ISSN 1512-1232

PERFORMANCE EVALUATION OF LINUX AND MICROSOFT WINDOWS OPERATING SYSTEMS ON CONFIGURED TUNNELS OF IPV6 AND IPV4

[1]OGUNLEYE G.O, [2]Bankole B.

[1]Department of Computer Science, Federal University, Oye-Ekiti, Ekiti State, Nigeria
[2]Department of Computer Science, Redeemer's University, Ede, Osun State, Nigeria

Abstract

The paper examined the background evolution of IPv4 and IPv6 and various research works that have done in the area.

The study evaluates the performance of IPv4 and IPv6 on Microsoft operating system and Linux operating system using two transition mechanisms. The performance measurement is examined on two types of transmission protocols namely: TCP and also UDP. The performance metrics used in the study are throughput, delay, jitter, CPU utilization and they were used over a range of bytes. The result confirmed that Linux performed effectively than its windows counterpart on configured tunnels of IPv4 and IPv6.

Keywords: *lPV4, IPV6, TCP, IP,Windows, LINUX*

1.0 Introduction

In the last few decades, internet has grown extensively as a result of its worldwide use and accessibility which has resulted into 1.5 trillion dollars worldwide economic benefits annually (Atkinson et al, 2010). Consequently, there is a need to examine the various communications that make up the internet so as to further enhance its use. A number of previous research works had been done on the use of Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6) over different operating systems (Kolahi and Soorty, 2011); while some over cable networks (Soorty and Sarkar, 2013).

According to Ogunde (2015), IP makes use of other supporting protocol like the:

- ARP (Address Resolution Protocol): this protocol is used to associate logical address with a physical address.
- RARP (Reverse Address Resolution Protocol): this is a protocol that allows a host to know its internet address when the physical address is known.
- ICMP (Internet Control Message Protocol): this is a protocol that allows the hosts and gateways to send messages back to the sender on datagram problems encountered.
- IGMP (Internet Group Message Protocol): this is a protocol that is used to send group of messages simultaneously to a group of recipients.

IPv6 is also called the next generation Internet Protocol. It was designed at first to provide solution to IPv4's inevitable and impending address exhaustion crisis (Lammle, 2013).

IPv 6 is arguably the IP protocol of choice, but transition from IPv4 internet to IPv6 promises to be a long process as they are two completely separate protocols and it is impossible to switch the entire internet to IPv6 at once. IPv4 host and routers will not be able to deal directly with IPv6 traffic and vice-versa, as IPv6 is not backward compatible with IPv4. Therefore, IPv4 and IPv6 coexist for a long time. Thus, this paper work looks into the transition performance evaluation and inter-operation mechanisms of IPv4 and IPv6 on two operating systems.

ISSN 1512-1232

To a great extent the internet primarily runs on IPv4, almost all communications involving the internet depend on this protocol. The address structure is very important in understanding this protocol. IPv4 addressing consists of four bytes with each byte containing eight bits. The whole address space of IPv4 consists of 32bits i.e. 2^32 (4,294,967,296) distinct addresses. According to Lammle (2013), IP version 4 consists of five groups of addresses which is shown table 1:

Table 1: Groups of IPv4 addresses

Class	Start	End	Comments
A	0.0.0.0	127.255.255.255	Address 0 is the default address while 127 has being reserved for Loopback (localhost).
B	128.0.0.0	191.255.255.255	169.0.0.0 to 169.255.255.255 reserved as APIPA addresses
C	192.0.0.0	223.255.255.255	
D	224.0.0.0	239.255.255.255	Multicast Group.
E	240.0.0.0	255.255.255.255	Reserved For Research/Testing

According to Grosse and Lakshman (2003), as the year goes by, there is frequent increase in the number of hosts (devices) on the internet as a result of technological growth in countries with very large populations who are gaining access to the internet which causes shortage in IPv4 address space. This is the main reason why IPv6 is being developed and implemented and it is referred to as the next generation Internet Protocol. It provides solution to the issue of address space and also comes with better features.

This is known as latest version of the Internet Protocol that was developed to provide solutions to IPv4 deficiencies. According to Green et al (2006), the development of IPv6 was aimed at incorporating all the improvements and best features that were developed from the inception of IPv4 (more than 20years) into a next–generation protocol to aid the rapid growth of internet applications and communications. IPv6 is not just going to provide solutions to the address space issues encountered in IPv4; it will also provide better features that can ensure a better performance than IPv4. This newer version of Internet Protocol supports unicast, anycast and multicast addresses. The addressing format of IPv4 and IPv6 varies, that of IPv4 is represented by dotted decimal while that of IPv6 is in hexadecimal notation (Govil et al, 2007).

Network Address Translation (NAT) and Classless Inter-Domain Routing (CIDR) were used to temporarily resolve address space limitation in IPv4 before the introduction of IPv6, and they accept that hosts within a Local Area Network (LAN) and on other LANs around the world can still use similar private IP address as hosts (Sotharith, 2010).

IPv6 has a better forwarding mechanism than IPv4 as a result of the 40 bytes fixed header that enables routers forward IPv6 packets faster (Davies, 2008).

2.0 Related Works

This section discusses the previous research works that have been done.

2.1 TRANSITION MECHANISMS

Sotharith (2010) stated that IPv6 was designed in such a way that makes it backward incompatible with IPv4 i.e. IPv4 hosts can transfer IPv4 packets to other IPv4 hosts and IPv6 hosts can also send IPv6 packets to other IPv6 hosts. The ability for both protocols to be able to operate when they coexist on the internet is an issue. In order to solve this issue, IPv4/IPv6 transition mechanisms was designed by IETF (Internet Engineering Task Force) NGtrans to enable transition

ISSN 1512-1232

period to progress without any problem. These mechanisms permit both IPv4 and IPv6 to exist together on the internet which can last for many years.

Some of these mechanisms include:

- DUAL STACK: it is a type of transition mechanism that is straightforward and simple to configure. In this mechanism, both IPv4 and IPv6 are enabled on a single network interface card. An IPv6/IPv4 node is a device enabled with both IPv6 and IPv4. IPv6/IPv4 nodes can operate with both IPv4 and IPv6 nodes using IPv4 and IPv6 packets respectively. Dual stack mechanism requires both IP version 4 and version 6 addresses to be assigned manually and/or automatically, using DHCP (Dynamic Host Configuration Protocol) to communicate with both IPv4 and IPv6 nodes. The diagram in figure 1 shows the architecture of Dual IP stack (Lewin *et al*, 2000):

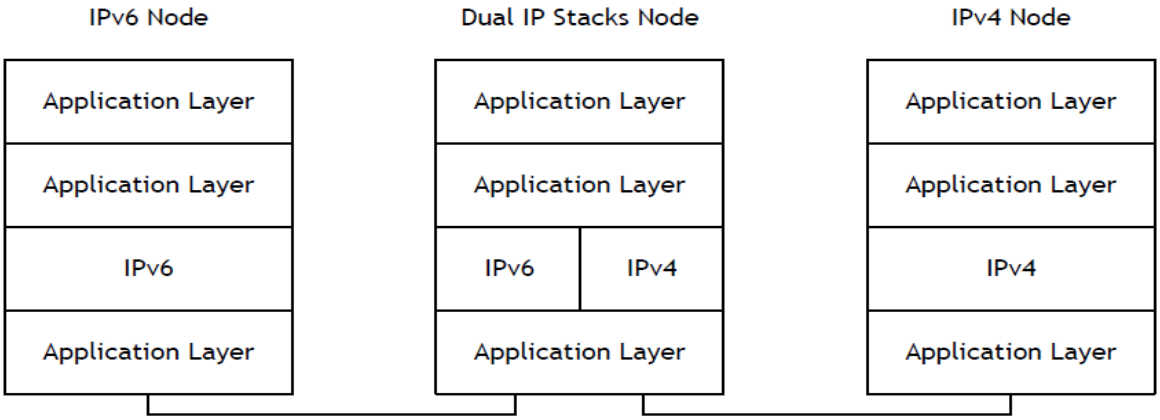

Figure1: Dual IP Stack (source: Lewin *et al*, 2000)

CONFIGURED TUNNEL: This is a type of mechanism that allows more than one IPv6 network to interact across IPv4 routing infrastructure through a medium (tunnel). The entry point of the tunnel is manually configured. IPv6 addresses are encapsulated with IPv4 packets in order to ensure communication between IPv6 nodes on IPv4 network. The router at the endpoint of each tunnel consists of two network interface cards with IPv6 addresses configured on the internal network interface card and IPv4 addresses configured on the external network interface card. This is shown in figure 2.

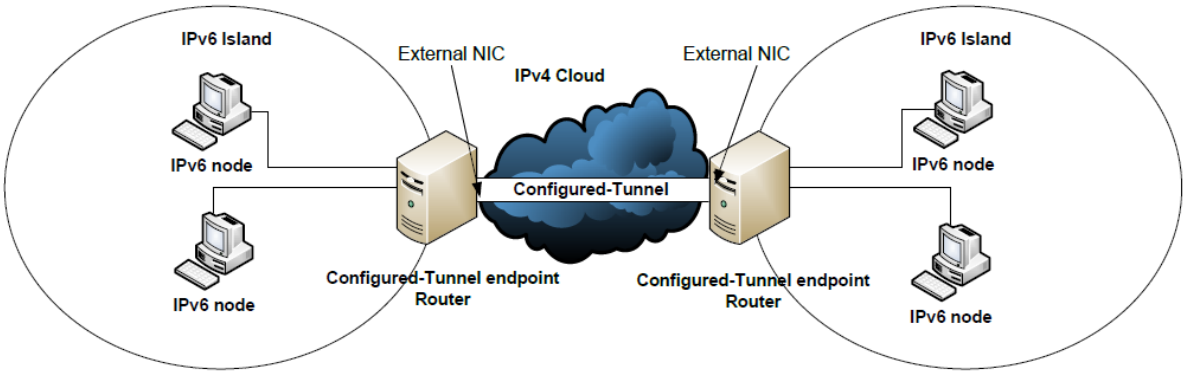

Figure 2: infrastructure of a configured tunnel network (source: Sotharith, 2010).

ISSN 1512-1232

- **6TO4 TUNNELLING MECHANISM**
  This is a tunneling technology used to assign/allocate blocks of IPv6 addresses to IPv4 hosts, enabling router- router, host- router, router- host IPv6 unicast connectivity between IPv6 sites and hosts across the IPv4 internet. The architecture of 6to4 address is shown in figure 3:

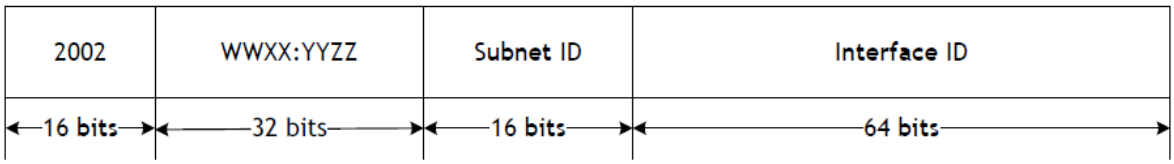

Figure3: Architecture of 6to4 address (source: Davies, 2008)

On the arrival of IPv6 packets at 6to4 router, the process of encapsulation is triggered by keeping IPv6 packet in IPv4 packet so that it can transmit across IPv4 internet infrastructure. The Source and destination of IPv4 addresses are specified with IPv4 header and the body of IPv4 packet contain IPv6 header and payload as stated in RFC3056 T (2001).6to4 packet is travelling across 6to4 tunneling established by 6to4 routers. As the encapsulated packet arrives at the destination tunneling end-point, 6to4 router performs de-capsulation process by removing IPv4 header and forward IPv6 packet through to IPv6 node. Various Components perform different functions in 6to4 tunneling mechanism. These components are:

  - 6to4 host: this is the client computer and it cannot carry out 6to4 tunneling over IPv4 internet.
  - 6to4 router: it can carry out 6to4 tunneling over the internet and also transferring 6to4 packet from 6to4 host in a site to another 6to4 host in another site across the internet.
  - 6to4 host/router: it can carry out tunneling with 6to4 host/routers, routers, and relays, but it does not have the ability to forward packet.
- **IPV4 OVER IPV6 TUNNELS:** IPv6 network environment supports IPv4 over IPv6 tunnels, which includes:
  - Generic Routing Encapsulation (GRE) IPv4 tunnel for IPv6: The traffic experienced in IPv6 can be brought through the standard GRE tunneling technique (which is a Cisco proprietary tunneling protocol and was created to give the administrations to actualize any standard point-to-point encapsulation scheme) over the IPv4 GRE tunnels. The tunnels convey IPv6 as the traveler protocol with the GRE as the bearer protocol and IPv4 or IPv6 as the transport protocol. These tunnels are connections between two points, and every connection has a different tunnel.
- **GRE** support over IPv6 transport: GRE has a protocol field that distinguishes the traveler protocol. The tunnels permit IPv6 to be made a traveler protocol, which permits IPv6 traffic to keep running over the same tunnel. In the event that GRE did not have a protocol field, it is difficult to know whether the tunnel was conveying IPv6 packets. The GRE protocol field makes it attractive to tunnel IPv6 inside GRE.
- **NETWORK ADDRESS TRANSLATION-PROTOCOL TRANSLATION (NAT-PT):** this is a translation mechanism that functions as translator between IPv4 and IPv6 packets. The main function is to IPv6 addresses to IPv4 addresses and also IPv4 addresses to IPv6

ISSN 1512-1232

addresses. It allows IPv4 and IPv6 to exist on the same network together, when implemented; it allows IPv4 network and IPv6 network to be able to communicate with one another using a single NAT-PT server. The hosts of each IPs do not need to have dual stack mode enabled but their networks must have its own DNS server. The diagram in figure 4 shows the implementation of NAT-PT network infrastructure:

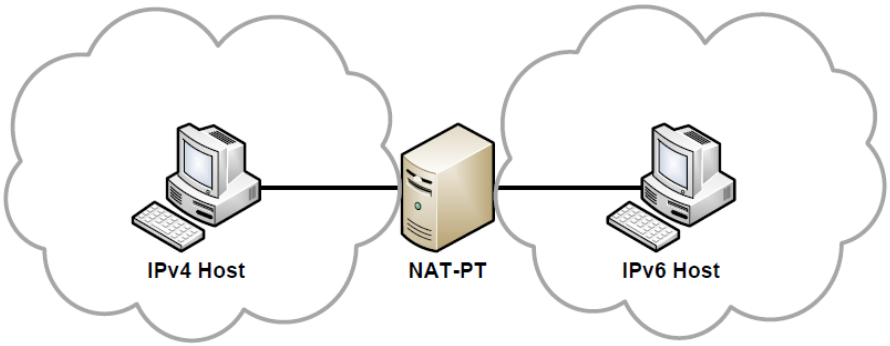

Figure 4: infrastructure of NAT-PT network (source: Sotharith, 2010)

NAT-PT functions primarily to convert IPv4 packet to IPv6 packet. According to Lee et al (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 goes about as an interpreter server that stores IPv4 worldwide routable address pool. This pool is utilized to consequently dispense addresses to the IPv6 node before the interpretation process begins crosswise over NAT-PT node. When the interpretation process closes, IPv4 address task will end (Atwood et al, 2010). NAT-PT is the IPv4 and IPv6 packet interpreter that sits in the middle of IPv4 and IPv6 system and it makes an interpretation of IPv4 packets to IPv6 packets and IPv6 packets to IPv4 packet.

2.2. PERFORMANCE MEASUREMENT TOOLS

There are different techniques that can be used to test performance of transition mechanism. These techniques include: throughput, delay, CPU utilization and jitter.

1. Throughput: this is a very common technique that is used in evaluating the performance of a network. It makes the amount of data transferred between two network hosts easier to know. 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 elements that can influence throughput execution, for example, the restriction of equipment handling power and system blockage or bottleneck because of the configuration of system topology. Megabits every second (Mbps) is the unit utilizes as a part of throughput estimation.
2. Delay: it is the measure of time it takes a packet to navigate from source to destination". Before measuring delay in an network, time synchronization in the middle of sender and recipient is essential. Ensure that sender and recipient hubs have the very same time settings.
3. CPU Utilization: is the rate measure of PC's CPU asset taken amid the handling of an application or an assignment. Higher rate of CPU Utilization demonstrates that the PC

ISSN 1512-1232

ideally utilized CPU assets. This is not the situation, which ought to dependably be worried about on the grounds that if a PC indicates higher CPU use however it produces higher throughput, implies the PC utilizes most ideal assets as a part of request to deliver best throughput result. In any case, if the PC produces lower throughput, which is a state of concern, that equipment is not proficiently utilizing the CPU assets. In this examination concentrate, all equipment requires to have indistinguishable detail so as to give consistency between every hub.

4. Jitter: is the between bundle delay fluctuation; that is, the contrast between packet landing and takeoff. Jitter is an imperative Quality of Service metric for voice and video applications." when various packets send over the system, the distinctions in time that every parcel touching base at the destination is known as jitter.

This metrics are however utilized in this study to evaluate the performance of ipv4 and ipv6 on windows and linux operating systems.

3.0 Materials and Methods

The experimental design, implementation and measurements used were conducted at the Computer Laboratory of Redeemer's University, which was obtained by using a simple performance measuring tool (D-ITG).

ALGORITHM FOR THROUGHPUT

//algorithm for D-ITG tool
//input: ITG commands (send,decode)
//output: throughput, delay, jitter, cpu utilization
CMD ⟵ Commandprompt
DIR ⟵ directory
{
get CMD1
set DIR= Tool_DIR;
Input(ITGRecev_command)
{
get CMD2
DIR=TOOL_DIR;
Input(ITGSend_command);
{
Get CMD3
Set DIR=Tool_DIR;
Input (itgdec _command)
Output (result)
}
}
}

The algorithm above explains the procedure involved in using the D-ITG performance tool. The command prompt is opened and changed to the directory of where the D-ITG tool is saved, then *ITGRecv:is entered in order to be able to receive the packets on the client 2 system.*

Open command prompt change the directory to the directory of where the D-ITG tool is saved, then enter ITGSend –T *transmission protocol* –a *ip address* –c 100 –C *packet size* –t 15000\ -Lsender.log –x receiver.log. Open command prompt change the directory to the directory of where

ISSN 1512-1232

the D-ITG tool is saved, then enter ITGDec receiver.log. This enables you to view the results from client 1's system.

3.1 CONFIGURED TUNNEL MECHANISM ON WINDOWS SERVER 2008

This section shows the experimental setup and configuration of configured tunnel on Windows Server 2008. The batch script used is shown below:

ROUTER 1

- *Create tunnel name boluwaji*

#netshint ipv6 add v6v4tunnel "boluwaji" 10.1.1.1 10.1.1.2

- *Add ipv6 address to tunnel interface*

#netshint ipv6 add address "boluwaji" 2001:200:20:2::1

- *Configure static routing*

#netshint ipv6 add route ::/0 "boluwaji" 2001:200:20:2::2

- *Set packet forwarding*

#netshint ipv6 set int "boluwaji" forwarding=enable

ROUTER 2

- *Create tunnel name boluwaji*

#netshint ipv6 add v6v4tunnel "boluwaji" 10.1.1.2 10.1.1.1

- *Add ipv6 address to tunnel interface*

#netshint ipv6 add address "boluwaji" 2001:200:20:2::2

- *Configure static routing*

#netshint ipv6 add route ::/0 "boluwaji" 2001:200:20:2::1

- *Set packet forwarding*

#netshint ipv6 set int "boluwaji" forwarding=enable

3.2 6TO4 MECHANISM ON UBUNTU

This section shows the experimental setup and configuration of 6to4 mechanism on Ubuntu. The batch script used is shown below:

Router 1:

- *Set IPv6 packet forwarding*

#sysctl –w net. Ipv6.conf.default.forwarding=1

- *Add 6to4 tunnel endpoints*

#ip tunnel add tun6to4 mode sit remote any local 10.1.1.1
#ip link set dev tun6to4 mtu 1472

- *Add IPv6 to tunnel*

#ip-6 addr add dev tun6to4 2001:200:20:2::1/64

- *Configure static route*

#ip-6 route add 2001::/16 dev tun6to4
#ip-6 route add::/0 via 2001:200:20:2::2 dev tun6to4 metric 1

Router 2:

- *Set IPv6 packet forwarding*

ISSN 1512-1232

#sysctl –w net. Ipv6.conf.default.forwarding=1

- *Add 6to4 tunnel endpoints*

#ip tunnel add tun6to4 mode sit remote any local 10.1.1.2
#ip link set dev tun6to4 mtu 1472

- *Add IPv6 to tunnel*

#ip-6 addr add dev tun6to4 2001:200:20:2::2/64

- *Configure static route*

#ip-6 route add 2001::/16 dev tun6to4
#ip-6 route add::/0 via 2001:200:20:2::1 dev tun6to4 metric 1

NETWORK DESIGN

Four computers where used in conducting this experiment and this form the network infrastructure. It contains two client nodes and two router nodes. The simulation involves two IPv6 LANs interconnected through a simulation of the internet (IPv4 network infrastructure) is contained in the infrastructure.

This infrastructure contains a combination of Fast Ethernet and Gigabit network cards on the computers used. This experimental design contains the use ofIPv4, IPv6 and IPv4/IPv6transition mechanisms. Client 1 was configured as the D-ITG packet sender and Client 2 as the D-ITG packet receiver.

GRAPHICAL USER INTERFACE

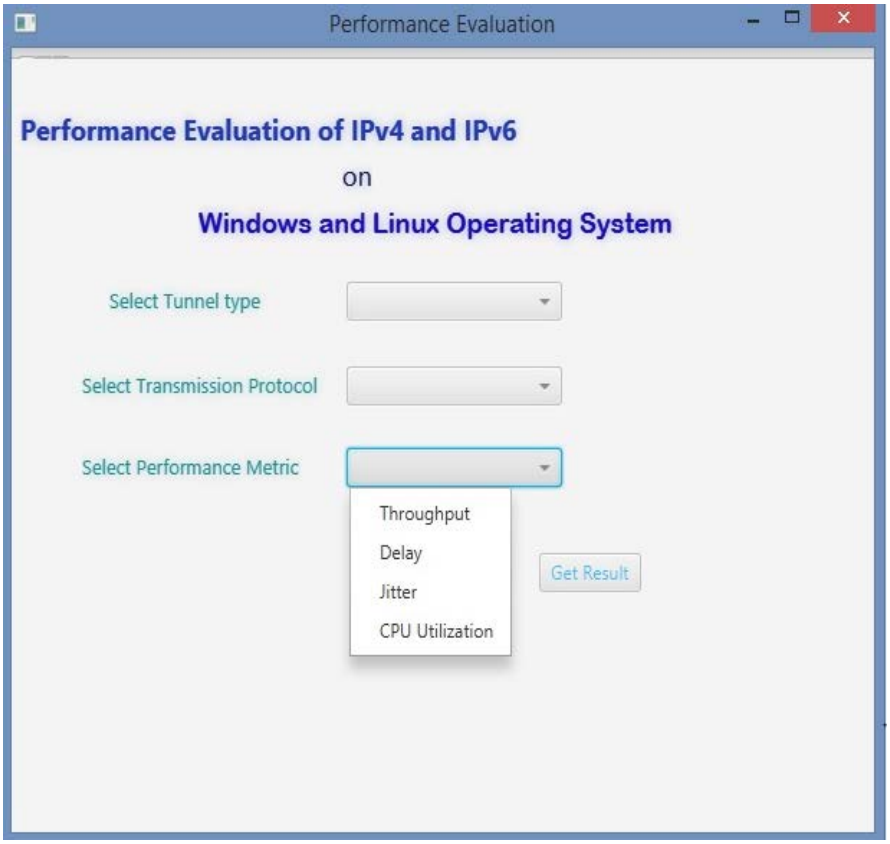

Figure 5: Graphical User Interface to display result

ISSN 1512-1232

From this GUI, performance metric could be selected to view its table and graph.

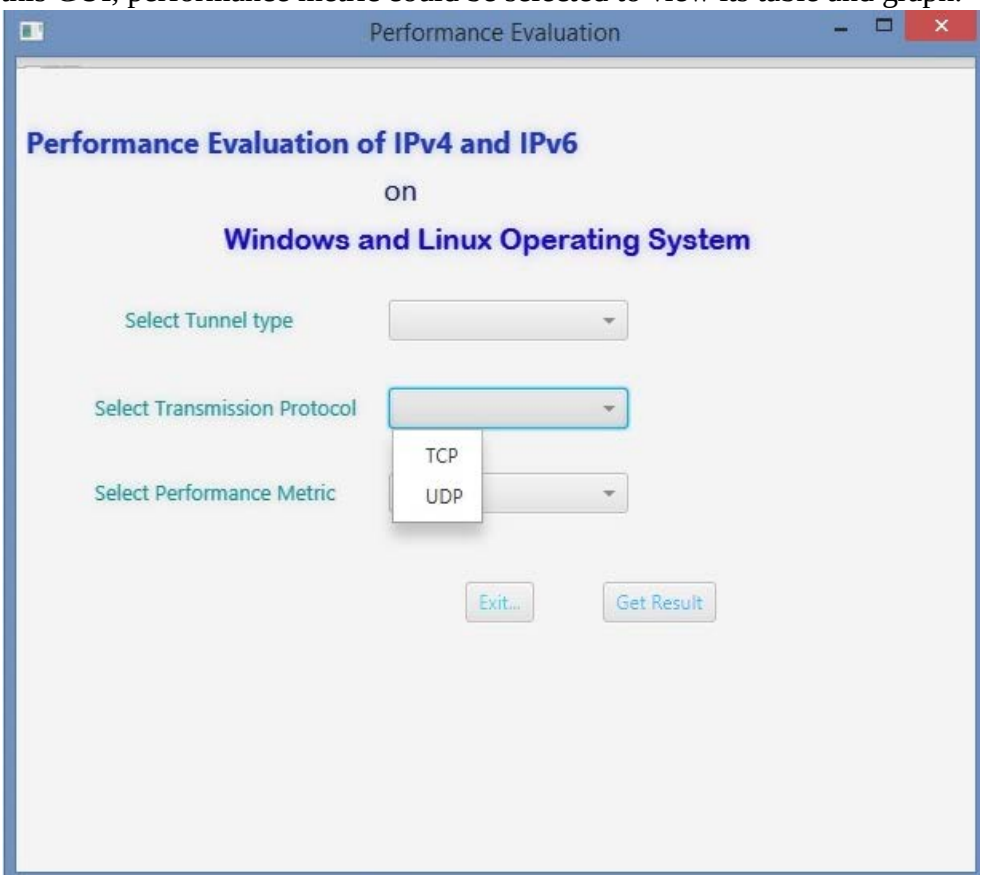

Figure 6: Graphical User Interface to display result

From this GUI, it is easier to select the particular transmission protocol you want to view its table and graph.

ISSN 1512-1232

DATA ANALYSIS

The results obtained from the experiment conducted in the laboratory using configured tunnel and 6to4 mechanisms, TCP and UDP as transmission protocols, and throughput, jitter, delay and CPU utilization are as follows:

CONFIGURED TUNNEL (TCP THROUGHPUT(Mbps))		
PACKET SIZE	WINDOWS SERVER 2008	UBUNTU
64	51.06	50.93
128	82.27	81.58
256	82.38	81.68
384	84.30	84.38
512	84.41	84.24
640	82.21	86.49
768	84.42	83.96
896	85.76	86.09
1024	87.17	87.06
1152	88.00	88.40
1280	82.08	89.19
1408	82.93	89.84
1536	84.07	84.00

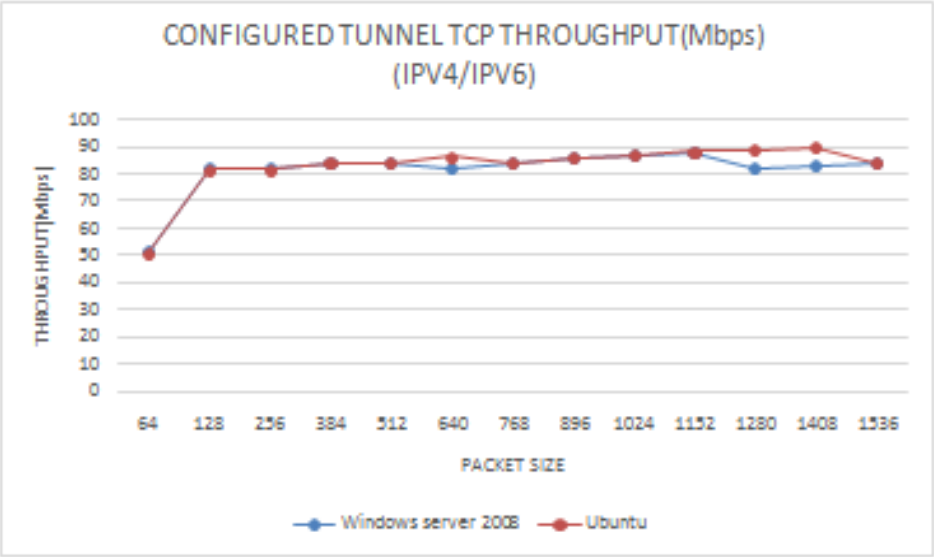

Figure 7: CONFIGURED TUNNEL TCP THROUGH PUT

Fig 7 shows the result obtained from the experiment performed in the lab. From the figure above, it is observed that at time 640, 1280 and 1408 windows server 2008 had lesser amount of throughput time when compared with Ubuntu but for all other packet sizes they had almost the same amount of throughput time.

ISSN 1512-1232

CONFIGURED TUNNEL (TCP JITTER (ms))		
PACKET SIZE	WINDOWS SERVER 2008	UBUNTU
64	0.86	1.33
128	0.28	0.35
256	0.37	0.38
384	0.50	0.45
512	0.58	0.38
640	0.58	0.41
768	0.59	0.55
896	0.66	0.34
1024	0.83	0.36
1152	0.79	0.36
1280	0.47	0.38
1408	0.62	0.46
1536	0.60	0.56

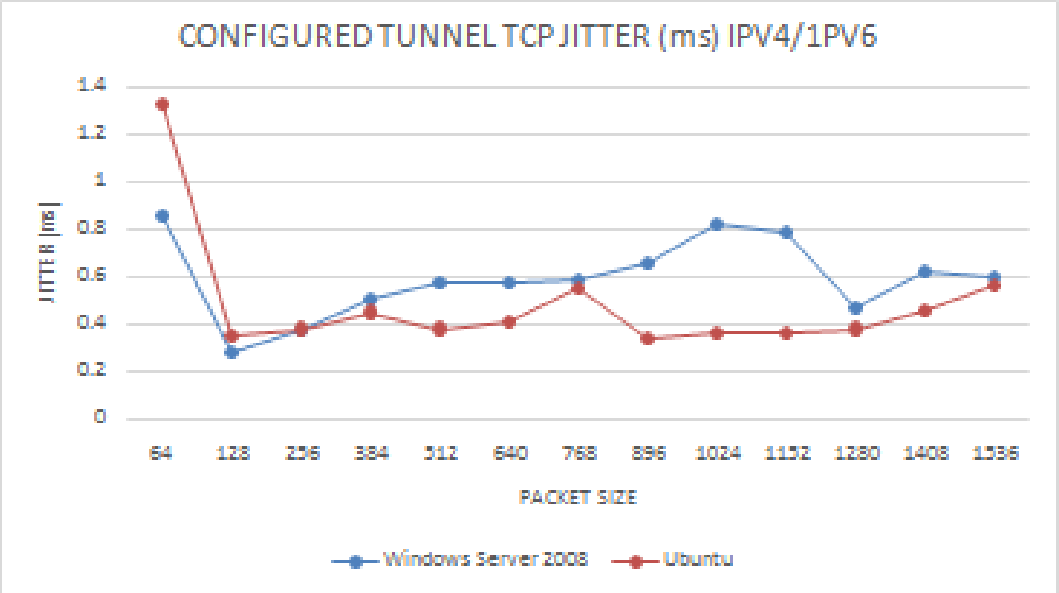

Figure 8: CONFIGURED TUNNEL TCP JITTER

Figure 8 shows the comparism of windows server 2008 and Ubuntu. From the figured above, it is observed that at time 64, 128 and 256 windows server 2008 had lower jitter time compared to Ubuntu but for all other packet sizes Ubuntu had lower jitter than windows server 2008 except at time 384 , they were the same.

ISSN 1512-1232

CONFIGURED TUNNEL (TCP DELAY(ms))		
PACKET SIZE	WINDOWS SERVER 2008	UBUNTU
64	180.86	173.53
128	39.31	18.11
256	45.43	22.72
384	41.82	20.24
512	46.11	24.09
640	45.96	27.31
768	47.12	29.50
896	49.61	35.57
1024	51.83	41.19
1152	49.47	39.54
1280	51.84	40.66
1408	54.80	42.30
1536	54.47	42.68

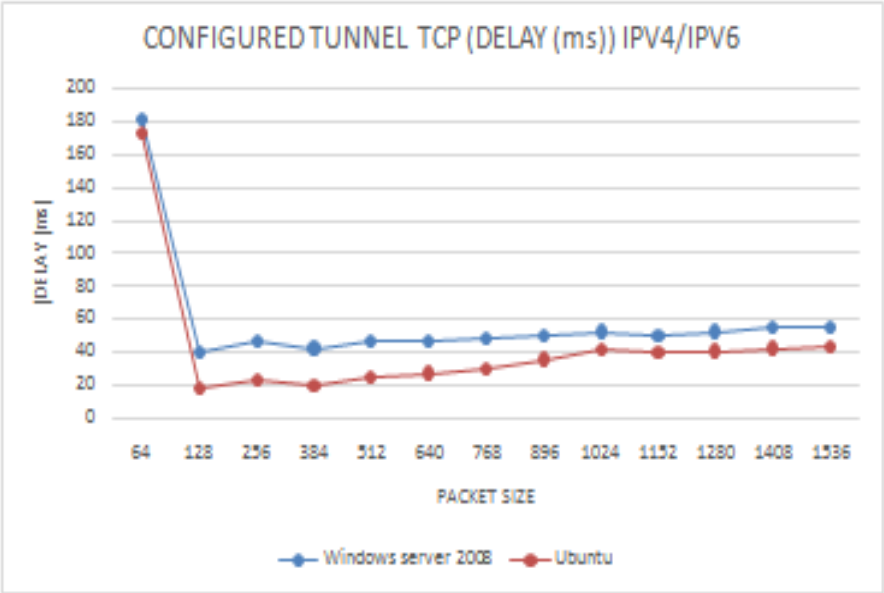

Figure 9: CONFIGURED TUNNEL TCP DELAY

Figure 9 shows the delay time experienced by both operating systems. At all packet sizes Ubuntu had lower amount of delay time compared to Windows server 2008.

ISSN 1512-1232

CONFIGURED TUNNEL (TCP CPU UTILIZATIONROUTER1 (%))		
PACKET SIZE	WINDOWS SERVER 2008	UBUNTU
64	14.84	18.00
128	24.22	14.00
256	28.13	17.00
384	27.34	16.00
512	26.56	16.00
640	31.13	14.00
768	24.23	15.00
896	21.00	16.00
1024	19.75	16.00
1152	30.36	17.00
1280	34.39	18.00
1408	27.64	13.00
1536	23.26	17.00

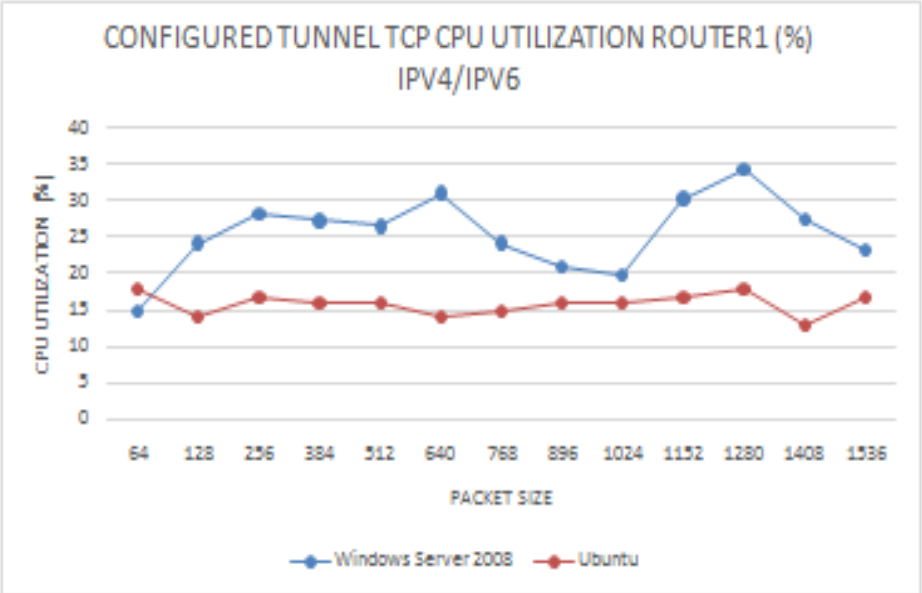

Figure 10:CONFIGURED TUNNEL TCP CPU UTILIZATION

Figure 10 shows the comparism of both operating systems using CPU utilization. From the figure above, it is observed that at size 64 windows server 2008 had a lower percentage than CPU utilization but for all other packet sizes Ubuntu had lower percentage for CPU utilization compared to windows server 2008. And Ubuntu had no decimal values for all, its value.

ISSN 1512-1232

CONFIGURED TUNNEL (UDPTHROUGHPUT (Mbps))		
PACKET SIZE	WINDOWS SERVER 2008	UBUNTU
64	33.24	35.34
128	50.49	52.60
256	67.32	68.66
384	74.61	76.24
512	78.96	80.59
640	82.15	83.63
768	84.11	85.48
896	85.70	87.13
1024	86.79	87.67
1152	87.95	89.19
1280	81.70	89.60
1408	82.03	90.35
1536	83.19	84.11

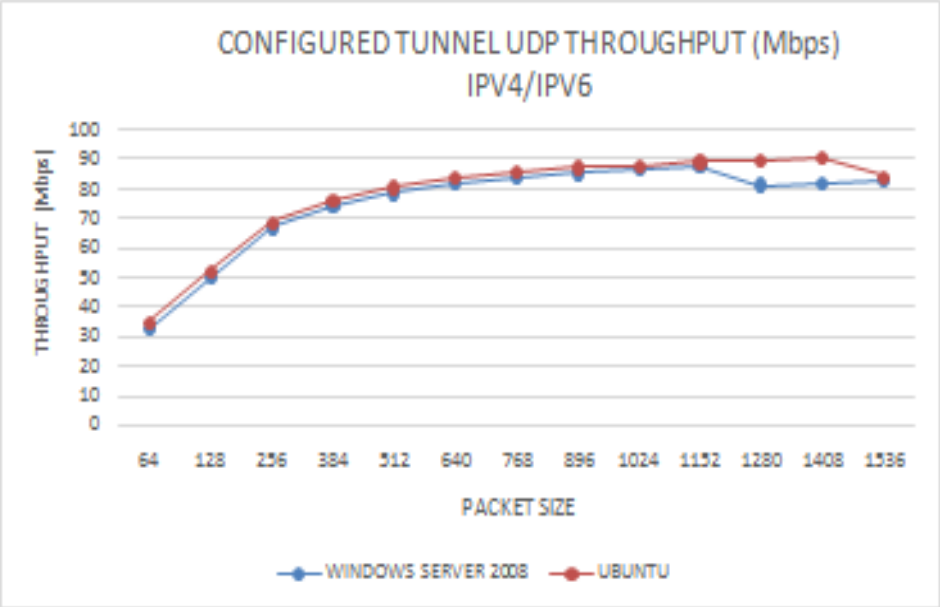

Figure 11: CONFIGURED TUNNEL UDP THROUGHPUT

In Figure 11 both operating systems had close range of amount of time from size 64 to 1152, at time 1280 and 1408 windows server 2008 had lower amount of throughput time with a difference of 7.9 and 8.32 respectively.

ISSN 1512-1232

CONFIGURED TUNNEL (UDP JITTER (ms))		
PACKET SIZE	WINDOWS SERVER 2008	UBUNTU
64	0.13	0.23
128	0.24	0.23
256	0.26	0.24
384	0.21	0.23
512	0.27	0.24
640	0.29	0.28
768	0.35	0.29
896	0.30	0.21
1024	0.36	0.25
1152	0.65	0.41
1280	0.79	0.72
1408	0.75	0.78
1536	0.84	0.81

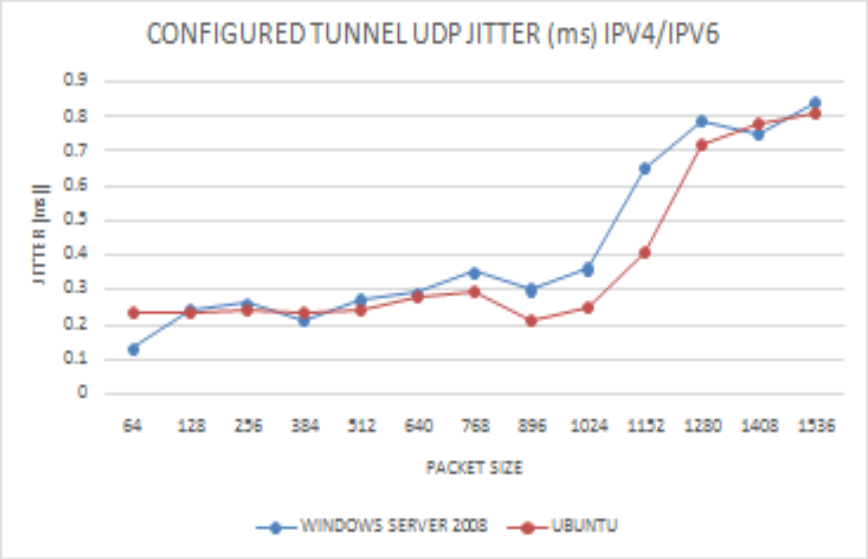

Figure 12: CONFIGURED TUNNEL UDP JITTER

In figure 12, at time 64 windows server 2008 had a lower jitter time but at almost all other packet sizes Ubuntu had lower amount of jitter time except at time 1408 with an higher time difference of 0.03 than windows server 2008.

ISSN 1512-1232

CONFIGURED TUNNEL (UDP DELAY (ms))		
PACKET SIZE	WINDOWS SERVER 2008	UBUNTU
64	54.22	71.26
128	41.31	71.70
256	39.26	82.09
384	34.67	90.06
512	43.83	97.80
640	47.21	108.00
768	47.59	110.70
896	51.02	126.71
1024	45.87	96.10
1152	42.13	98.66
1280	49.49	91.12
1408	49.06	88.74
1536	54.92	92.34

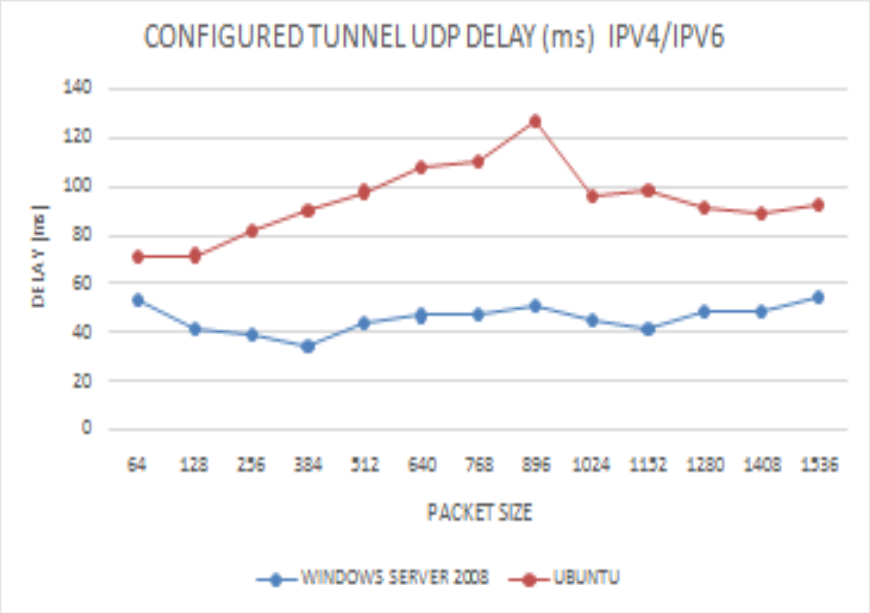

Figure 13: CONFIGURED TUNNEL UDP DELAY

In figure 13, windows server 2008 had lower amount of delay compared to Ubuntu for all packet sizes used.

ISSN 1512-1232

6to 4 (UDP DELAY (ms))		
PACKET SIZE	WINDOWS SERVER 2008	UBUNTU
64	1103.32	23.45
128	1102.58	22.58
256	1101.14	32.95
384	1105.47	43.77
512	1106.39	50.23
640	1106.94	59.85
768	1111.54	71.04
896	1113.78	74.66
1024	1112.94	74.40
1152	1108.67	73.72
1280	1108.31	64.83
1408	1108.76	75.11
1536	1114.79	64.57

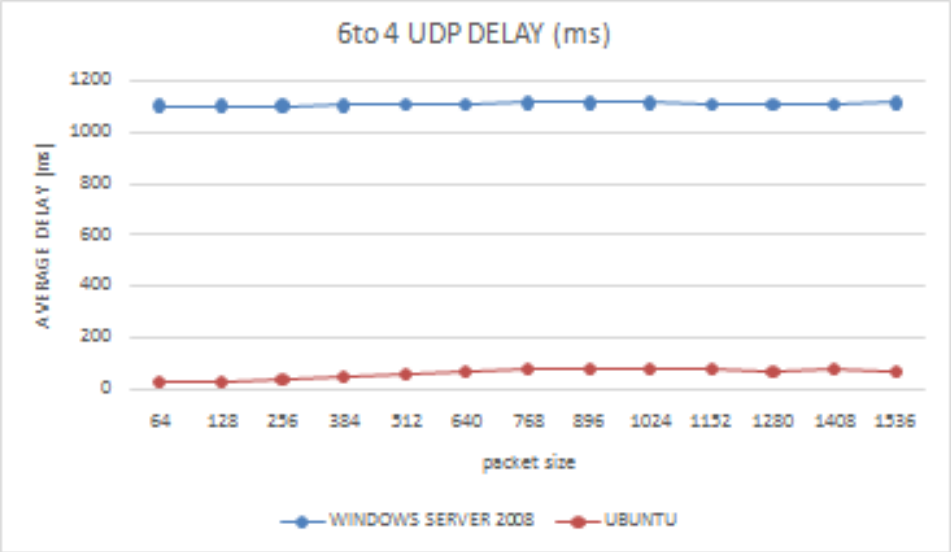

Figure 14: 6TO4 UDP DELAY

In figure 14, there was a very high difference between the delay experienced by windows server 2008 and Ubuntu. Ubuntu experienced very low amount of delay time compared to windows server 2008 that experienced very high delay time.

ISSN 1512-1232

6to 4 (UDP JITTER (ms))		
PACKET SIZE	WINDOWS SERVER 2008	UBUNTU
64	0.15	0.23
128	0.24	0.28
256	0.26	0.21
384	0.29	0.10
512	0.29	0.21
640	0.28	0.16
768	0.31	0.25
896	0.19	0.27
1024	0.26	0.31
1152	0.64	0.33
1280	0.57	0.61
1408	0.70	0.67
1536	0.83	0.77

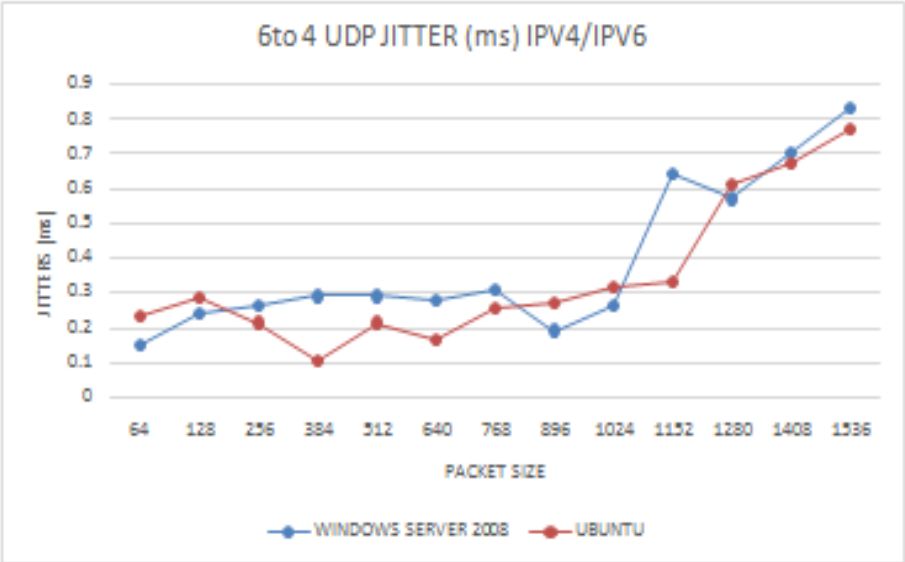

Figure 15: 6TO4 UDP JITTER

In figure 15, at time 256, 384, 512, 640, 768, 1152,1408 and 1536, Ubuntu had lower jitter time compared to windows server 2008 but for all other packet sizes windows server 2008 recorded lower amount of jitter time.

ISSN 1512-1232

6to 4 (UDP TROUGHPUT (Mbps))		
PACKET SIZE	WINDOWS SERVER 2008	UBUNTU
64	36.05	33.78
128	53.13	52.30
256	68.85	68.74
384	76.30	76.37
512	80.68	80.57
640	83.48	83.33
768	85.15	85.52
896	86.86	86.96
1024	88.12	88.08
1152	88.69	89.20
1280	82.86	89.62
1408	83.35	90.44
1536	84.58	84.30

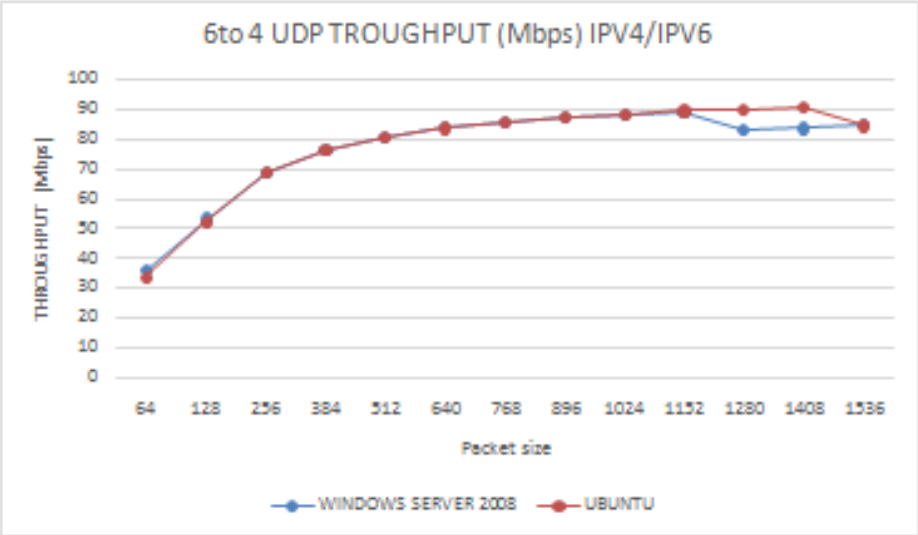

Figure 16: 6TO4 UDP THROUGHPUT

In figure 16, both operating system had close range of throughput time except at time 1208 and 1408, where windows server 2008 had lower throughput time with a difference of 6.94 and 6.89 respectively compared to Ubuntu.

DISCUSSION

In configured tunnel of TCP, it was observed that at time 640, 1280 and 1408 windows server 2008 had lesser amount of throughput time when compared with Ubuntu but for all other packet sizes they had almost the same amount of throughput time. In configured tunnel of UDP, both operating systems had a close range of amount of time from size 64 to 1152, at time 1280 and 1408 windows server 2008 had lower amount of throughput time with a difference of 7.9 and 8.32 respectively. Generally, for configured tunnel of TCP throughput, the throughput time is better in Ubuntu than in Microsoft windows 2008. In addition, the configured tunnel of TCP throughput time of Ubuntu is also better than in configured tunnel of UDP as the former involves better transfer of packets than the latter. This correlates with the experiment performed by Soorty and Sarkar(2013); Kolahi et al. 2011.

ISSN 1512-1232

Implementing a configured tunnel on IPv4 and IPv6 showed no significant changes on the two operating systems for jitter. Though, it was noticed that at time 64, 128 and 256 windows server 2008 had lower jitter time compared to Ubuntu but for all other packet sizes Ubuntu had lower jitter than windows server 2008.

For the delay time experienced by both operating systems, all packet sizes in Ubuntu had lower amount of delay time compared to Windows server 2008.

In 6TO4 throughput, both operating systems had a close range of throughput time except at time 256, 640, 1280 and 1408, where windows server 2008 had lower throughput time with a difference of 1.32, 4.52, 7.38 and 6.72 respectively compared to Ubuntu.

In 6 to 4 TCP Jitter and UDP jitter, Ubuntu had lower jitter time compared to windows server 2008 but for all other packet sizes windows server 2008 recorded lower amount of jitter time.

Furthermore, In 6to4 TCP and UDP delay, there was a very high difference between the delay experienced by windows server 2008 and Ubuntu. Ubuntu experienced very low amount of delay time compared to windows server 2008 that experienced very high delay time (Narayan and Tauch, 2010); (Chandra and Lalitha, 2015). The line graph was also similar for both operating systems.

In this study, implementing configured tunnel is better in Ubuntu than in windows operating system as it shows much better performance on linux. The research work has been able to establish configured tunnel mechanisms using some metrics to compare the performance of two operating systems.

CONCLUSION

The aim of this paper is to evaluate the performance of IPv4 and IPv6 on Microsoft operating system and Linux operating system using two transition mechanisms. The performance measurement was examined on two types of transmission protocols namely: TCP (Transmission Control Protocol) and also UDP (User Datagram Protocol). The performance metrics used are throughput, delay, jitter, CPU utilization and they were used over a range of bytes.

After the successful completion of the experimental design and network setup, this paper was able to show:

[1] The evaluation of the performance of IPv4 and IPv6 on Microsoft operating system and Linux operating system also depends on the transition mechanisms being used. For example, 6to4 and configured tunnel perform better on Linux than on Microsoft operating system.

[2] TCP is better for smaller packet sizes while UDP is better for larger packet sizes.

A good and clear knowledge of the understanding of the transition mechanisms is necessary for a better understanding of this work.

Further research can also be done for this paper because there are lots of other transition mechanisms and performance metrics that can be used for performance evaluation such as: Dual Stack, 4over 6 tunnels, NAT-PT.

In order to carry out a proper research on this topic, there is a need to be in an environment where IPv4 are IPv6 are being used for further evaluation of this work.

References

1. Ogunde, A. (2015). Lecture on Introduction to Data Communication. Personal Collection of A. Ogunde, Redeemer's University, Ede, Nigeria.
2. Lammle, T. (2013). CCNA: Routing and Switching. Study Guide.
3. Sotharith, T. (2010). *Performance Evaluation of IP version 4 and IP version 6 transitions mechanisms on various operating Systems* (Master's thesis, UNITEC Institute of Technology, Auckland, New Zealand).

ISSN 1512-1232

4. 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.
5. Davies, J. (2008). Understanding IPv6 (2nd Ed.). Washington: Microsoft Press.
6. Blum, R. (2003).Network performance open source toolkit: Using Netperf, tcptrace, NIST Net, and SSFNet. Indianapolis: Wiley.
7. SourceForge. (2009). Iperf. Retrieved March 23, 2016, from http://iperf.sourceforge.net
8. Soorty B. and Sarkar N (2013), Quantifying TCP Performance for IPv6 in Linux-Based Server Operating Systems, Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT),Volume 3, Issue 11.
9. Kolahi S. and Soorty B.(2011) "Evaluation of Gigabit Ethernet Local Area Networks in Windows Vista-Server 2008 Environment," in IEEE Workshops of Int. Conf. Advanced Information Networking and Applications (AINA), pp. 308-312.
10. Narayan S. and Tauch S.(2010), "IPv4-v6 Transition Mechanisms network Performance Evaluation on Operating Systems", IEEE 2nd International Conference on Signal Processing Systems and Applications (ICSPS) .
11. Chandra A. and Lalitha K. (2015), IPv4 to IPv6 Network Migration and Co-Existence, International Research Journal of Engineering and Technology (IRJET), Vol 2, Issue 2.

Article received: 2016-10-21