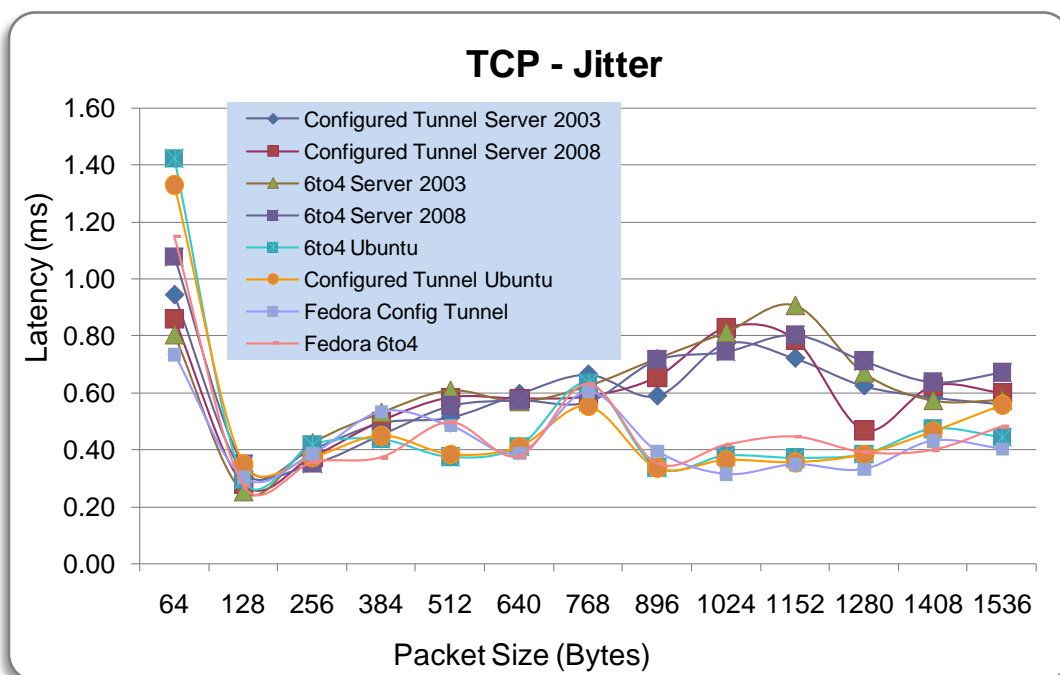


Figure 5-9: TCP Jitter

- At the smallest packet size, 6to4 on Ubuntu produced the highest jitter and 6to4 on Windows Server 2003 produced the lowest jitter with a significant difference of 30%.
- As the packet sizes increase from 64 bytes to 256 bytes, both tunnelling mechanisms produced similar jitter.
- From packet sizes of 64 bytes to 128 bytes, there is a significant decrease in jitter for both tunnelling mechanisms on all four operating systems, with an average of less than 40%.
- From packet sizes of 64 bytes to 768 bytes, the graph shows similar pattern for both tunnelling mechanisms on all four operating systems.
- As the packet sizes increase from 768 bytes to 1536 bytes, the graph clearly shows an interesting pattern. The pattern shows that both tunnelling mechanisms on both Linux operating systems produced lower jitter than both tunnelling mechanisms on both Windows operating systems. However, Configured Tunnel on Fedora produced the lowest jitter on large packet sizes.

Over all, on small packet sizes Configured Tunnel and 6to4 show the same pattern with similar jitter values. For large packet sizes, the graph clearly shows that both Configured Tunnel and 6to4 on both Linux operating systems produced lower jitter than Configured Tunnel and 6to4 on Windows operating systems. Next section will presents and discusses the results of Configured Tunnel and 6to4 TCP delay on all four operating systems.

5.1.3 TCP Delay

Figure 5-10 below shows line chart of TCP delay with different packet size range from 64 bytes to 1536 bytes. The values of TCP delay presented in Appendix C.

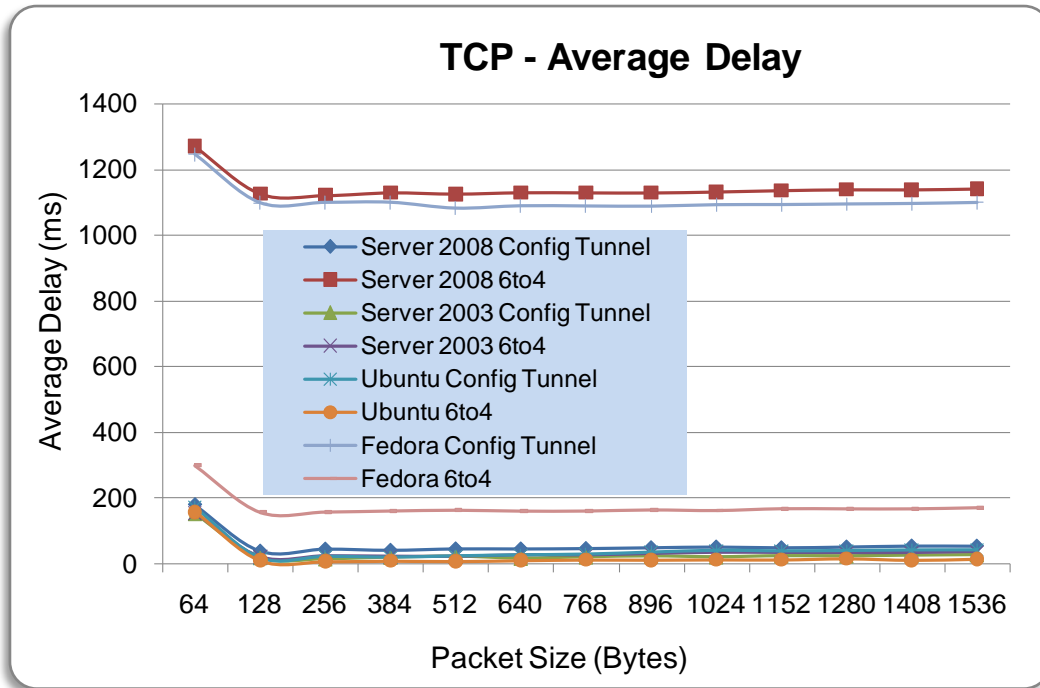


Figure 5-10: TCP Delay

- From packet sizes of 64 bytes to 128 bytes, there is a slight decrease in delay with an average of 9% for both transition mechanisms over all four operating systems.
- From packet sizes of 64 bytes to 1536 bytes, Configured Tunnel on Fedora produced similar average delay to 6to4 on Windows Server 2008 with the differences of less than 0.2%.
- 6to4 on Windows Server 2008 and Configured Tunnel on Fedora have highest delay, with an average of 1000ms higher than every other delay performance.
- As the packet sizes increase from 128 bytes to 1536 bytes, 6to4 on Ubuntu has the lowest delay and 6to4 on Windows server 2008 has the highest delay.

Overall, 6to4 on Windows Server 2008 and Configured Tunnel on Fedora show the highest significant delay. Next section will presents and discusses the results of Router 1 CPU usage of all four operating systems.

5.1.4 TCP - Router1 CPU Utilization

Figure 5-11 below shows line chart of Router1 CPU utilisation with different packet size range from 64 bytes to 1536 bytes. The values of TCP Router 1 CPU Utilisation presented in Appendix D.

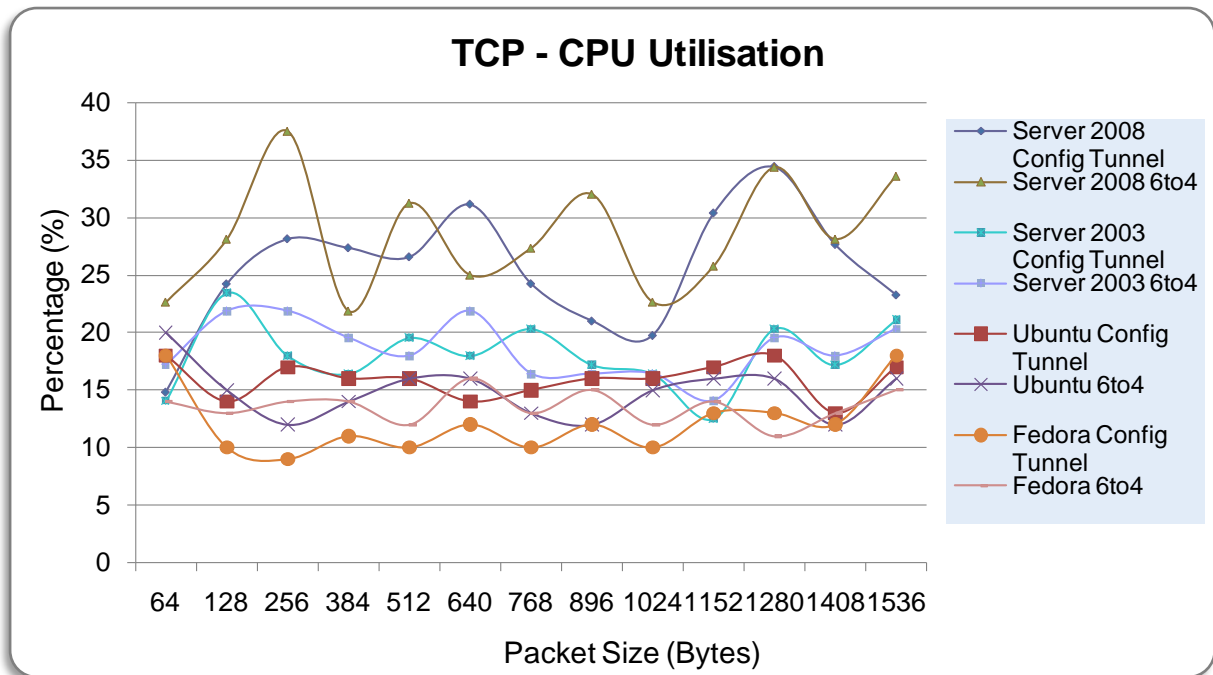


Figure 5-11: Router 1 - CPU Utilisation of TCP performance

- At the smallest packet size of 64 bytes, 6to4 on Fedora and Configured Tunnel on Windows Server 2003 shows the lowest percentage of CPU usage.
- Between packet sizes of 64 bytes to 384 bytes, 6to4 on Windows Server 2008 shows the highest percentage of CPU usage. On the other hand, Configured Tunnel on Fedora shows the lowest percentage of CPU usage.
- From packet sizes of 128 bytes to 1536 bytes, both 6to4 and Configured Tunnel on Windows Server 2008 shows the highest percentage of CPU usage.
- As packet sizes increase from 128 bytes to 1024 bytes, Configured Tunnel on Fedora shows the lowest percentage of CPU usage. However, at packet size of 896 bytes, 6to4 on Ubuntu and Configured Tunnel on Fedora shows the same lowest percentage of CPU usage.

Overall, Windows Server 2008 used the highest CPU resources for both Configured Tunnel and 6to4 mechanisms. Configured Tunnel on Fedora gives the lowest CPU utilisation. This aspect required further study in conjunction with throughputs achieved. For final finding, this feature needs to take into account the function being performed by the system (Router 1 and Router 2). Next section will presents and discusses the results of Router 1 CPU usage of all four operating systems.

5.1.5 TCP - Router2 CPU Utilization

Figure 5-12 below shows line chart of rotuer2 CPU utilisation with different packet size range from 64 bytes to 1536 bytes. The values of TCP Router 2 CPU Utilisation presented in Appendix E.

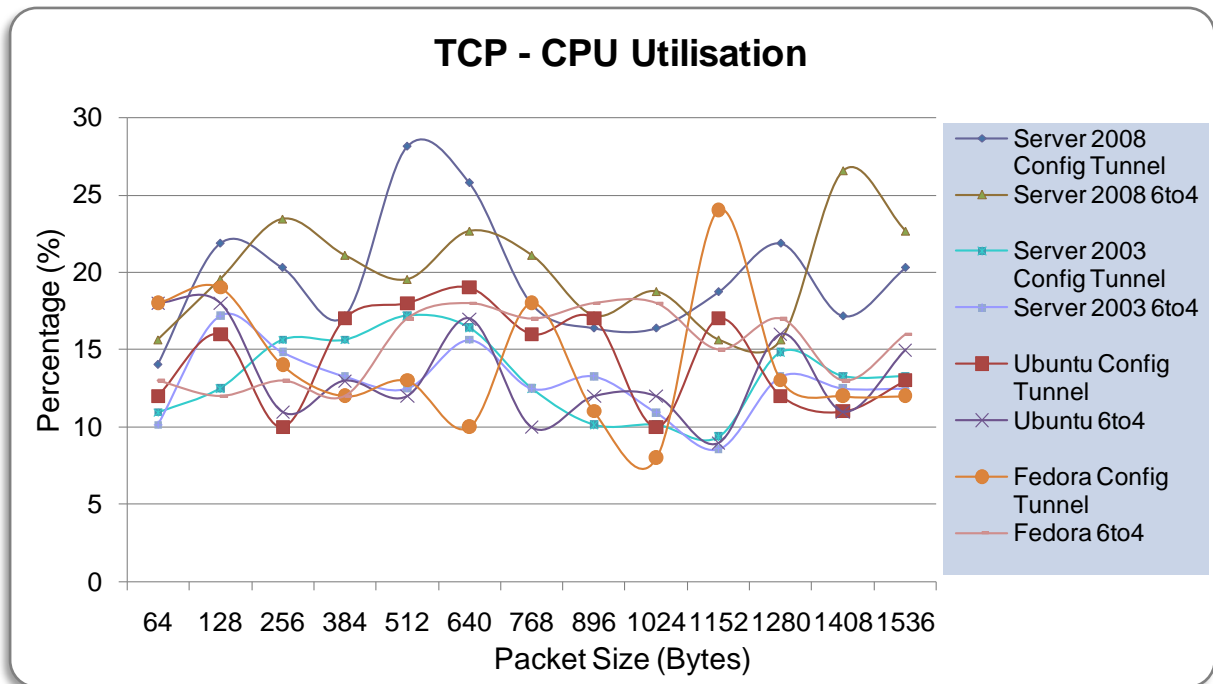


Figure 5-12: Router 2 - CPU Utilisation of TCP performance

From Figure 5-8 above, conclusion can be made as follows:

- At packet size 64 bytes:
 - Configured Tunnel on Fedora and 6to4 on Ubuntu, show the highest percentage of CPU usage.
 - Configured Tunnel and 6to4 on Windows Server 2003 shows the lowest percentage of CPU usage.
- At packet size of 128 bytes:
 - Configured Tunnel on Windows Server 2008 shows the highest percentage of CPU usage.
 - Configured Tunnel on Windows Server 2003 and 6to4 on Fedora shows the lowest percentage of CPU usage.
- At packet size of 256 bytes:

- 6to4 on Windows Server 2008 shows the highest percentage of CPU usage.
- Configured Tunnel and 6to4 shows the lowest percentage of CPU usage on Ubuntu.
- From packet sizes of 512 to 640 bytes:
 - Configured Tunnel shows the highest percentage of CPU usage on Windows Server 2008.
 - 6to4 shows the lowest percentage of CPU usage on Ubuntu.
- At packet size 1152 bytes, Configured Tunnel shows the highest percentage of CPU usage on Fedora.
- At packet size 1280 bytes, Configured Tunnel shows the highest percentage of CPU usage on Windows Server 2008.
- From packet sizes of 1408 to 1536 bytes, 6to4 shows the highest percentage of CPU Utilisation over Windows Server 2008.
- Overall, there is no clear evident show that which operating system used the highest CPU usage.

UDP results of Configured Tunnel and 6to4 will present and discuss next.

5.2 UDP Data Analysis

UDP data analysis is included UDP result of Configured Tunnel and 6to4 transition mechanisms on Windows Server 2003, Windows Server 2008, Ubuntu 9.10, and Fedora Core 11. Next section will presents UDP throughput performance of Configured Tunnel and 6to4 on all four operating systems.

5.2.1 UDP Throughput

This section presents the line charts and the analysis of UDP results for both Configured Tunnel and 6to4. The values of TCP throughput on each packet size presented in Appendix F.

5.2.1.1 Windows Server 2008 UDP Throughput Results

Figure 5-13 below shows line chart of UDP throughput for Configured Tunnel and 6to4 on Windows Server 2008 operating system with different packet sizes.

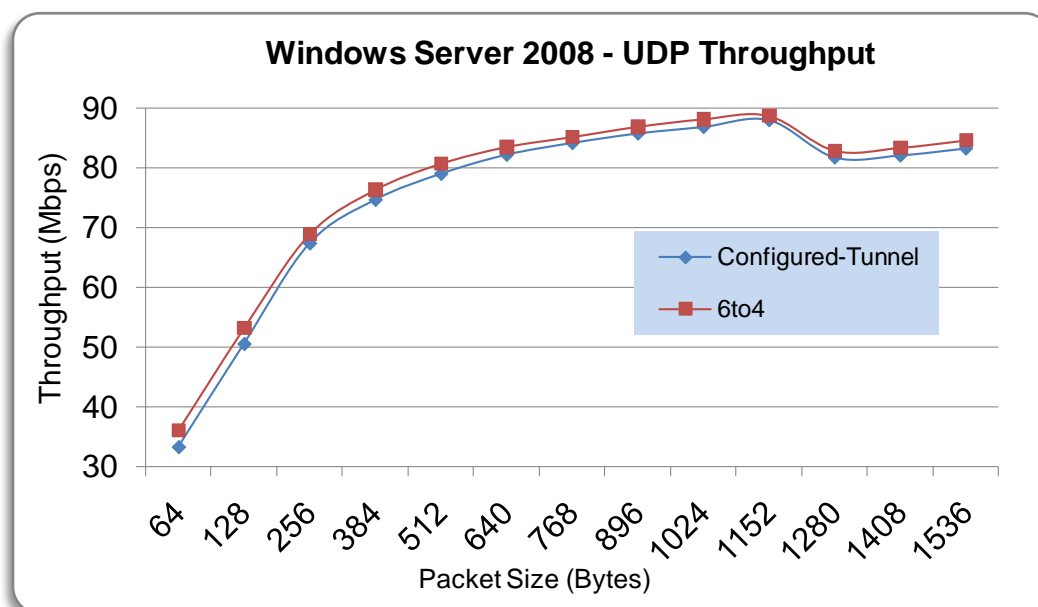


Figure 5-13: Configured Tunnel and 6to4 on Windows Server 2008 – UDP Throughput

- According to line chart in Figure 5-13 above, 6to4 produced 2% higher throughput than Configured Tunnel for every packet size ranging from 64 bytes to 1536 bytes.
- From the packet sizes of 64 bytes to 1152 bytes, there is a significant increase of 55% in throughput values for both Configured Tunnel and 6to4. However, there is an 8% drop in throughput when approaching the packet size of 1280 bytes and increase 5% in throughput when approaching packet size of 1536 bytes.
- UDP throughput for both Configured Tunnel and 6to4 achieved the highest throughput between 87 Mbps to 89 Mbps at the packet size of 1152 bytes.
- According to line chart in Figure 5-13 above, Configured Tunnel and 6to4 show the same pattern.
- Overall, 6to4 perform better than Configured Tunnel on Windows Server 2008.

Next section will present the results of Configured Tunnel and 6to4 UDP throughput on Windows Server 2003.

5.2.1.2 Windows Server 2003 UDP Throughput Results

Figure 5-14 below shows line chart of UDP throughput for Configured Tunnel and 6to4 on Windows Server 2003 operating system with different packet sizes.

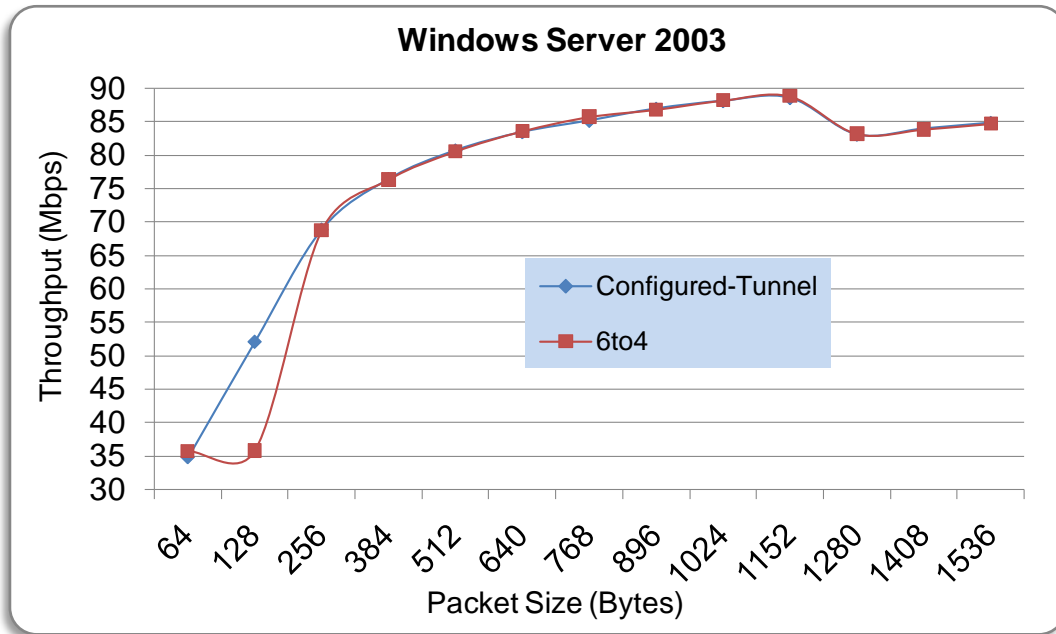


Figure 5-14: Configured Tunnel and 6to4 on Windows Server 2003 - UDP Throughput

- At the smallest packet size of 64 bytes, 6to4 produced 1% higher in throughput than Configured Tunnel.
- At packet size of 128 bytes, Configured Tunnel produced 17% higher in throughput than 6to4.
- As packet sizes increase from 256 bytes to 1536 bytes, 6to4 and Configured Tunnel produced similar throughput.
- From packet sizes of 64 bytes to 1152 bytes, there is a significant increase of 55% in throughput values for both Configured Tunnel and 6to4. However, there is a 7% drop when approaching the packet size of 1280 bytes and a 2% increase in throughput when approaching the packet size of 1536 bytes.
- UDP throughput for both Configured Tunnel and 6to4 achieved the highest throughput between 88 Mbps to 89 Mbps at the packet size of 1152 bytes.
- According to line chart in Figure 5-14 above, Configured Tunnel and 6to4 show the same pattern.

- Overall, Configured Tunnel and 6to4 produced similar throughput, except at a particular packet size of 128 bytes, Configured Tunnel produced the highest significant throughput.

Next section will presents and discusses the results of Configured Tunnel and 6to4 UDP throughput on Ubuntu 9.10.

5.2.1.3 *Ubuntu 9.10 UDP Throughput Results*

Figure 5-15 below shows line chart of UDP throughput for Configured Tunnel and 6to4 on Ubuntu 9.10 operating system with different packet sizes.

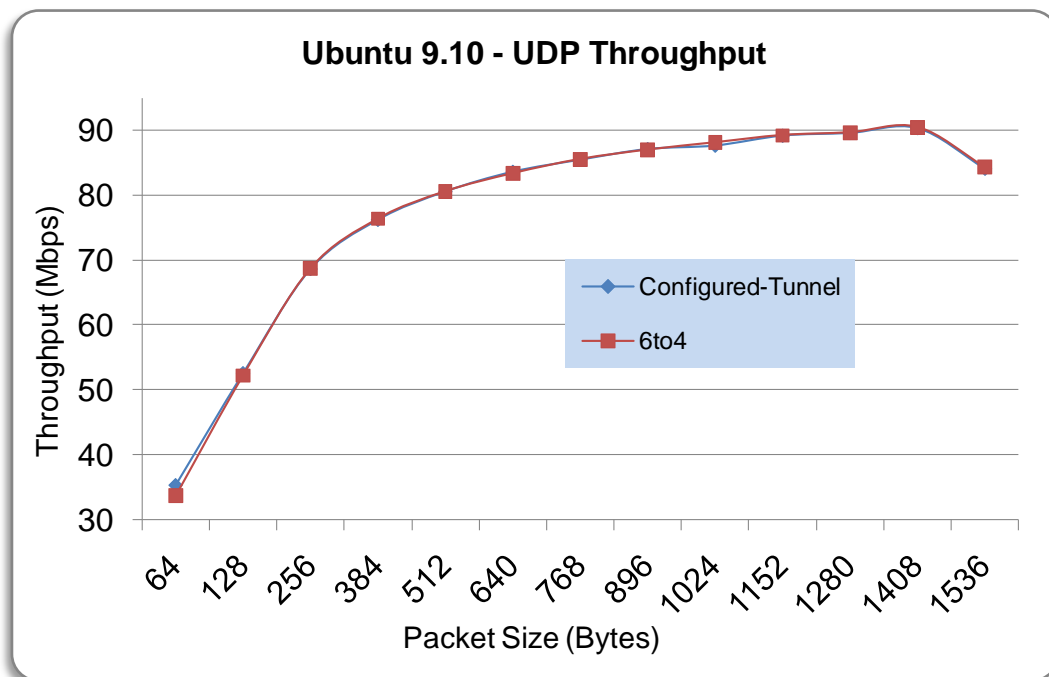


Figure 5-15: Configured Tunnel and 6to4 on Ubuntu - UDP Throughput

- For small packet sizes of 64 bytes and 128 bytes, Configured Tunnel produced 1% higher throughput than 6to4.
- As the packet sizes increase from 256 bytes to 384 bytes, 6to4 produced 0.2% higher throughput than Configured Tunnel.
- From packet sizes of 512 bytes to 640 bytes, Configured Tunnel produced 0.2% higher throughput than 6to4.

- When packet sizes increase from 1024 bytes to 1536 bytes, 6to4 produced 0.2% higher throughput than Configured Tunnel.
- From packet sizes of 64 bytes to 1408 bytes, there is a significant increase of 55% in throughput values for both Configured Tunnel and 6to4. However, there is a 7% drop in throughput when approaching the packet size of 1536 bytes.
- UDP throughput for both Configured Tunnel and 6to4 achieved the highest throughput between 90 Mbps to 91 Mbps at the packet size of 1408 bytes.
- According to line chart in Figure 5-15 above, Configured Tunnel and 6to4 show the same pattern.
- Overall, there is no significant different in UDP throughput between Configured Tunnel and 6to4. However, 6to4 performs slightly better than Configured Tunnel on most packet sizes.

Next section will presents and discusses the results of Configured Tunnel and 6to4 UDP throughput on Fedora 11.

5.2.1.4 Fedora 11 UDP Throughput Results

Figure 5-16 below shows line chart of UDP throughput for Configured Tunnel and 6to4 on Fedora 11 operating system with different packet sizes.

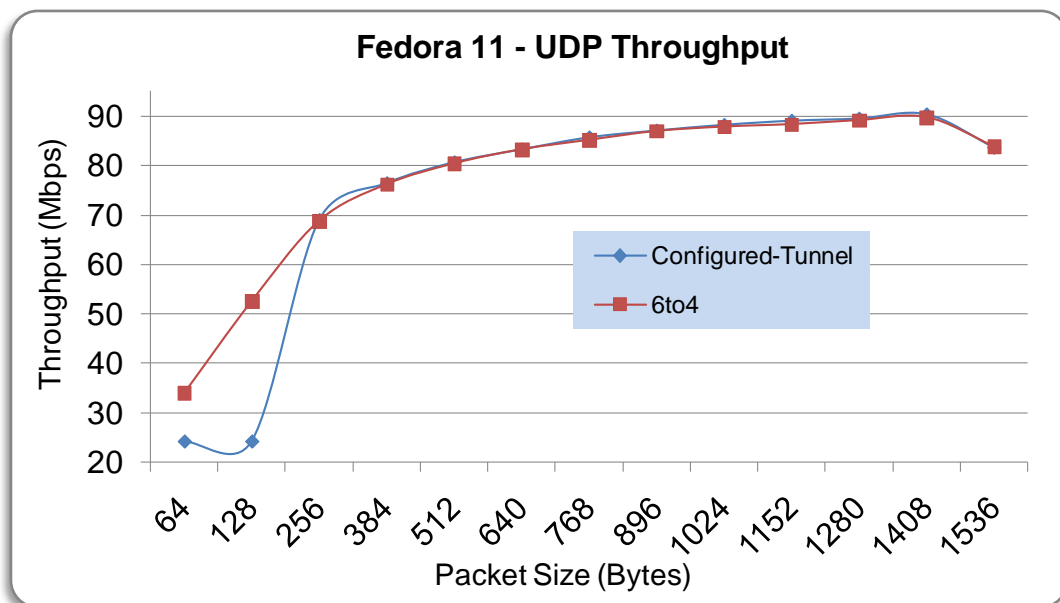


Figure 5-16: Configured Tunnel and 6to4 on Fedora 11 – UDP Throughput

- At packet size of 64 bytes, 6to4 outperforms Configured Tunnel by less than 10% in throughput values.
- As packet size increase to 128 bytes, the gap between 6to4 and Configured Tunnel become larger. At this packet size, 6to4 outperforms Configured Tunnel by 28% in throughput values. However, as the packet size increase, the gap between 6to4 and Configured Tunnel become small.
- From packet sizes of 256 bytes to 1408 bytes, Configured Tunnel produced less than 1% higher in throughput than 6to4. As the packet sizes increase to 1536 bytes, 6to4 produced slightly higher throughput than Configured Tunnel.
- From packet of 64 bytes to 1536 bytes, the graph shows a significant increase in throughput with an average increase of 60% for both Configured Tunnel and 6to4.
- UDP throughput for both Configured Tunnel and 6to4 achieved the highest throughput between 89 Mbps and 91 Mbps at the packet size of 1408 bytes.

UDP results of Configured Tunnel and 6to4 on each of the selected operating systems have been discussed and analysed above. The comparison results between Configured Tunnel and 6to4 on all four operating systems will present in the next section.

5.2.1.5 Compare Four Operating systems TCP Throughput Result

Figure 5-17 below shows line chart of UDP throughput with different packet sizes ranging from 64 bytes to 1536 bytes.

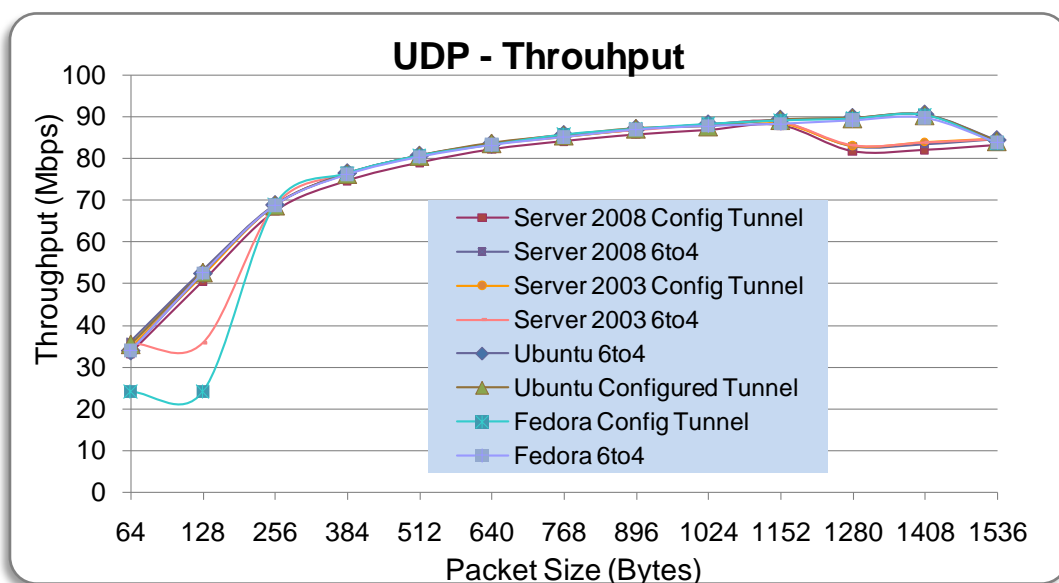


Figure 5-17: UDP Throughput Result

- At the smallest packet size of 64 bytes, Configured Tunnel on Fedora produced the lowest UDP throughput. On the other hand, 6to4 on Windows Server 2008 produced the highest UDP throughput with the difference of 10%.
- As the packet size increase to 128 bytes, Configured Tunnel on Fedora again has the lowest UDP throughput and 6to4 on Windows Server 2008 produced the highest throughput but this time the difference has been increase from 10% to 30% in throughput value.
- From packet sizes of 64 to 1152 bytes, the graph shows a significant incline in UDP throughput with an average increase of 55% for both transition mechanisms on all four operating systems.
- From packet sizes of 256 to 1152 bytes, both tunnelling mechanisms on all four operating systems produced similar UDP throughput.
- From packet sizes of 1152 to 1408 bytes, both tunnelling mechanisms perform better on both Linux operating systems than on both Windows operating systems with the differences of approximately 8%.However, 6to4 on Ubuntu produced the highest throughput and Configured Tunnel on Windows Server 2008 produced the lowest throughput.
- From packet size of 1152 to 1536 bytes, both tunnelling mechanisms on both Windows operating systems show a significant dropped of 8% in throughput values. On the other hand, both tunnelling mechanisms on both Linux operating systems show a slight increase in throughput value until it reached the packet size of 1408 bytes then the graph start to incline, which shows a significant dropped in throughput at the packet size of 1536 bytes.
- At the largest packet size of 1536 bytes, both tunnelling mechanisms produced similar UDP throughput on all four operating systems.
- Overall, 6to4 produced the highest throughput.

The results of Configured Tunnel and 6to4 UDP jitter on all four operating systems will present in the following section.

5.2.2 UDP Jitter

Figure 5-18 below shows line chart of UDP jitter with different packet size range from 64 bytes to 1536 bytes.

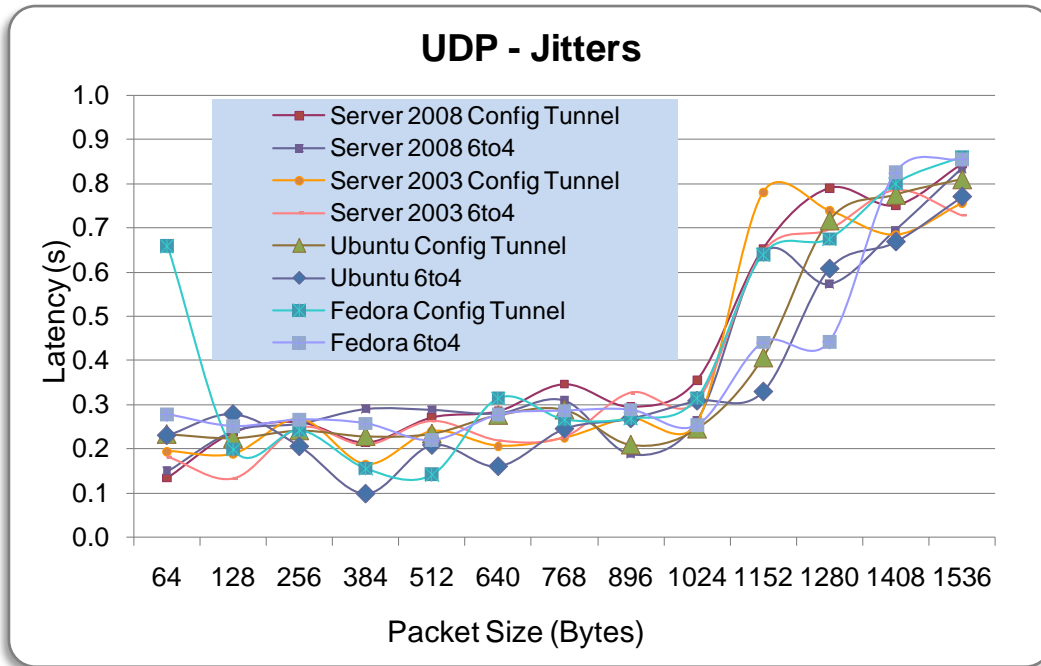


Figure 5-18: UDP Jitter Result

- At packet size of 64 bytes, both transition mechanisms have similar UDP jitter on all four operating systems except Configured Tunnel on Fedora, which produced the highest UDP jitter.
- From packet sizes of 128 bytes to 1024 bytes, both transition mechanisms have similar UDP jitter on all four operating systems.
- From packet sizes of 1024 bytes to 1536 bytes, both transition mechanisms on all four operating systems show a significant incline with an average of 50% increase.
- At packet size of 1152 bytes, Configured Tunnel produced the highest UDP jitter on Windows Sever 2003 and 6to4 on Ubuntu produced the lowest UDP jitter.
- At packet size 1280 bytes, Configured Tunnel produced the highest UDP jitter on Windows Server 2008 and 6to4 produced the lowest UDP jitter on Ubuntu.
- From packet sizes of 1408 bytes to 1536 bytes, both transition mechanisms produced similar UDP jitter on all four operating systems.

The results of Configured Tunnel and 6to4 UDP delay on all four operating systems will present in the following section.

5.2.3 UDP Delay

Figure 5-19 below shows line chart of UDP delay with different packet size range from 64 bytes to 1536 bytes.

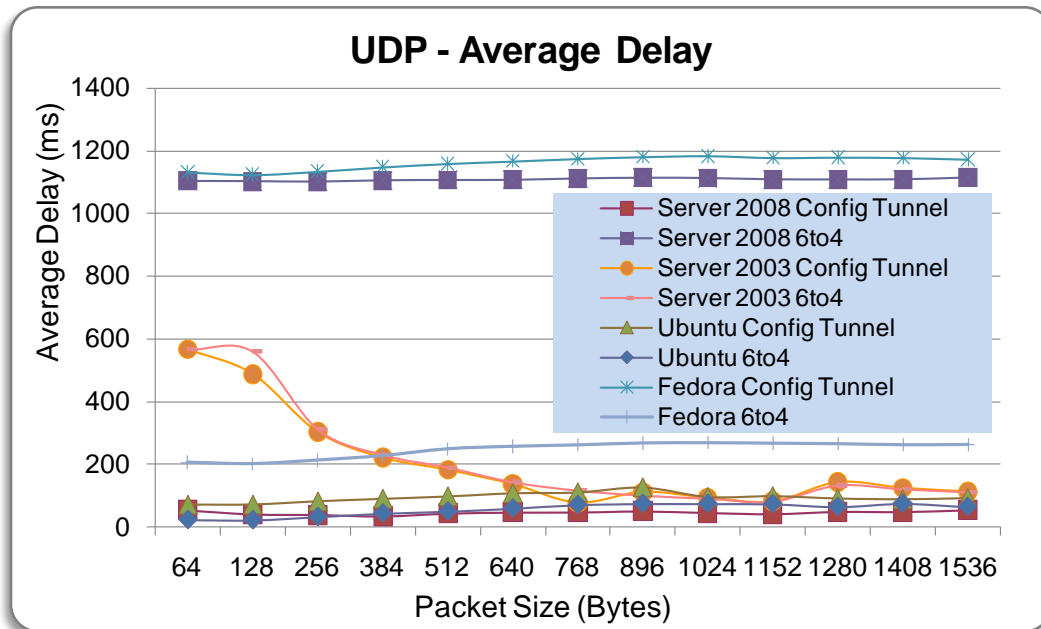


Figure 5-19: UDP Average Delay Result

- From packet sizes of 64 bytes to 1536 bytes:
 - Configured Tunnel on Fedora has similar delay performance to 6to4 on Windows Server 2008 with the differences of less than 0.2%.
 - 6to4 on Windows Server 2008 and Configured Tunnel on Fedora have the highest delay with an average of 1000 ms.
- From packet sizes of 64 bytes to 1536 bytes, Configured Tunnel on Windows Server 2008 and 6to4 on Ubuntu have the lowest delay.
- From packet sizes of 64 bytes to 768 bytes, Configured Tunnel and 6to4 on Windows Server 2003 shows a significant drop in UDP delay with an average of approximately 20%.
- Overall, 6to4 on Windows Server 2008 and Configured Tunnel on Fedora produced the highest delay.

The results of Configured Tunnel and 6to4 Router 1 CPU usage on all four operating systems will present in the following section.

5.2.4 UDP Router1 CPU Utilization

Figure 5-20 below shows line chart of router1 CPU utilisation with different packet size range from 64 bytes to 1536 bytes.

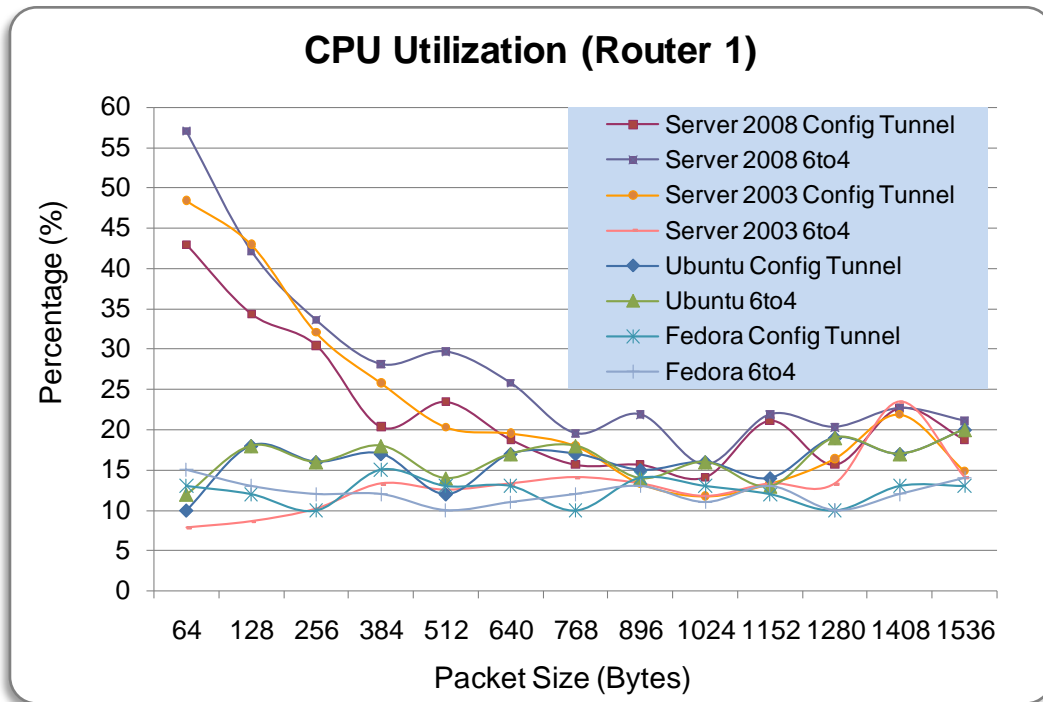


Figure 5-20: Router 1 – CPU Utilisation of TCP performance

- At packet size 64 bytes, 6to4 shows the highest percentage of CPU usage on Windows Server 2008 and 6to4 on Windows Server 2003 shows the lowest percentage of CPU usage.
- As packet sizes increase from 64 bytes to 1024 bytes, Configured Tunnel on both Windows operating systems and 6to4 on Windows Server 2008 show a significant decline in percentage of CPU usage with a drop average of 35%.
- At packet size 128 bytes, Configured Tunnel shows the highest percentage of CPU usage on Windows Server 2003 and 6to4 on Windows Server 2003 shows the lowest percentage of CPU usage.
- As packet sizes increase from 256 bytes to 896 bytes, 6to4 shows the highest percentage of CPU usage on Windows Server 2008.
- From packet size 1024 bytes and 1408 bytes, 6to4 shows highest percentage of CPU usage on Ubuntu.

The results of Configured Tunnel and 6to4 Router 2 CPU usage on all four operating systems will present in the following section.

5.2.5 UDP Router2 CPU Utilization

Figure 5-21 below shows line chart of router2 CPU utilisation with different packet size range from 64 bytes to 1536 bytes.

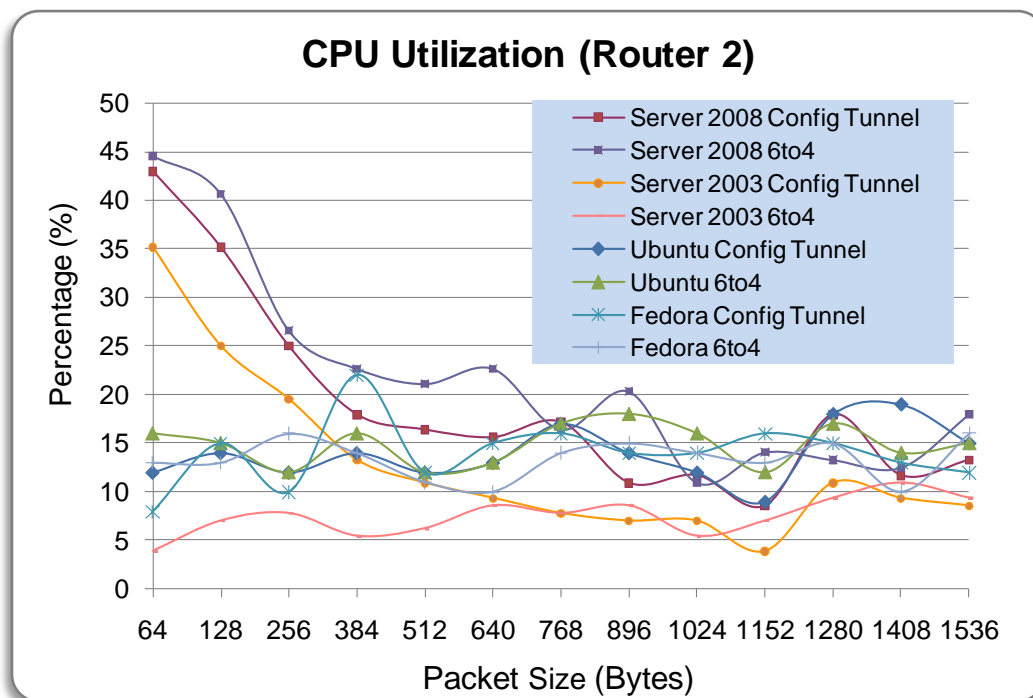


Figure 5-21: Router 2 – CPU Utilisation of TCP performance

- From packet sizes of 64 bytes to 640 bytes:
 - 6to4 has highest percentage of CPU usage on Windows Server 2008.
 - 6to4 has lowest percentage of CPU Utilisation on Windows Server 2003.
 - 6to4 on Windows Server 2008, Configured Tunnel on Windows Server 2008, and Configured Tunnel on Windows Server 2003 show significant decrease in percentage of CPU usage.
- From packet sizes of 640 bytes to 1536 bytes, Configured Tunnel on Windows Server 2003 shows the lowest percentage except at packet sizes of 1024 bytes and 1280 bytes, 6to4 on Windows Server 2003 shows the lowest percentage of CPU usage.

The comparison results TCP and UDP will present in the following section.

5.3 Comparison of TCP and UDP

This section presents the comparison between TCP performance and UDP performance by combining TCP and UDP line charts into one single line chart to identify the performance differences between these two transmission protocols.

5.3.1 UDP and TCP Throughput

Figure 5-14 below shows line chart of the comparison between UDP and TCP throughput with different packet size range from 64 bytes to 1536 bytes.

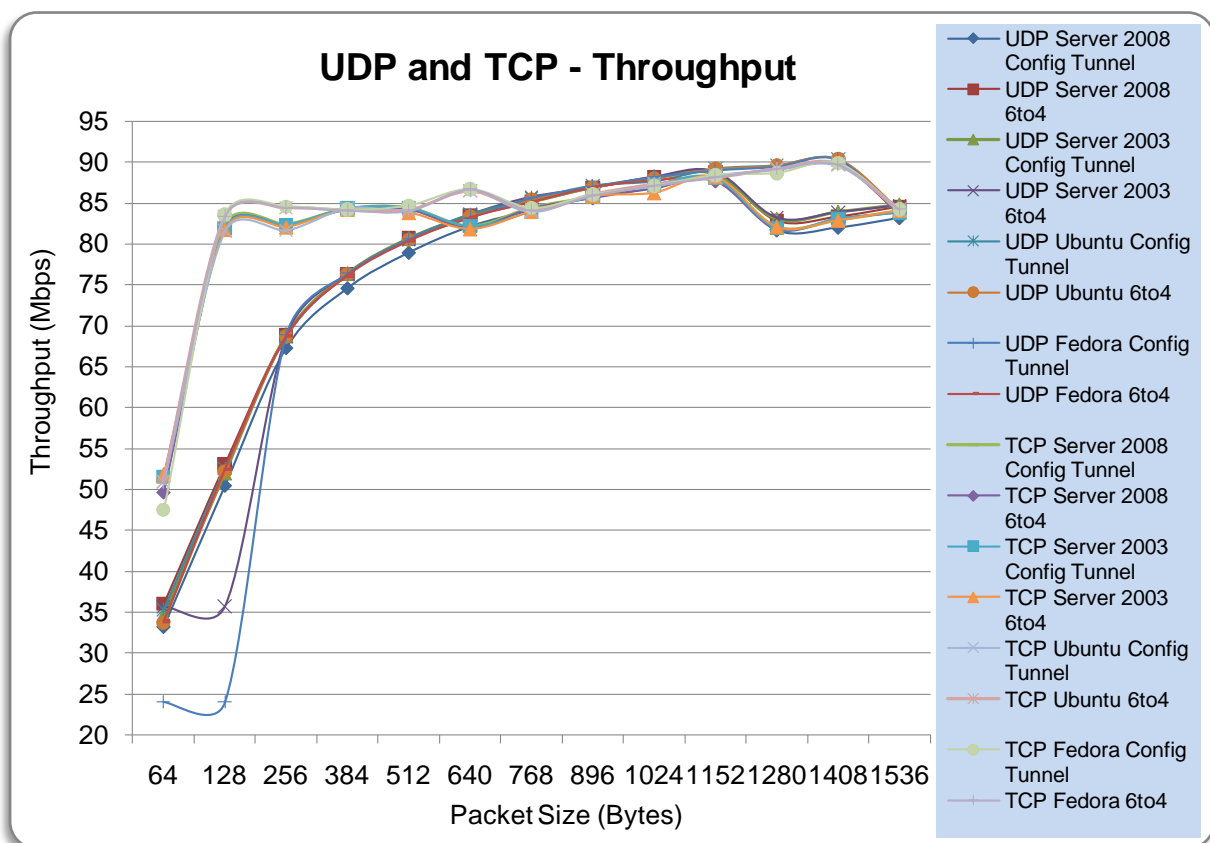


Figure 5-22: UDP and TCP Throughput Result

From Figure 5-22 above, conclusion can be drawn as follows:

- From packet sizes of 64 bytes to 512 bytes, TCP have higher throughput than UDP for both transition mechanisms on all four operating systems.
- At packet size 64 bytes, the average difference between TCP and UDP throughput is 12%.
- At packet size of 128 bytes, the average difference between TCP and UDP throughput increased from 12% to approximately 30%. However, at packet size of 256 bytes, the average difference between TCP and UDP throughput has decreased from 30% to approximately 13%.
- At packet size of 384 bytes, the average difference between TCP and UDP throughput has decreased to approximately 10%.
- At packet size of 512 bytes, the average difference between TCP and UDP throughput decreased to approximately 5%.
- From packet sizes of 64 bytes to 512 bytes, the graph shows that both TCP and UDP show significant incline in throughput. As the packet sizes increase, UDP throughput values become closer to TCP throughput values.
- Between packet sizes of 512 bytes to 768 bytes, 6to4 and Configured Tunnel on both Linux operating systems have the highest throughput. However, Fedora tends to perform slightly better than Ubuntu with the difference of approximately 0.2%. Compare to the rest of operating systems, 6to4 and Configured Tunnel on both Linux have an average of 3% higher throughput.
- From packet sizes of 768 bytes to 1152 bytes, TCP and UDP have similar throughput values.
- From packet sizes of 1152 bytes to 1536 bytes, line chart for both TCP and UDP throughput show the same pattern. Between packet sizes of 1152 bytes to 1536 bytes, TCP and UDP of both tunnelling mechanisms have similar throughput values on both Linux operating systems. In addition, both tunnelling mechanisms on both Linux have the highest throughput values. However, UDP of Configured Tunnel has the lowest throughput on Windows Server 2008.
- At packet size of 1536 bytes, TCP and UDP have similar throughput with the difference of less than 1%.
- From packet sizes of 768 bytes to 1536 bytes, both TCP and UDP throughput shows similar pattern.
- Overall, for small packet sizes, TCP produced higher throughput than UDP. However, for large packet size UDP produced higher throughput than TCP. Comparison between UDP and TCP jitter will present in the following section.

5.3.2 UDP and TCP Jitter

Figure 5-23 below shows line chart of the comparison between UDP and TCP jitter with different packet size range from 64 bytes to 1536 bytes.

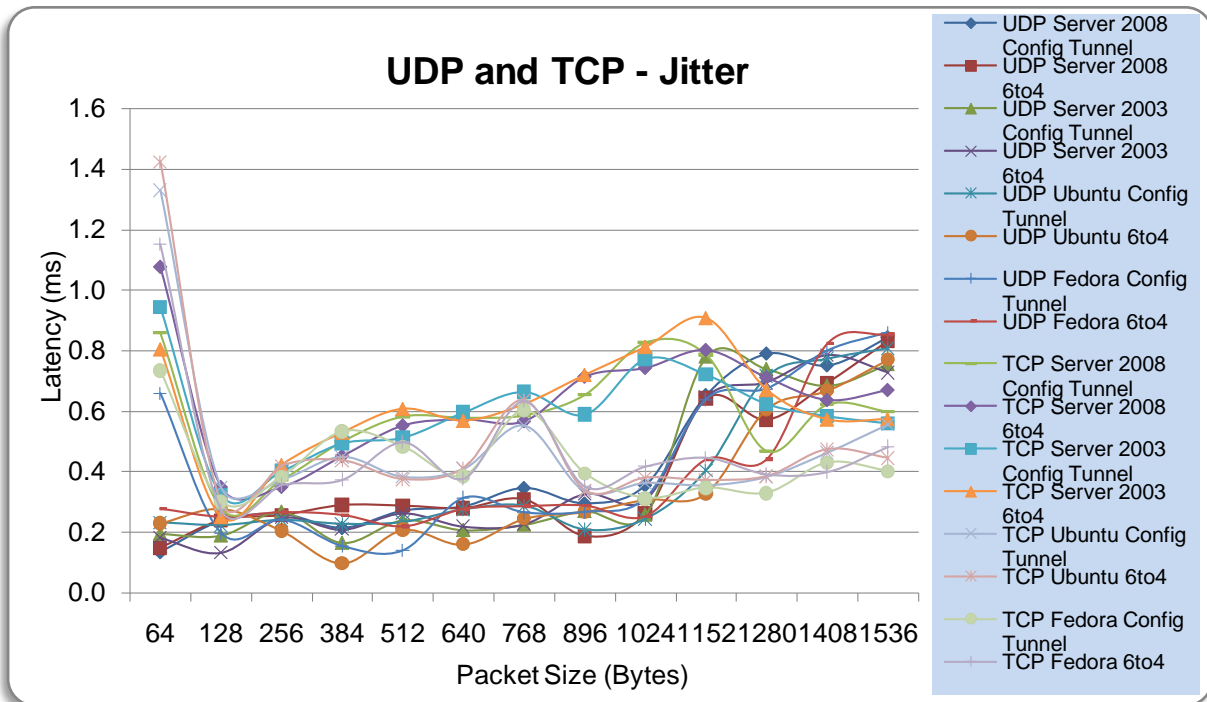


Figure 5-23: UDP and TCP Jitter

- At packet size of 64 bytes, TCP jitter produced higher latency than UDP jitter with an average difference of less than 30%.
- From packet sizes of 64 bytes to 1152 bytes, UDP has lower latency than TCP.
- As the packet sizes increase from 1280 bytes to 1536 bytes, TCP has lower latency than UDP.
- From packet sizes of 128 bytes to 1024 bytes, UDP tend to produce lower latency than TCP.
- As packet sizes become larger, latency of UDP seem to increase significantly.
- As the packet sizes increase from 1152 bytes to 1536 bytes, both Linux operating systems show the lowest latency.
- Overall, UDP produced lower jitter than TCP from packet sizes ranging from 64 bytes to 1024 bytes. However, both Linux operating systems produced lower latency as the getting larger. The comparison between UDP and TCP delay will present in the following section.

5.3.3 UDP and TCP Delay

Figure 5-24 below shows line chart of the comparison between UDP and TCP delay with different packet size range from 64 bytes to 1536 bytes.

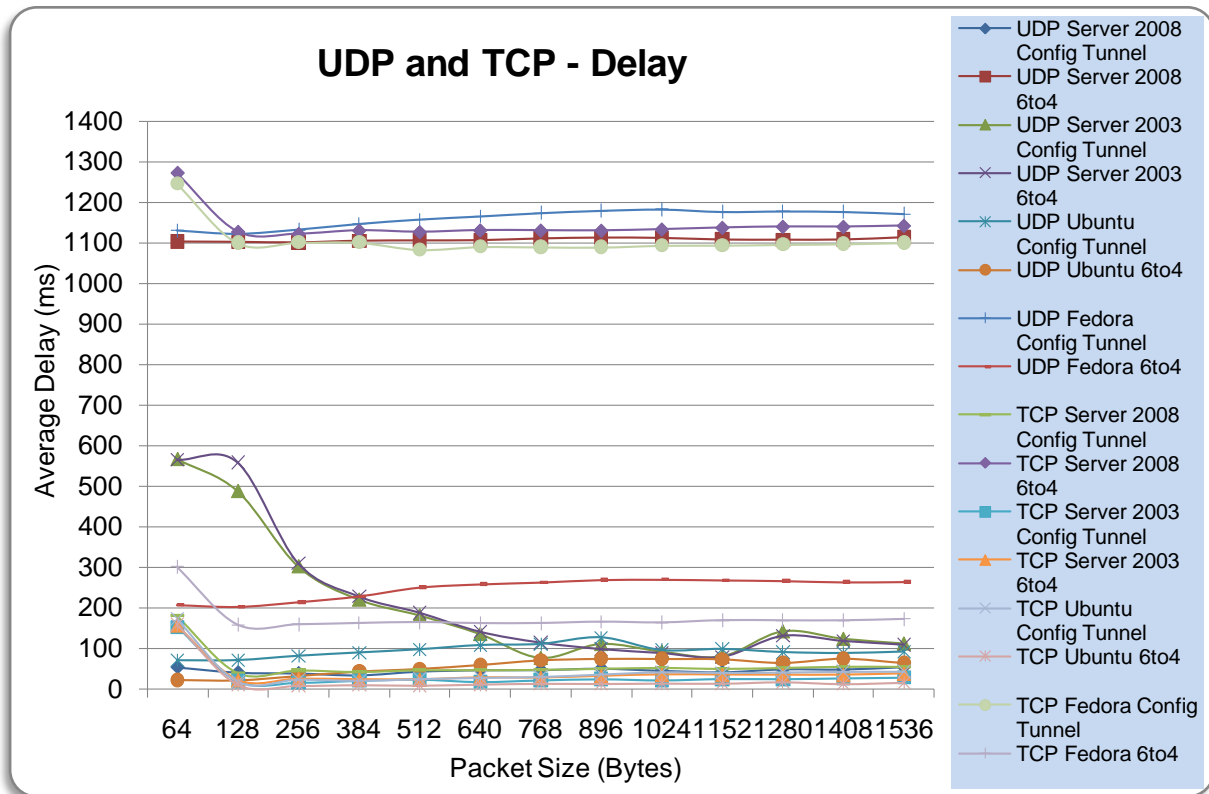


Figure 5-24: UDP and TCP Delay

- At small packet size of 64 bytes, Configured Tunnel on Windows Server 2008 using UDP has the lowest delay. On the other hand, Configured Tunnel on Fedora using TCP has the highest delay.
- As the packet size increase from 128 bytes to 1536 bytes, 6to4 on Ubuntu with TCP has the lowest delay. On the other hand, Configured Tunnel on Fedora using UDP has the highest delay.
- Over all, 6to4 on Windows Server 2008 and Configured Tunnel on Fedora 11 for both TCP and UDP produced the highest significant delay.

The comparison between UDP and TCP Router 1 CPU usage will presents in the following section.

5.3.4 UDP and TCP Router 1 CPU Utilisation

Figure 5-14 below shows line chart of the comparison between UDP and TCP Router 1 CPU utilisation with different packet size range from 64 bytes to 1536 bytes.

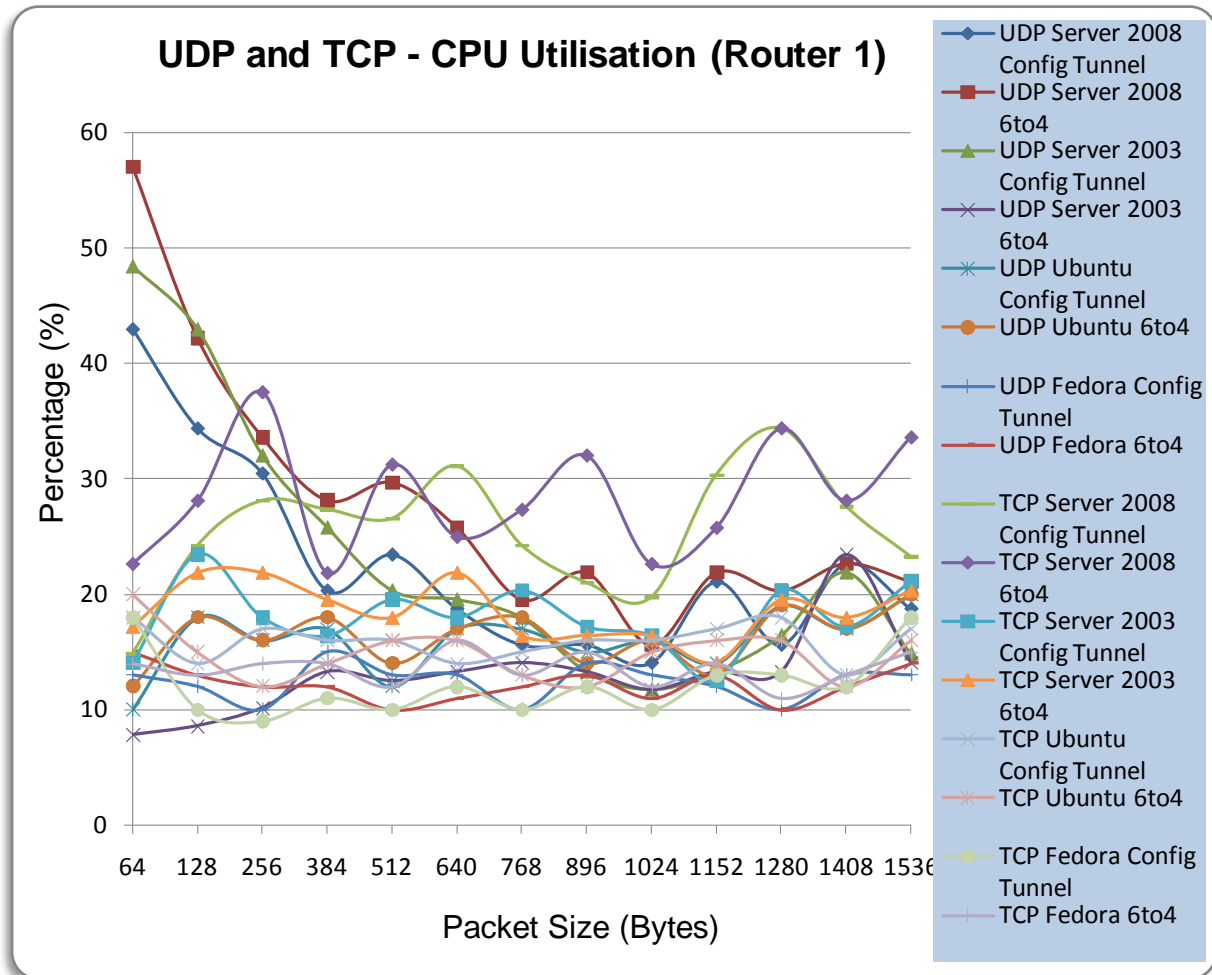


Figure 5-25: TCP and UDP Router 1 CPU Utilisation

From Figure 5-25 above, conclusion can be made as follows:

- From the smallest packet size of 64 bytes to the largest packet size of 1536 bytes, Configured Tunnel and 6to4 on Windows Server 2008 used the highest percentage of CPU resources for both TCP and UDP.

The comparison between UDP and TCP Router 2 CPU usage will presents in the following.

5.3.5 UDP and TCP Router 2 CPU Utilisation

Figure 5-14 below shows line chart of the comparison between UDP and TCP Router 2 CPU utilisation with different packet size range from 64 bytes to 1536 bytes.

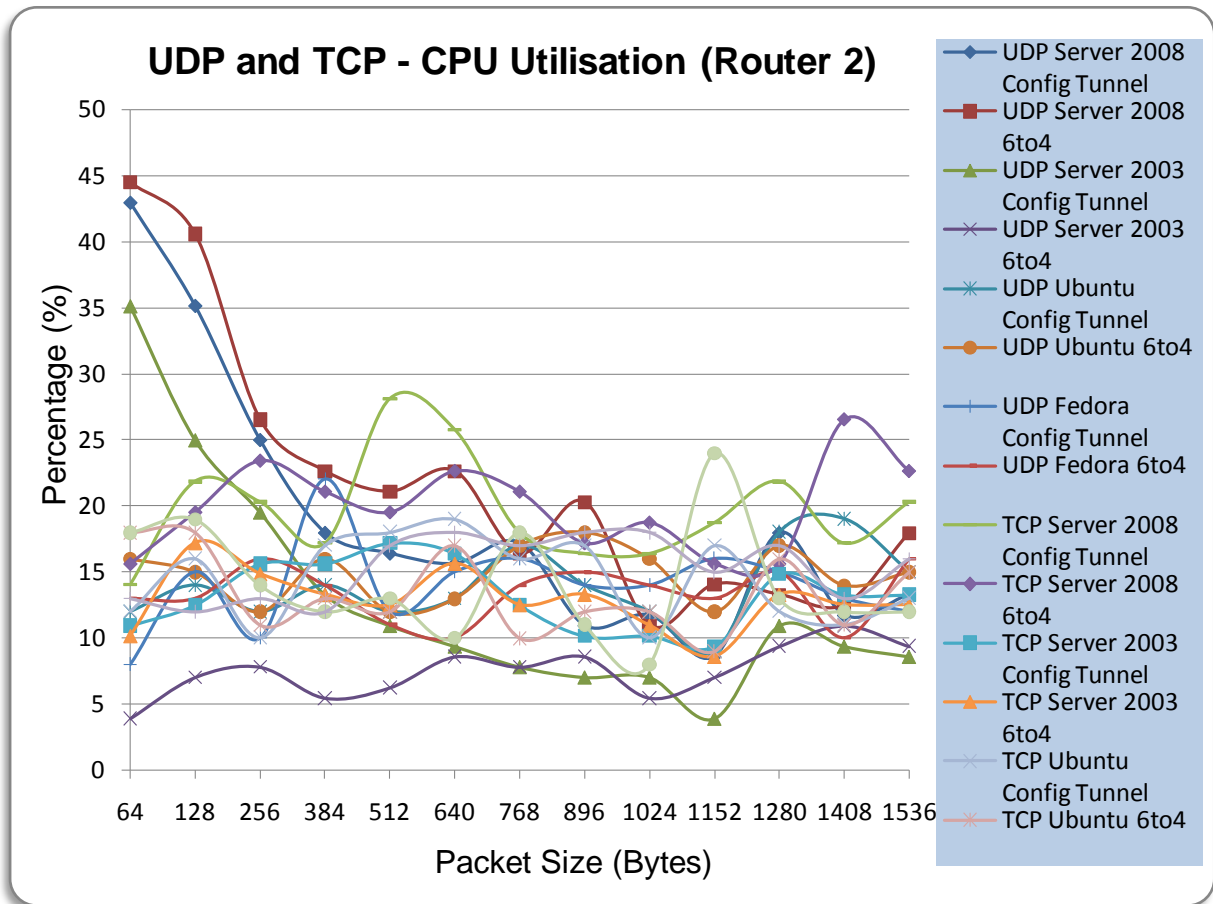


Figure 5-26: TCP and UDP Router 2 CPU Utilisation

- From packet sizes of 64 bytes to 384 bytes, UDP of 6to4 on Windows Server 2008 used the highest CPU resources.

5.4 Summary

This chapter presents the analysis of the experimental results in line charts and comparing the performance between Configured Tunnel and 6to4 transition mechanisms on four operating systems (Windows Server 2008, Windows Server 2003, Linux Ubuntu 9.10, and Linux Fedora Core 11) based on TCP and UDP transmission protocols. The analysis contained three sections included TCP analysis, UDP analysis, and TCP and UDP comparison. Discussion of this chapter will be covered in the next chapter.

6 Chapter 6: Discussion

6.1 Discussion of Findings

This chapter discusses the findings from the results of the experiment of this research. The scope of this research is to evaluate performance differences between 6to4 and Configured Tunnel on four selected operating systems. These four operating systems are Microsoft Windows operating systems (Windows Server 2003 & Windows Server 2008) and Linux operating systems (Ubuntu 9.10 & Fedora 11). Four parameters were examined which included throughput, jitter, delay, and CPU utilisation. D-ITG was the selected performance measurement tool used for the experiment. The experimental results gathered by generating network traffics from D-ITG sender to DI-ITG receiver using thirteen different packet sizes ranging from 64 bytes to 1536 bytes. The following sections discuss the findings of the analysis results found in chapter five above.

6.1.1 TCP Performance

The analysis results of TCP throughput for each operating system presented in chapter five above show that, Configured Tunnel has 0.2% higher throughput than 6to4 on all four operating systems. Furthermore, the graphs above show obvious patterns for both transition mechanisms on different operating systems. Configured Tunnel and 6to4 appeared to have almost identical pattern on both Windows Server operating systems and have almost identical pattern on both Linux operating systems.

Both tunnelling mechanisms on both Windows Server operating systems achieved the highest TCP throughput of approximately 87 to 89 Mbps at the packet size of 1152 bytes and drop by 6% at the packet size of 1280 bytes. On the other hand, TCP throughput for both tunnelling mechanisms on Ubuntu 9.10 and Fedora 11 achieved the highest throughput of approximately 88 to 89 Mbps at the packet size of 1408 bytes and drop by 5% at packet size of 1536 bytes. This can be concluded that, both Windows Servers and both Linux operating systems achieved the highest throughput at different packet sizes. Windows Server operating systems reached the highest throughput at the packet size of 1152 bytes, which is earlier than Linux operating systems that reached the highest throughput at the packet size of 1408 bytes.

The comparison results in chapter five above show that, both tunnelling mechanisms on both Linux operating systems outperform the two Windows operating systems for most packet sizes. However, overall results show that Configured Tunnel on Fedora 11 has slightly better performance than both transition mechanisms on Ubuntu 9.10 and 6to4 on Fedora 11. The graph of TCP throughput shows that, at the packet size of 1536 bytes every line chart on the graph is almost intercept at the same point.

The discussion with regards the relationship between throughput and jitter in a way that how jitter can affect the performance of throughput is presented below.

- Both transition mechanisms on both Windows operating systems produced higher jitter than both transition mechanisms on both Linux operating systems.
- For most packet sizes, Configured Tunnel on Fedora 11 produced the lowest jitter especially for the packet sizes ranging from 1024 bytes to 1536 bytes except on packet size 1408 bytes.

After investigating the relationship between throughput and jitter, the outcome shows that jitter is a factor that causes degradation on network performance and particularly, jitter affected network throughput.

In summary, both tunnelling mechanisms on both Linux operating systems produce higher throughput than both tunnelling mechanisms on both Windows operating systems especially for large packet sizes ranging from 1280 bytes to 1408bytes, which shows a significant difference of 6%.Configured Tunnel on Fedora 11 produced the lowest jitter for most packet sizes, which is obvious that Configured Tunnel has better performance. However, the differences in throughput between both tunnelling mechanisms on both Linux operating systems are less than 0.3%, which does not show a significant difference.

The most significant finding from this research is the delay produced by Configured Tunnel on Fedora 11 and 6to4 on Windows Server 2008. Configured Tunnel on Fedora 11 and 6to4 on Windows Server 2008 produced the highest delay, which is evidence that these are not suitable for voice and video packet due to voice and video packets are delay sensitive. Considering that, most voice and video applications would use UDP for transport. The discussion in the next section should be of interest.

6.1.2 UDP Performance

The analysis results of UDP throughput for each operating system presented in chapter five above show that, 6to4 outperforms Configured Tunnel by 2% for Windows Server 2008 on every packet size. However, for Windows Server 2003, Ubuntu 9.10, Fedora 11, 6to4 performs slightly better than Configured Tunnel.

Both tunnelling mechanisms on both Windows Server operating systems achieved the highest UDP throughput at a range of 87 to 89 Mbps at packet size 1152 bytes and drop by 7% at packet size 1280 bytes. On the other hand, on Ubuntu 9.10 and Fedora 11, UDP throughput for both tunnelling mechanisms achieved the highest throughput at a range of 89 to 91 Mbps at the packet size of 1408 bytes and drop by 6% at packet size 1536 bytes. This can be concluded that UDP achieved the highest throughput at the same packet sizes as TCP.

The comparison results in chapter five above show that, both tunnelling mechanisms on both Linux operating systems outperform both tunnelling mechanisms on both Windows operating systems for most packet sizes. However, overall results show that 6to4 on Ubuntu 9.10 has slightly better performance than any other operating systems.

After investigating the relationship between throughput and jitter, this can be concluded that jitter causes degradation on network performance, throughput in particular. The analysis of jitter results show that:

- For most packet sizes, 6to4 on Ubuntu 9.10 produced the lowest jitter especially for the packet sizes ranging from 1024 bytes to 1536 bytes.

The above discussion shows evidence that 6to4 on Ubuntu 9.10 operating system produced the highest throughput, especially for large packet sizes ranging from 1280 to 1536 bytes.

The most significant finding from this research is the delay produced by Configured Tunnel on Fedora 11 and 6to4 on Windows Server 2008. Configured Tunnel on Fedora 11 and 6to4 on Windows Server 2008 produced the most significant highest delay, which is an evident that these are not suitable for voice and video packet. In this case, 6to4 on Ubuntu 9.10 is an ideal solution for data, voice, and video over Fedora, Windows Server 2003, and Windows Server 2008.

6.1.3 UDP and TCP Comparison

The comparison between UDP and TCP results for both tunnelling mechanisms on all four operating systems in chapter five above show that, for small packet sizes ranging from 64 to 512 bytes, TCP outperforms UDP with an average of 20% in throughput values, which shows the significant difference. As the packet sizes increased from 640 to 1152 bytes, TCP and UDP throughput are almost identical.

From packet sizes of 1152 to 1408 bytes, line charts of both TCP and UDP show an interesting pattern. The graph clearly shows that, at this packet sizes range, TCP and UDP for both Linux operating systems show the same pattern whereas, TCP and UDP also show the same pattern on both Windows Server operating systems. The results can be concluded that, for small packet sizes, TCP outperforms UDP with significant different in throughput values. For large packet sizes, both TCP and UDP show the same pattern and produced similarity in throughput result.

6.1.4 Summary of Findings

From the discussion of TCP analysis, the findings can be summarised as below:

- Configured Tunnel performs slightly better than 6to4 on each of the selected operating system.
- In comparison between Configured Tunnel and 6to4 on all four operating systems, both transition mechanisms on both Linux operating systems produced higher throughput results than on both Windows Server operating systems.
- 6to4 on Windows Server 2008 produced the highest delay.
- Out of four selected operating systems, Configured Tunnel on Fedora has the best throughput result. However, it produced the highest delay that almost identical to 6to4 on Windows Server.
- However, it is quite interesting to see that the graphs show the same pattern in Configured Tunnel and 6to4 on Windows Server operating systems. The graph also shows the same pattern in Configured Tunnel and 6to4 on Linux operating systems.

From the discussion of UDP analysis, the findings can be summarised as below:

- On each individual operating system, 6to4 performs slightly better than Configured Tunnel.

- In comparison between Configured Tunnel and 6to4 on all four operating systems, both tunnelling mechanisms on both Linux operating systems produced higher throughput results than on both Windows Server operating systems.
- 6to4 on Windows Server 2008 produced the highest significant delay.
- Configured Tunnel on Fedora produced the highest delay that almost identical to 6to4 on Windows Server 2008 which result in poor performance for voice and video packets.
- Out of all selected operating systems, 6to4 on Ubuntu 9.10 has the best throughput result with low delay. This result is an evident that 6to4 on Ubuntu 9.10 is ideally best for data, voice, and video packets.
- It is interesting to see the pattern of the graph, which shows that line charts of both Configured Tunnel and 6to4 on Windows Server operating systems are almost identical. The graph also shows line charts of both transition mechanisms are almost identical on both Linux operating systems.

From the discussion of TCP and UDP comparison, the findings can be summarised as below:

- For small packet sizes ranging from 64 to 512 bytes, TCP yields the best throughput performance. However, as the packet sizes increase from 512 to 1408 bytes, UDP yields better throughput performance than TCP.
- 6to4 on Windows Server 2008 and Configured Tunnel on Fedora produced almost identical and show the highest significant delay results on both UDP and TCP.

6.2 Future Study

This research presented the study of performance differences in throughput, jitter, delay, and CPU utilisation of Configured Tunnel and 6to4 on two Windows Server operating systems (Windows Server 2008 and Windows Server 2003) and two Linux operating systems (Ubuntu 9.10 and Fedora Core 11) based on UDP and TCP transmission protocol. The result of this study is a reference guide for network engineers to have a brief idea of the performance of the two tunnelling mechanisms on all four operating systems. This study can be extended by:

- Select different IPv4/IPv6 transition mechanisms such as ISATAP, Terado, NAT-PT, and DSTM.
- Changing to hardware router or comparing between software and hardware routers.
- Add multiple measurement tools to the experiment.
- Measure different network traffic types (VoIP, DNS).
- Conduct the experiment by using different network design such as host-to-host, host-to-router, and router-to-host.

7 Chapter 7: Conclusion

This research was conducted on the evaluation of IPv4/IPv6 transition mechanisms over various operating systems in the purpose of studying the performance differences between two tunnelling mechanisms on four operating systems. This study is an experimental base research, which focused on two tunnelling mechanisms (Configured Tunnel and 6to4) and four selected operating systems (Windows Server 2003, Windows Server 2008, Linux Ubuntu 9.10, and Linux Fedora Core 11). Both transition mechanisms were implemented on all four operating systems as per the experimental design and setup discussed in this study.

The study focused on two types of network traffic, known as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Performance metrics used in this study are throughput, jitter, delay, and CPU utilisation. According to the evaluation of related studies in chapter two above, this research has not yet been study as on the commencement of this research. After having completed the experiments and data analysis, the summary of this research is as follow:

- On each of the selected operating systems, Configured Tunnel performs better than 6to4 for TCP traffic. On the other hand, 6to4 performs better than Configured Tunnel for UDP traffic.
- The performance comparison between Windows operating system family and Linux operating system family shows that, Configured Tunnel and 6to4 on both Linux operating systems perform better than on both Windows operating systems.
- The result of TCP and UDP comparison shows that, TCP performs better than UDP for small packet sizes. However, for large packet sizes, UDP performs better than TCP.

Considering the research hypotheses, the above experiments help in arriving at the following conclusions:

Hypothesis 1: There are performance differences between different IP version 4 and IP version 6 transition mechanisms.

The findings of this study show that Configured Tunnel produced 0.2% higher throughput than 6to4 while using TCP as transport protocol. However, 6to4 produced 0.2% higher throughput than Configured Tunnel while using UDP. Delay is the most significant finding of this research. Configured Tunnel on Fedora 11 and 6to4 on Windows Server 2008 shows

the highest delay on both TCP and UDP protocols. Overall, the result shows that Configured Tunnel and 6to4 transition mechanisms show different performance.

Hypothesis 2: Different IP version 4 and IP version 6 transition mechanisms perform differently on various Operating Systems.

The findings show that Configured Tunnel on Fedora yields the highest performance in throughput when using TCP. However, 6to4 on Ubuntu 9.10 yields the highest performance in throughput when using UDP. However, Configured Tunnel on Fedora and 6to4 on Windows Server 2008 have the highest significant delay with approximately 2% different from each other. Overall, the result shows that Configured Tunnel and 6to4 perform differently on Windows Server 2003, Windows Server 2008, Ubuntu 9.10, and Fedora 11.

Hypothesis 3: There are factors that cause performance differences between IP version 4 and IP version 6 transition mechanisms.

By looking the analysis result, jitter affects throughput performance on both transition mechanisms. In addition, Figure 5-23 shows that, TCP produced higher jitter than UDP. This proves jitter values are different for various transmission protocols. In addition, the level of variance in jitter depends on the choice of traffic types (UDP or TCP) and transition mechanisms.

Appendices

Appendix A: TCP Throughput Results

Table below shows the result of TCP throughput:

TCP Throughput (Mbps)

Packet Size	Windows Server 2008		Windows Server 2003		Ubuntu 9.10		Fedora 11	
	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4
64	51.06	49.69	51.60	51.92	50.93	51.03	47.53	50.92
128	82.27	82.04	81.99	81.77	81.58	83.36	83.59	83.39
256	82.38	82.30	82.35	82.07	81.68	84.41	84.55	84.43
384	84.30	84.21	84.41	84.23	84.38	84.17	84.19	84.11
512	84.41	84.22	84.44	83.84	84.24	84.18	84.70	84.06
640	82.18	82.00	82.22	81.85	86.49	86.50	86.70	86.67
768	84.42	84.39	83.96	83.98	83.96	84.20	84.42	84.11
896	85.76	85.64	85.80	85.86	86.09	86.09	86.04	85.93
1024	87.17	86.89	87.26	86.26	87.06	87.32	87.09	87.13
1152	88.00	87.71	88.09	88.21	88.40	88.14	88.41	88.19
1280	82.08	81.81	81.98	82.11	89.10	89.19	88.67	89.18
1408	82.93	83.12	83.13	82.90	89.57	89.84	89.86	89.69
1536	84.07	83.86	83.91	84.11	84.21	84.00	84.23	84.25

Appendix B: TCP Jitter Results

Table below shows the result of TCP Jitter in second:

TCP Jitter (ms)

Packet Size	Windows Server 2008		Windows Server 2003		Ubuntu 9.10		Fedora 11	
	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4
64	0.86	1.08	0.94	0.80	1.33	1.42	0.74	1.15
128	0.28	0.35	0.33	0.25	0.35	0.30	0.30	0.27
256	0.37	0.35	0.40	0.42	0.38	0.42	0.39	0.36
384	0.50	0.45	0.50	0.53	0.45	0.44	0.53	0.37
512	0.58	0.55	0.51	0.61	0.38	0.38	0.48	0.50
640	0.58	0.57	0.60	0.57	0.41	0.41	0.39	0.38
768	0.59	0.57	0.67	0.62	0.55	0.64	0.60	0.63
896	0.66	0.72	0.59	0.72	0.34	0.34	0.40	0.35
1024	0.83	0.74	0.77	0.81	0.36	0.38	0.31	0.42
1152	0.79	0.80	0.72	0.91	0.36	0.37	0.35	0.45
1280	0.47	0.71	0.63	0.67	0.38	0.39	0.33	0.39
1408	0.62	0.64	0.59	0.57	0.46	0.47	0.43	0.40
1536	0.60	0.67	0.56	0.57	0.56	0.45	0.40	0.48

Appendix C: TCP Delay Results

Table below shows the result of TCP Delay in second:

TCP Delay (ms)

Packet Size	Windows Server 2008		Windows Server 2003		Ubuntu 9.10		Fedora	
	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4
64	180.86	1271.77	153.18	156.36	173.53	158.68	1246.63	300.72
128	39.31	1127.53	19.62	22.52	18.11	11.14	1100.46	158.78
256	45.43	1122.12	15.77	25.17	22.72	7.93	1101.21	159.52
384	41.82	1130.61	20.57	23.82	20.24	9.79	1101.60	162.73
512	46.11	1126.87	23.44	23.76	24.09	8.47	1083.91	165.08
640	45.96	1131.01	17.68	28.24	27.31	11.23	1090.99	162.45
768	47.12	1130.67	21.54	27.95	29.50	13.16	1090.34	162.69
896	49.61	1130.30	24.38	32.20	35.57	12.71	1089.80	166.01
1024	51.83	1133.27	21.64	35.85	41.19	13.67	1094.21	164.19
1152	49.47	1137.40	24.99	35.42	39.54	13.37	1094.62	169.69
1280	51.84	1139.94	24.62	35.03	40.66	16.85	1096.40	169.36
1408	54.80	1139.73	26.66	35.11	42.30	12.29	1097.91	169.30
1536	54.47	1142.34	28.26	38.37	42.68	15.60	1100.99	172.72

Appendix D: Router1 - TCP CPU Utilisation Results

Table below shows the result of CPU utilization of Router 1:

TCP CPU Utilization (Router 1) (%)

Packet Size	Windows Server 2008		Windows Server 2003		Ubuntu 9.10		Fedora	
	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4
64	14.84	22.66	14.06	17.19	18.00	20.00	18.00	14.00
128	24.22	28.13	23.44	21.88	14.00	15.00	10.00	13.00
256	28.13	37.50	17.97	21.88	17.00	12.00	9.00	14.00
384	27.34	21.88	16.41	19.53	16.00	14.00	11.00	14.00
512	26.56	31.25	19.53	17.97	16.00	16.00	10.00	12.00
640	31.13	25.00	17.97	21.88	14.00	16.00	12.00	16.00
768	24.23	27.34	20.31	16.41	15.00	13.00	10.00	13.00
896	21.00	32.03	17.19	16.41	16.00	12.00	12.00	15.00
1024	19.75	22.66	16.41	16.41	16.00	15.00	10.00	12.00
1152	30.36	25.78	12.50	14.06	17.00	16.00	13.00	14.00
1280	34.39	34.38	20.31	19.53	18.00	16.00	13.00	11.00
1408	27.64	28.13	17.19	17.97	13.00	12.00	12.00	13.00
1536	23.26	33.59	21.09	20.31	17.00	16.00	18.00	15.00

Appendix E: Router2 - TCP CPU Utilisation Results

Table below shows the result of CPU utilization of Router 2:

TCP CPU Utilization (Router 2) (%)

Packet Size	Windows Server 2008		Windows Server 2003		Ubuntu 9.10		Fedora	
	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4
64	14.06	15.63	10.94	10.16	12.00	18.00	18.00	13.00
128	21.88	19.53	12.50	17.19	16.00	18.00	19.00	12.00
256	20.31	23.44	15.63	14.84	10.00	11.00	14.00	13.00
384	17.19	21.09	15.63	13.28	17.00	13.00	12.00	12.00
512	28.13	19.53	17.19	12.50	18.00	12.00	13.00	17.00
640	25.78	22.66	16.41	15.63	19.00	17.00	10.00	18.00
768	17.97	21.09	12.50	12.50	16.00	10.00	18.00	17.00
896	16.41	17.19	10.16	13.28	17.00	12.00	11.00	18.00
1024	16.41	18.75	10.16	10.94	10.00	12.00	8.00	18.00
1152	18.75	15.63	9.38	8.59	17.00	9.00	24.00	15.00
1280	21.88	15.63	14.84	13.28	12.00	16.00	13.00	17.00
1408	17.19	26.56	13.28	12.50	11.00	11.00	12.00	13.00
1536	20.31	22.66	13.28	12.50	13.00	15.00	12.00	16.00

Appendix F: UDP Throughput Results

Table below shows the result of UDP throughput:

UDP Throughput (Mbps)

Packet Size	Windows Server 2008		Windows Server 2003		Ubuntu		Fedora	
	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4
64	33.24	36.05	34.69	35.70	35.34	33.78	24.07	33.92
128	50.49	53.13	52.01	35.74	52.60	52.30	24.07	52.49
256	67.32	68.85	68.82	68.72	68.66	68.74	68.91	68.76
384	74.61	76.30	76.38	76.31	76.24	76.37	76.37	76.14
512	78.96	80.68	80.68	80.55	80.59	80.57	80.61	80.34
640	82.15	83.48	83.42	83.57	83.63	83.33	83.22	83.20
768	84.11	85.15	85.14	85.71	85.48	85.52	85.66	85.10
896	85.70	86.86	86.92	86.81	87.13	86.96	87.03	86.89
1024	86.79	88.12	88.08	88.17	87.67	88.08	88.20	87.77
1152	87.95	88.69	88.53	88.81	89.19	89.20	89.02	88.27
1280	81.70	82.86	83.06	83.19	89.60	89.62	89.47	89.09
1408	82.03	83.35	83.93	83.85	90.35	90.44	90.33	89.69
1536	83.19	84.58	84.80	84.69	84.11	84.30	83.60	83.79

Appendix G: UDP Jitter Results

Table below shows the result of Jitter in second:

UDP Jitter (ms)

Packet Size	Windows Server 2008		Windows Server 2003		Ubuntu		Fedora	
	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4
64	0.13	0.15	0.20	0.18	0.23	0.23	0.66	0.28
128	0.24	0.24	0.19	0.13	0.23	0.28	0.20	0.25
256	0.26	0.26	0.27	0.25	0.24	0.21	0.24	0.27
384	0.21	0.29	0.17	0.21	0.23	0.10	0.16	0.26
512	0.27	0.29	0.24	0.26	0.24	0.21	0.14	0.22
640	0.29	0.28	0.21	0.22	0.28	0.16	0.32	0.28
768	0.35	0.31	0.23	0.23	0.29	0.25	0.27	0.29
896	0.30	0.19	0.27	0.33	0.21	0.27	0.27	0.29
1024	0.36	0.26	0.26	0.31	0.25	0.31	0.32	0.25
1152	0.65	0.64	0.78	0.65	0.41	0.33	0.64	0.44
1280	0.79	0.57	0.74	0.70	0.72	0.61	0.68	0.44
1408	0.75	0.70	0.69	0.79	0.78	0.67	0.80	0.83
1536	0.84	0.83	0.76	0.73	0.81	0.77	0.86	0.86

Appendix H: UDP Delay Results

Table below shows the result of Delay in second:

UDP Delay (ms)

Packet Size	Windows Server 2008		Windows Server 2003		Ubuntu		Fedora	
	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4
64	54.22	1103.32	566.63	566.20	71.26	23.45	1131.05	206.94
128	41.31	1102.58	487.90	560.06	71.70	22.58	1122.84	202.40
256	39.26	1101.14	302.85	311.79	82.09	32.95	1133.18	213.99
384	34.67	1105.47	220.99	229.76	90.06	43.77	1146.85	228.54
512	43.83	1106.39	182.06	189.71	97.80	50.23	1157.68	250.66
640	47.21	1106.94	136.40	142.75	108.00	59.85	1165.29	258.75
768	47.59	1111.54	77.59	115.97	110.70	71.04	1173.28	263.36
896	51.02	1113.78	111.76	99.93	126.71	74.66	1179.04	269.61
1024	45.87	1112.94	93.07	90.12	96.10	74.40	1182.20	270.34
1152	42.13	1108.67	81.37	80.67	98.66	73.72	1176.30	268.86
1280	49.49	1108.31	142.91	132.42	91.12	64.83	1177.57	267.12
1408	49.06	1108.76	124.56	120.01	88.74	75.11	1176.31	263.86
1536	54.92	1114.79	112.97	110.79	92.34	64.57	1171.19	264.70

Appendix I: Router1 - UDP CPU Utilisation Results

Table below shows the result of CPU Utilization of Router 1:

UDP CPU Utilization (Router 1) (%)

Packet Size	Windows Server 2008		Windows Server 2003		Ubuntu		Fedora	
	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4
64	42.97	57.03	48.44	7.81	10.00	12.00	13.00	15.00
128	34.38	42.19	42.97	8.59	18.00	18.00	12.00	13.00
256	30.47	33.59	32.03	10.16	16.00	16.00	10.00	12.00
384	20.31	28.13	25.78	13.28	17.00	18.00	15.00	12.00
512	23.44	29.69	20.31	12.50	12.00	14.00	13.00	10.00
640	18.75	25.78	19.53	13.28	17.00	17.00	13.00	11.00
768	15.63	19.53	17.97	14.06	17.00	18.00	10.00	12.00
896	15.63	21.88	13.28	13.28	15.00	14.00	14.00	13.00
1024	14.06	15.63	11.72	11.72	16.00	16.00	13.00	11.00
1152	21.09	21.88	13.28	13.28	14.00	13.00	12.00	13.00
1280	15.63	20.31	16.41	13.28	19.00	19.00	10.00	10.00
1408	22.66	22.66	21.88	23.44	17.00	17.00	13.00	12.00
1536	18.75	21.09	14.84	14.06	20.00	20.00	13.00	14.00

Appendix J: Router2 - UDP CPU Utilisation Results

Table below shows the result of CPU Utilization of Router 2:

UDP CPU Utilization (Router 2) (%)

Packet Size	Windows Server 2008		Windows Server 2003		Ubuntu		Fedora	
	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4	Configured Tunnel	6to4
64	42.97	44.53	35.16	3.91	12.00	16.00	8.00	13.00
128	35.16	40.63	25.00	7.03	14.00	15.00	15.00	13.00
256	25.00	26.56	19.53	7.81	12.00	12.00	10.00	16.00
384	17.97	22.66	13.28	5.47	14.00	16.00	22.00	14.00
512	16.41	21.09	10.94	6.25	12.00	12.00	12.00	11.00
640	15.63	22.66	9.38	8.59	13.00	13.00	15.00	10.00
768	17.19	16.41	7.81	7.81	17.00	17.00	16.00	14.00
896	10.94	20.31	7.03	8.59	14.00	18.00	14.00	15.00
1024	11.72	10.94	7.03	5.47	12.00	16.00	14.00	14.00
1152	8.59	14.06	3.91	7.03	9.00	12.00	16.00	13.00
1280	17.97	13.28	10.94	9.38	18.00	17.00	15.00	15.00
1408	11.72	12.50	9.38	10.94	19.00	14.00	13.00	10.00
1536	13.28	17.97	8.59	9.38	15.00	15.00	12.00	16.00

References

- AlJa'afreh, R., Mellor, J., & Awan, I. (2008). Evaluating BDMS and DSTM Transition Mechanisms. *Proceedings of the Second UKSIM European Symposium on Computer Modelling and Simulation*(pp. 488-493). Washington: IEEE Computer Society.
- Avallone, S., Guadagno, S., Emma, D., Pescap, A., & Ventre, G. (2004). D-ITG Distributed Internet Traffic Generator. *Proceedings of the First International Conference on the Quantitative Evaluation of Systems* (pp. 316-317). Washington: IEEE Computer Society.
- Atwood, J. W., Das, K. C., & Haddad, I (2010). *NAT-PT: Providing IPv4/IPv6 and IPv6/IPv4 address translation*. Retrieved February 12, 2010, from http://profile.iiita.ac.in/mlalavat_b03/work/v4_v6_translation.pdf
- Blanchet, M. & Parent, F. (2000). *IPv6 transition mechanisms*. Retrieved September 12, 2009, from http://www.viagenie.qc.ca/en/ipv6/presentations/IPv6-transition-mechanisms_v1.pdf.
- Blum, R. (2003). *Network performance open source toolkit: Using Netperf, tcptrace, NIST Net, and SSFNet*. Indianapolis: Wiley.
- Chen, J., Chang, Y., & Lin, C. (2004). Performance investigation of IPv4/IPv6 transition mechanisms. *Proceedings of the 6th International Conference on Advanced Communication Technology* (pp. 545-550). Washington: IEEE Computer Society.
- Cisco (2007-2009). *Classful IP addressing*. Retrieved May 20, 2009, from http://curriculum.netacad.net/virtuoso/servlet/org.cli.delivery.rendering.servlet.CCServlet/SESSION_ID=1243417553459332,LMS_ID=CNAMS,Theme=ccna3theme,Style=ccna3,Language=en,Version=1,RootID=knet-lcms_exploration2_en_40,Engine=static/CHAPID=null/RLOID=null/RIOID=null/theme/ch eetah.html?cid=0900000000&l1=en&l2=none&chapter=6
- Cisco Systems (2001). *Dictionary of internetworking terms and acronyms*. Indianapolis: Cisco Press.
- Davies, J. (2008a). *Understanding IPv6* (2nd ed.). Washington: Microsoft Press.
- Davies, J. (2008b). 6to4. In M. Delre, K. Szall, and M. Gargiulo (Eds.), *Understanding IPv6*. (pp.295-316). Washington: Microsoft Press.
- Deveriya, A (2006). *Network administrator survival guide*. Indianapolis: Cisco Press.
- Govil, J. (2007). On the investigation of transactional and interoperability issues between IPv4 and IPv6. *Proceedings of the 2007 IEEE International Conference on Electro/Information Technology*(pp. 604-609). Washington: IEEE Computer Society.

- Govil, J., Govil, J., Kaur, N., & Kaur, H. (2008). An examination of IPv4 and IPv6 Networks: Constraints and Various Transition Mechanisms. *Proceedings of the 2008 IEEE Southeastcon* (pp. 178-185). Washington: IEEE Computer Society.
- Green, D., Fiuczynski, M. E., & Marc, E. (2006). IPv6 translation for IPv4 embedded systems. *Proceedings of the 2006 IEEE Sarnoff Symposium*. Washington: IEEE Computer Society.
- Grosse, E. & Lakshman, Y. N. (2003). Network processors applied to IPv4/IPv6 transition. *IEEE Network*, 17(14), 35–39.
- Hiroimi, R. & Yoshifuji, H. (2006). Problems on IPv4-IPv6 network transition. *Proceedings of the International Symposium on Applications and Internet Workshops* (pp. 38-42). Washington: IEEE Computer Society.
- Hong, S., Ko, N., & Kim, N. (2006). New IPv6 transition mechanism based on end-to-end tunnel. *Proceedings of the Joint International Conference on Optical Internet and Next Generation Network* (pp. 168-170). Washington: IEEE Computer Society.
- Internet Systems Consortium (2001-2009). *Number of Internet hosts*. Retrieved May 17, 2009, from <https://www.isc.org/solutions/survey/history>.
- Jones, R. (2006). *Welcome to the Netperf homepage*. Retrieved May 10, 2009, from <http://www.netperf.org/netperf>.
- Lee, J., Shin, M., & Kim H. (2004). Implementation of NAT-PT/SIIT, ALGs and consideration to the mobility support in NAT-PT environment. *Proceedings of the IEEE 59th Vehicular Technology Conference* (pp. 2714-2718). Washington: IEEE Computer Society.
- Mark, A. & Miller, P.E. (2000a). Transition Strategies. In L. Lewin, E. Wollensky, J. Thompson, A. S. Devlin, and J. M. Smith (Eds.), *Implementing IPv6*. 2nd Edition . (pp.285-308). California: M&T Books.
- Openmaniak.com (2009). *iPerf*. Retrieved January 4, 2010, from <http://openmaniak.com/iperf.php>.
- Raicu, I. & Zeadally, S. (2003). Evaluating IPv4 to IPv6 transition mechanisms. *Proceedings of the 10th International Conference on Telecommunications* (pp. 1091-1098). Washington: IEEE Computer Society.
- RFC2373 (1998). *IP version 6 addressing architecture*. Retrieved May 17, 2009, from www.ietf.org/rfc/rfc2373.txt?number=2373.
- RFC2893 (2000). *Transition Mechanisms for IPv6 hosts and routers*. Retrieved May 17, 2009, from www.ietf.org/rfc/rfc2893.txt.
- RFC2766 (2000). *Network Address Translation – Protocol Translation (NAT-PT)*. Retrieved December 12, 2009, from www.ietf.org/rfc/rfc2766.txt.

- RFC3056 (2001). *Connection of IPv6 domain via IPv4 cloud*. Retrieved July 15, 2009, from <http://www.ietf.org/rfc/rfc3056.txt>.
- Sailan, M. K., Hassan, R., and Patel, A. (2009). A comparative review of IPv4 and IPv6 for research test bed. *Proceedings of the International conference on Electrical Engineering and Informatics*, (pp.427-433). Washington: IEEE Computer Society.
- Sanguankotchakorn, T. & Somrobru, M. (2005). Performance evaluation of IPv6/IPv4 deployment over dedicated data links. *Proceedings of the 5th International Conference on Information Communications and Signal Processing*, (pp. 244-248). Washington: IEEE Computer Society.
- Schroder, C. (2007). *Measure network performance: iperf and ntop*. Retried January 4, 2010, from http://www.enterprisenetworkingplanet.com/netos/article.php/10951_3658331_1
- SourceForge. (2009). *Iperf*. Retrieved May 5, 2009, from <http://iperf.sourceforge.net>.
- Thomas R. M. (2003). *Blending qualitative and quantitative research methods in theses and dissertations*. California: Corwin Press.
- Visoottiviseth, V. & Bureenuk, N (2008). Performance comparison of ISATAP implementations on FreeBSD, RedHat, and Windows 2003. *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications – Workshops* (pp. 547-552). Washington: IEEE Computer Society.
- Waddington, D. & Chang, F. (2002). Realising the transition to IPv6. *IEEE Computer Magazine*, 40(2), 138-147.
- Wang, Y., Ye, S., & Li, X. (2005). Understanding current IPv6 performance: A measurement study. *Proceedings of the 10th IEEE Symposium on Computers and Communication(71-76)*.Washington: IEEE Computer Society.