Evaluating IPv4 to IPv6 Transition Mechanisms

*Ioan Raicu
Department of Computer Science
Purdue University
1398 Computer Science Building
West Lafayette, IN 47907, USA
iraicu@cs.purdue.edu

Sherali Zeadally
Department of Computer Science
Wayne State University
5143 Cass Ave., 454 State Hall
Detroit, MI 48202, USA
zeadally@cs.wayne.edu

Abstract

The next-generation Internet Protocol, initially known as IP Next Generation (Ipng), and then later as IPv6, has been developed by the Internet Engineering Task Force (IETF) to replace the current Internet Protocol (also known as IPv4). To enable the integration of IPv6 into current networks, several transition mechanisms have been proposed by the IETF IPng Transition Working Group. This work examines and empirically evaluates two transition mechanisms, namely 6-over-4, and IPv6 in IPv4 tunneling, as they relate to the performance of IPv6. We explore the impact of these approaches on end-to-end user application performance using metrics such as throughput, latency, host CPU utilization, TCP connection time, and the number of TCP connections per second that a client can establish with a remote server. All experiments were conducted using two dual stack (IPv4/IPv6) routers and two end-stations running Windows 2000, loaded with a dual IPv4/IPv6 stack.

Keywords: IPv4, IPv6, 6-over-4, encapsulation, tunneling, performance evaluation

1 INTRODUCTION

Over the last decade, IETF has been working on the deployment of IPv6 [1, 2] to replace the current IPv4 [3] protocol. The motivations behind IPv6 are briefly discussed in the following section and are covered in the literature [4, 5, 6, 7]. One of the biggest challenges in the deployment of IPv6 is how to migrate IPv4-based infrastructures to those supporting IPv6. It is impractical and costly to replace existing IPv4-based networking infrastructures with IPv6. To ensure a smooth and successful integration of IPv6 into existing networks, the IETF IPng Transition Working Group [1] has been working on several transition strategies, tools, and mechanisms. In general, these transition mechanisms encapsulate IPv6 packets into IPv4 packets and transport them over an IPv4 network infrastructure. We expect to rely on these transition strategies as the Internet shifts from the traditional IPv4 to an IPv6-based Internet while retaining both IPv4 and IPv6 throughout the transition phase.

* *The work presented herein was completed by the author while he was a graduate student in the Department of Computer Science at Wayne State University, Detroit, Michigan, USA.*

The main goal of this work is to empirically evaluate two transition mechanisms, namely *6-over-4* [8] and *IPv6 in IPv4 tunneling* [9], and assess the impact of these approaches on end-to-end application performance. We evaluate the performance impact of these mechanisms in a real world setting, which includes hosts and routers supporting dual IPv4/IPv6 stacks. [10] The 6-over-4 mechanism performs the encapsulation of IPV6 into IPV4 packets at the host, and we will refer to it as *host-to-host encapsulation* henceforth. The IPv6 in IPv4 tunneling performs the encapsulation at the routers, and we will refer to it as *router-to-router tunneling* throughout the paper.

Section 2 covers background information about the fundamental differences between IPv4 and IPv6 and some related works. It also presents a brief overview of the various transition mechanisms, including a significant review of router-to-router tunneling and host-to-host encapsulation. Section 3 describes the test-bed configurations used and our measurement procedures. Section 4 discusses our experimental results. Finally, in section 5, we make some concluding remarks and discuss future work.

2 BACKGROUND

2.1 IPv4 and IPv6 Architecture

Internet Protocol was first developed in the early 1980s. In the early 1990s, it became pretty evident that if the Internet will continue to grow at the rate it was growing, the IPv4 address space would be depleted by the turn of the millennium. Some temporary solutions were offered, such as NAT (Network Address Translator) [11] or CIDR

(Classless InterDomain Routing) [7], however work began on a new Internet Protocol, namely IPv6.

The main reason for a new version of the Internet Protocol was to increase the address space; IPv6 was designed with a 128 bit address scheme, enough to label every molecule on the surface of the earth with a unique address [7]. Furthermore, the only kind of traffic that existed on the internet twenty years ago was elastic traffic, such as emails or file transfers. These kinds of traffic were very flexible regardless of the network conditions; on the other hand, inelastic traffic requires a certain level of guaranteed performance, which if not met, the application does not have the same usefulness. IPv6 was designed for efficiently supporting both elastic and inelastic traffic.

The goals of IPv6 were to support scalability, security, and multimedia transmissions. First, the address space is increased from 32 bits to 128 bits. Unlike IPv4, IPSec [12] support has become a requirement in the IPv6 header. Payload identification for QoS handling by routers is now supported by the *Flow Label* field in the IPv6 packet header. Fragmentation support has been moved from routers to the sending hosts. The IPv6 header does not include a checksum and has no options included in the header, but rather introduces extension headers. Finally, IPv6 requires no manual configuration or DHCP (Dynamic Host Configuration Protocol), which will become important as the number of nodes increases. Overall, IPv6 was carefully thought out and was designed with future applications in mind. [7]

The main difference in the packet layout between IPv4 and IPv6 is that IPv4 has a 20 byte header while IPv6 has a 40 byte header. Although the address space in IPv6 is four times the size of its counterpart, IPv6 has reduced the number of required fields and made them optional as extension headers. Since the Ethernet MTU size is 1514 bytes, the additional 20 bytes of header information only incurs an additional 1.3% overhead; an additional 20 bytes of header information when an IPv6 packet is encapsulated in an IPv4 packet raises the overall overhead to 2.6%. In theory, this performance overhead between these two protocols is minimal.

2.2 IPv4 to IPv6 Transition Mechanisms

Some currently available transition mechanisms are: Dual Stacks [10], DTI & Bump-in-dual-stack, NAT Protocol Translator [13], Stateless IP/ ICMP Translator (SIIT), Assignment of IPv4 Global Addresses to IPv6 Hosts (AIIH), Tunnel Broker [14], 6-to-4 Mechanism [15], 6-over-4 Mechanism [8], and IPv6 in IPv4 tunneling [9]. Dual Stacks are easiest to implement, however complexity increases at the hosts due to both infrastructures and the cost is higher due to a more complex technology stack. NAT Protocol Translator has scaling and DNS issues, and has single point of failure disadvantage. The Tunnel Broker dynamically gains access to tunnel servers, but has authentication and scaling issues. 6-to-4 mechanism creates dynamic stateless tunnels over IPv4 infrastructure to connect 6-to-4 domains. 6-over-4 mechanism allows the interconnection of isolated IPv6 hosts to connect over the IPv4 infrastructure without requiring IPv6 enabled routers or explicit tunnels. IPv6 in IPv4 tunneling allows existing infrastructure to be utilized via manually configured tunnels.

We chose to pursue the *IPv6 in IPv4 tunneling* and *6-over-4* as a transition mechanism. 6-over-4 mechanism performs the encapsulation at the host, and therefore we will refer to it as *host-to-host encapsulation* throughout the paper. The IPv6 in IPv4 tunneling performs the encapsulation at the routers, and hence we will refer to it as *router-to-router tunneling*. The router-to-router tunneling enables two entire LANs to be upgraded to IPv6 while maintaining connectivity to the rest of the Internet. Host-to-host encapsulation is also addressed mainly because of its simplicity of implementation, and offers another method of making the transition from IPv4 to IPv6 as smooth as possible.

Encapsulation of IPv6 packets within IPv4 packets allows two IPv6 hosts/networks to be connected with each other while running on existing IPv4 networks. IPv6 packets are encapsulated in IPv4 packets and then are transmitted over IPv4 networks like ordinary IPv4 packets. At the destination, these packets are de-capsulated to the original IPv6 packets. It should be noted that encapsulation of IPv6 packets in IPv4 packets, only IPv4 routing properties will be utilized and hence the IPv6 packet will lose any special IPv6 features until it is de-capsulated at the receiving host/router. It requires a hole in a firewall to allow protocol 41 (IP in IP encapsulation, or in our case, IPv6 in IPv4) passage.

2.2.1 Host-to-Host Encapsulation

In the host-to-host encapsulation method, the encapsulation is done at the source host and the de-capsulation is done at the destination host. The encapsulated datagrams are sent through a native IPv4 network that has no knowledge of the IPv6 network protocol. Figure 1 illustrates two hosts, each with dual IPv4/IPv6 stacks which encapsulate the packets of IPv6 in IPv4 packets and transmit over the network as an IPv4 packet. With this transition mechanism, it is possible to support IPv6 simply by upgrading the end hosts protocol stacks to IPv6 while leaving the IPv4 infrastructure unchanged.

Figure 2 depicts an IPv6 packet encapsulated in an IPv4 packet. All the various header fields (IP version, Flow Label, Next Header, etc) are clearly visible just as we had described them in the earlier section.

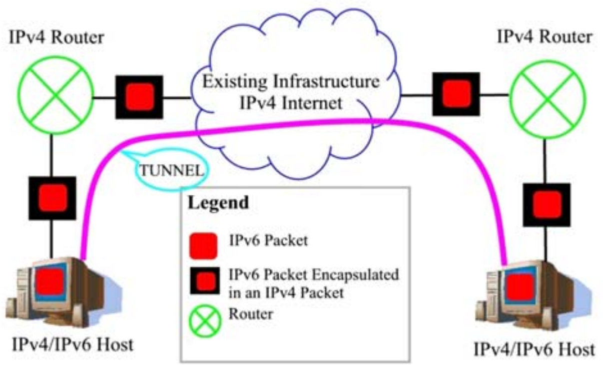
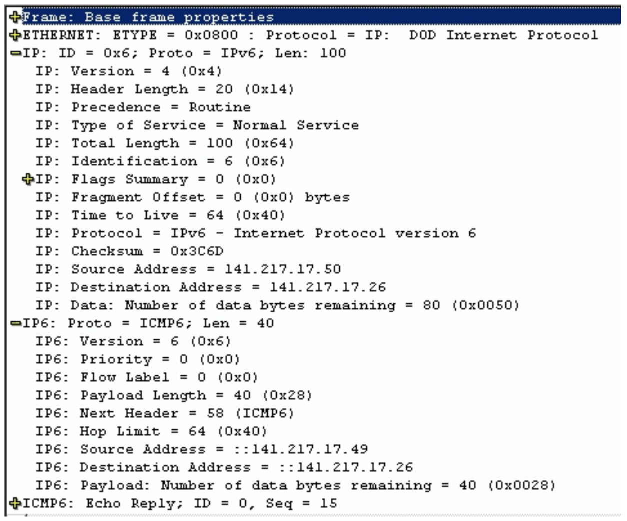


Figure 1: Host-to-Host tunneling



```
+Frame: Base frame properties
+ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
-IP: ID = 0x6; Proto = IPv6; Len: 100
   IP: Version = 4 (0x4)
   IP: Header Length = 20 (0x14)
   IP: Precedence = Routine
   IP: Type of Service = Normal Service
   IP: Total Length = 100 (0x64)
   IP: Identification = 6 (0x6)
  +IP: Flags Summary = 0 (0x0)
   IP: Fragment Offset = 0 (0x0) bytes
   IP: Time to Live = 64 (0x40)
   IP: Protocol = IPv6 - Internet Protocol version 6
   IP: Checksum = 0x3C6D
   IP: Source Address = 141.217.17.50
   IP: Destination Address = 141.217.17.26
   IP: Data: Number of data bytes remaining = 80 (0x0050)
-IP6: Proto = ICMP6; Len = 40
   IP6: Version = 6 (0x6)
   IP6: Priority = 0 (0x0)
   IP6: Flow Label = 0 (0x0)
   IP6: Payload Length = 40 (0x28)
   IP6: Next Header = 58 (ICMP6)
   IP6: Hop Limit = 64 (0x40)
   IP6: Source Address = ::141.217.17.49
   IP6: Destination Address = ::141.217.17.26
   IP6: Payload: Number of data bytes remaining = 40 (0x0028)
+ICMP6: Echo Reply; ID = 0, Seq = 15
```

Figure 2: IPv6 packet encapsulated in an IPv4 packet depicted by the Microsoft Network Monitor

2.2.2 Router-to-Router Tunneling

In router-to-router tunneling mechanism, encapsulation is done at the *edge router* of the originating host and de-capsulation is done at the edge router of the destination host.

The tunnel is created between the two edge routers supporting both IPv4 and IPv6 stacks. Therefore, the end hosts can support native IPv6 protocol stack while the edge routers create the tunnels and handle the encapsulation and de-capsulation of IPv6 packets over the existing IPv4 infrastructure.

Figure 3 shows a tunnel established between two dual stack edge routers. The IPv6 packets are forwarded from host to edge routers while encapsulation takes place at the router level; similarly at the other end, the reverse process takes place. In this method, both edge routers need to support dual stacks and established a tunnel prior to transmission. Note that other transition mechanisms exists that dynamically establish the tunnels as needed.

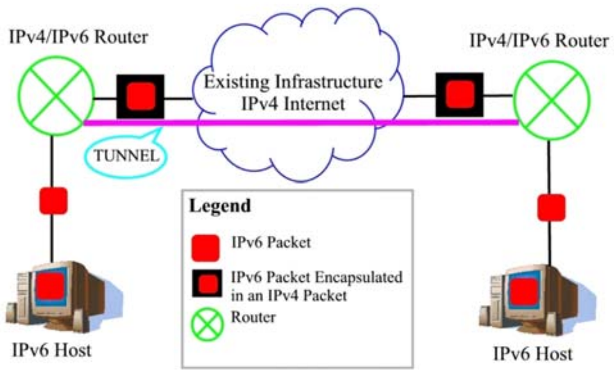


Figure 3: Router-to-Router Tunneling

2.3 Related Work

It is important to point out that our work was driven by the fact that there is no empirical performance experimentation to evaluate any of the transition mechanisms that currently exist to assist in the deployment of IPv6 networks. This work extends our previous work [16] where we compared the IPv4 protocol and the IPv6 protocol under various operating systems and testbed configurations.

Most of the industry wide routers implement their functionality in hardware and are therefore we believe that hardware based routers are more efficient than a software-based router implementation. The reason few researchers tested IPv6's performance using hardware-based routers supporting dual stack IPv4/IPv6 are relatively expensive; as an example, the two routers we used for our experiments cost a total of US $60,000. As a result, most of the work done in the research community has been performed using software-based routers utilizing off-the-shelf PCs. Various works have been attempted [17, 18, 19, 20, 21] which evaluated the IPv6 protocol stack, however none of them used hardware-based routers, had such a wide range of metrics, and none investigated transition mechanisms.

3 EXPERIMENTAL TESTBED AND MEASUREMENT PROCEDURES

3.1 Testbed Configuration

Our testbed consisted of two dual stack (IPv4/IPv6) routers: an Ericsson AXI 462, and an IBM 2216 Nways Multiaccess Connector Model 400. In addition, two identical workstations were connected directly to the routers and were configured to be on separate networks.

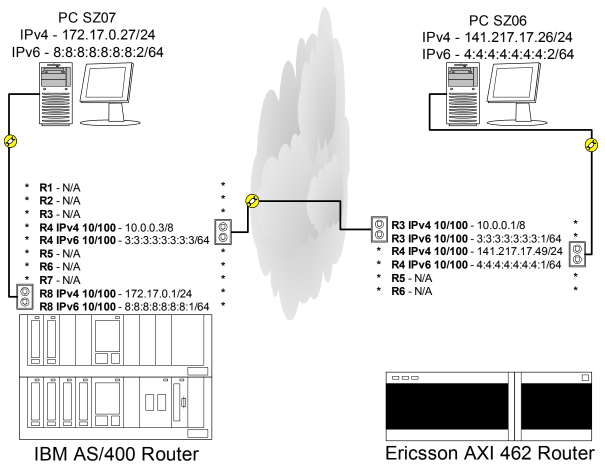

PC SZ07
IPv4 - 172.17.0.27/24
IPv6 - 8:8:8:8:8:8:8:2/64
PC SZ06
IPv4 - 141.217.17.26/24
IPv6 - 4:4:4:4:4:4:4:2/64
R1 - N/A
R2 - N/A
R3 - N/A
R4 IPv4 10/100 - 10.0.0.3/8
R4 IPv6 10/100 - 3:3:3:3:3:3:3:3/64
R5 - N/A
R6 - N/A
R7 - N/A
R8 IPv4 10/100 - 172.17.0.1/24
R8 IPv6 10/100 - 8:8:8:8:8:8:8:1/64
R3 IPv4 10/100 - 10.0.0.1/8
R3 IPv6 10/100 - 3:3:3:3:3:3:3:1/64
R4 IPv4 10/100 - 141.217.17.49/24
R4 IPv6 10/100 - 4:4:4:4:4:4:4:1/64
R5 - N/A
R6 - N/A
IBM AS/400 Router
Ericsson AXI 462 Router

Figure 4: IBM-Ericsson Testbed architecture; two routers are depicted, an IBM 2216 Nways Multiaccess Connector Model 400 and an Ericsson AXI 462

Both workstations used were equipped with Intel Pentium III 500 MHz processors, 256 megabytes of SDRAM PC100, 30GB IBM 7200 RPM IDE hard drive, and COM 10/100 PCI network adapters. The workstations were loaded with Windows 2000 Professional. Windows 2000 had the IPv4 stack as a standard protocol; however in order to get IPv6 support, an add-on package was installed. There were two add-on package choices to choose from, both written by Microsoft and in Beta testing. We chose the newer release of the two, "Microsoft IPv6 Technology Preview for Windows 2000" [22] which is supported by Winsock 2.

Our testbed configuration shown in Figure 4 a depicts the entire testbed utilizing two routers in between the two end hosts connected via a 100 Mbit/s link. Note that the IP addresses of the end hosts are not on the same network, and hence we have the routers to allow communication between the two distinct networks. In the Ericsson router, R3 through R6 are the various network cards available; each interface card has both an IPv4 and an IPv6 address. Similarly, on the IBM router, R1 through R8 are the various network cards that are available; each interface card has both an IPv4 and an IPv6 address.

3.2 Measurement Procedures

The majority of the tests were done for a sufficiently long period of time and resulted in the exchange of 50,000 packets to 1,000,000 packets, depending on the size of the packets sent and the corresponding test. We conducted an empirical measurement based on the following performance metrics: throughput, latency, CPU utilization, TCP connection time, and the number of TCP connections per second. All the performance measurement software was written in C++.

- **Throughput:**
  The rate at which bulk data transfers can be transmitted from one host to another over a sufficiently long period of time; the performance is measured in Mbit/s
- **Latency**
  Latency, also known as RTT (round trip time), is the amount of time it takes one packet to travel from one host to another and back to the originating host; the performance is measured in microseconds per RTT
- **CPU Utilization**
  This metric was measured only for Windows 2000 using the Windows Task Manager under the performance tab; the average CPU utilization was recorded throughout each corresponding experiment; the performance is measured in percentage of utilization
- **TCP Connection Time**
  This metric measured the amount of time it took to establish a TCP connection between the client and the server; the performance is measured in microseconds per TCP connection
- **Web Client/Server Simulation**
  This simulation investigated the maximum number of tests that could be performed per second by the client/server applications; the test comprised of: create a socket, make a TCP connection, send a 1-byte message, receive a 1-byte message, tear down the connection, and destroy the socket; the performance is measured in number of tests per second

We conducted three basic performance comparison tests over the TCP transport protocol under the Windows 2000 operating system:

- Homogeneous IPv6 Network (Figure 5) is referred to as TCP/IPv6 Stack *Test Name* in Figure 8 through Figure 11
- Heterogeneous IPv4 and IPv6 Network (Figure 6), known as IPv6 in IPv4 Tunneling, is referred to as TCP/IPv6 Stack *Test Name* (Router-to-router Tunneling) in Figure 8 through Figure 11
- Heterogeneous IPv4 and IPv6 Network (Figure 7), known as 6-over-4, is referred to as TCP/IPv6 Stack *Test Name* (Host-to-host Encapsulation) in Figure 8 through Figure 11

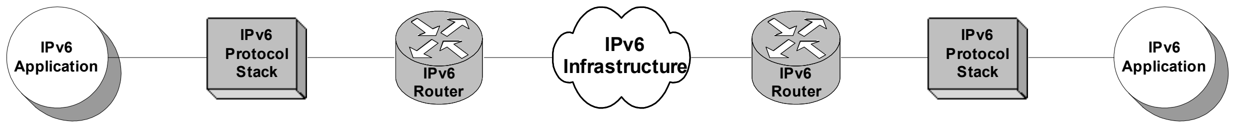


Figure 5: IPv6 Homogeneous Network

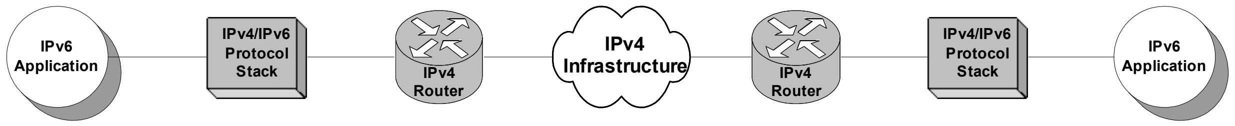


Figure 6: Router-to-router Tunneling IPv4/IPv6 Heterogeneous Network

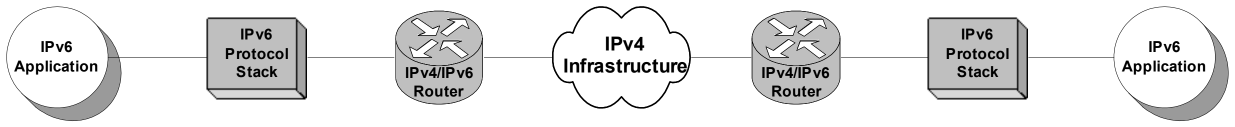


Figure 7: Host-to-host Encapsulation IPv4/IPv6 Heterogeneous Network

Note that all the basic configurations above all have IPv6-based applications, and the major difference between the various configurations is the infrastructure that is required for two end applications to communicate.

4 EXPERIMENTAL RESULTS

4.1 Throughput

As Figure 8 indicates, every layer of complexity adds additional overheads for all packet sizes.

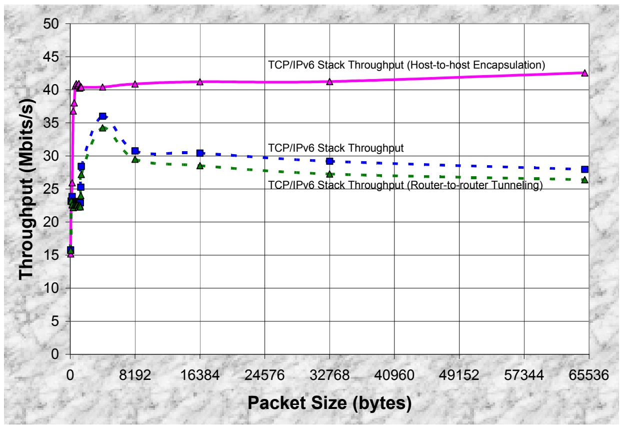


Figure 8: TCP throughput results for IPv6, and transition mechanisms with packet size ranging from 64 bytes to 64 Kbytes

Router-to-router tunneling seems to have very little overhead on top of the IPv6 protocol stack. Specifically, it incurs about 1% to 7% overhead. On the other hand, the host-to-host encapsulation performed surprisingly better than IPv6; this may seem counter intuitive, however the better performance can be justified to the better performance of the IPv4 infrastructure [16]. Once a packet is encapsulated at the host, it will traverse the entire network as an IPv4 packet, and only at the receiving host will the packet be de-capsulated. As for the CPU utilization depicted in Figure 9, the host-to-host encapsulation incurred the most CPU overhead at the host. This was expected since the end host had to encapsulate and de-capsulate (at the host where the CPU utilization measurements were taken) every single packet that was transmitted or received, while the other two tests simply sent and received IPv6 packets.

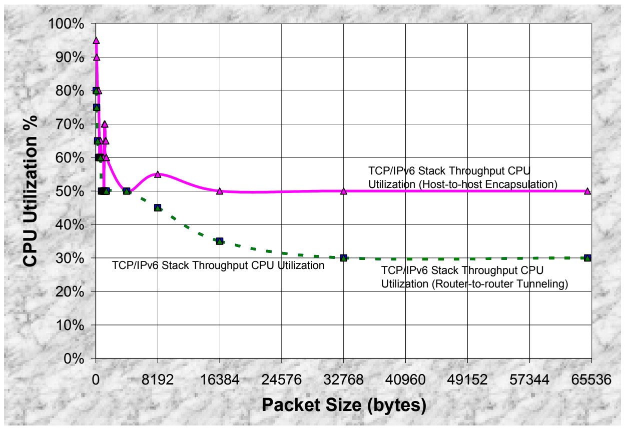


Figure 9: CPU utilization for TCP throughput results for IPv6, and IPv4-IPv6 transition mechanisms with packet size ranging from 64 bytes to 64 Kbytes

4.2 Latency

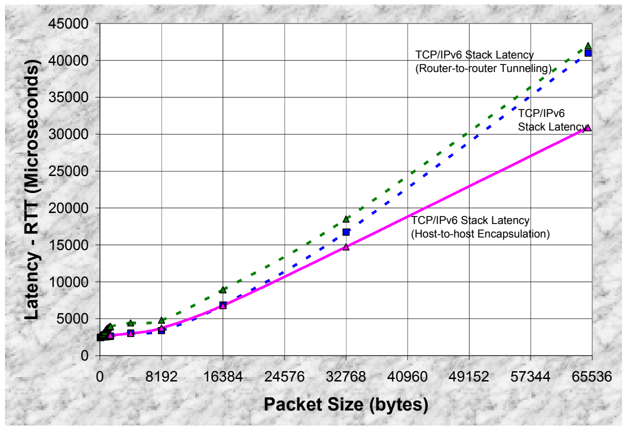


Figure 10: TCP latency results for IPv6, and transition mechanisms with packet size ranging from 64 bytes to 64 Kbytes

For the 64 Kbyte packets, IPv6 has a latency of about 40 ms while the host-to-host encapsulation were 30 ms. The router-to-router tunneling experienced similar trends as IPv6 with RTTs as high as 42 ms for 64 Kbyte packets. Again, the host-to-host encapsulation RTT seems to be counter intuitive to be less than that of IPv6 with no encapsulation mechanism; however the better performance can be justified to the lower latency of the IPv4 infrastructure [16]. Once a packet is encapsulated at the host, it will traverse the entire network as an IPv4 packet, and only at the receiving host will the packet be de-capsulated.

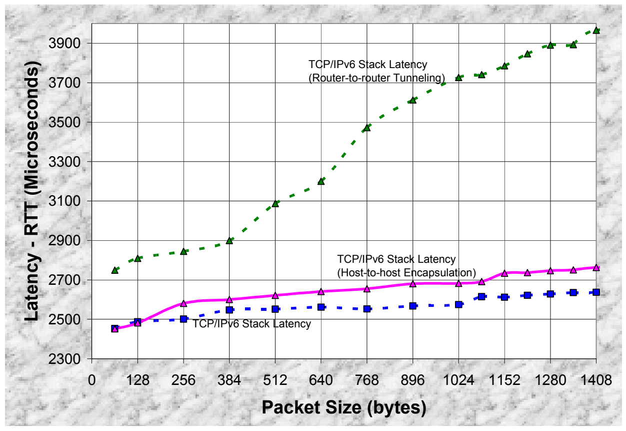


Figure 11: TCP latency results for IPv6, and transition mechanisms with packet size ranging from 64 bytes to 1408 bytes

As Figure 11 indicates, the router-to-router tunneling incurs a relatively high performance overhead on top of all the other experiments. When the packet sizes are small, the routers take a relative constant amount of time in order to process each encapsulation and de-capsulation, and therefore we see the large increase in RTT's for small packet sizes. The difference is amortized as the packet size is larger and eventually the router-to-router tunneling performance overhead is minimized for larger packet sizes.

4.3 TCP Connection Time

Table 1 shows that various connection times in each experiment conducted for the TCP protocol. The host-to-host encapsulation had the fastest connection time. The next best was the native IPv6 network protocol and the worst was the router-to-router tunneling. These numbers represent performance overhead of 6% to 17% between IPv6 and the two transition mechanisms. This metric is important for applications that have many TCP connections they must perform in relatively small lengths of time.

IP Version	Connection Time (microseconds)
IPv6	2959.13
Host-to-host Encapsulation	2784.42
Router-to-router Tunneling	3261.58

Table 1: TCP connection time in microseconds for IPv6, and the transition mechanisms

4.4 Web Client/Server Simulation

The results from Table 2 shows that the order of the results is really the same as it was in the previous subsection with the connection times for the TCP transport protocol. These experiments depend on the time it takes to set up a socket and the time it takes to perform a connect operation. Note that this experiment also transmits a 1-byte message and waits to receive a 1-byte message. This metric can be relevant for web servers and the World Wide Web due to their strong presence in today's Internet. The results depicted indicate that both transition mechanisms will have relatively good performance for web services.

IP Version	Number of Connections
IPv6	79
Host-to-host Encapsulation	80
Router-to-router Tunneling	76

Table 2: The number of Web Client/Server Simulation tests per second performed over IPv6, and the transition mechanisms

5 CONCLUSIONS AND FUTURE WORK

In this paper, we presented an unbiased empirical performance evaluation of IPv6, and two transition mechanisms, namely host-to-host encapsulation (6-over-4), and router-to-router tunneling (IPv6 in IPv4 tunneling), over a local area network testbed.

We summarize our major results presented in this work below.

- The Host-to-host encapsulation transition mechanism experienced as much as 66% increase in CPU utilization at the end hosts when compared to either IPv6 or the router-to-router tunneling.
- The router-to-router tunneling had very little overhead compared to the IPv6 protocol stack. There is no guarantee that as the IPv6 stack matures and its performance improves, that the overhead of the router-to-router tunneling will remain small. As transmission speeds increase, processing becomes a bottleneck, and the overhead of the router-to-router tunneling might widen its gap as the performance of IPv6 improves.
- The host-to-host encapsulation transition mechanism performed slightly better than the native IPv6 protocol stack or the router-to-router tunneling. Note that once the host has converted the IPv6 packet to an IPv4 packet, the packet is transmitted over the IPv4 infrastructure that yields relatively better performance [16] than that of IPv6; the relatively poor performance of the host-to-host encapsulation is due to the processing limitation of the host as it encapsulates/de-capsulates each packet in software. Most likely, as processing power increases on the end hosts, we will see an improvement in the performance of this transition mechanism.

Future work will continue our evaluation with more transition mechanisms in the hopes to eventually empirically evaluate all the available transition mechanisms. We also intend to investigate the performance of IPv6 when exploiting IPv6 features (such as the flow label field in the IPv6 header) to investigate end-to-end QoS support in IPv6 over IP-based networks.

REFERENCES

[1] IETF IPv6 Transition Working Group, http://www.6bone.net/ngtrans.

[2] I. Raicu. "An Empirical Analysis of Internet Protocol version 6 (IPv6)", Master Thesis, Wayne State University, 2002.

[3] Information Sciences Institute, University of Southern California, "Internet Protocol," Request for Comments 791, Internet Engineering Task Force, September 1981

[4] S. Bradner, A. Mankin, "IP: Next Generation (IPng) White Paper Solicitation," Request for Comments 1550, Internet Engineering Task Force, December 1993

[5] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," Request for Comments 1883, Internet Engineering Task Force, December 1995

[6] S. King, et al. "The Case for IPv6", Internet Draft draft-ietf-iab-case-for-ipv6-06.txt, Internet Architecture Board of Internet Engineering Task Force, December 1999, http://www.6bone.net/misc/case-for-ipv6.html.

[7] A. S. Tanenbaum, Computer Networks, Third Edition, Prentice Hall Inc., 1996, pp. 686, 413-436, 437-449

[8] B. Carpenter, C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels," Request for Comments 2529, Internet Engineering Task Force, March 1999.

[9] A. Conta, S. Deering, "Generic Packet Tunneling in IPv6 Specification," Request for Comments 2473, Internet Engineering Task Force, December 1998

[10] R. Gilligan, E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers," Request for Comments 1933, Internet Engineering Task Force, April 1996

[11] P. Srisuresh, M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations," Request for Comments 2663, Internet Engineering Task Force, August 1999

[12] S. Kent, R. Atkinson. "Security Architecture for the Internet Protocol", Request for Comments 2401, Internet Engineering Task Force, November 1998

[13] G. Tsirtsis, P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)," Request for Comments 2766, Internet Engineering Task Force, February 2000.

[14] A. Durand, P. Fasano, I. Guardini, D. Lento, "IPv6 Tunnel Broker, " Request for Comments 3053, Internet Engineering Task Force, January 2001.

[15] B. Carpenter, K. Moore, "Connection of IPv6 Domains via IPv4 Clouds," Request for

Comments 3056, Internet Engineering Task Force, February 2001.

[16] I. Raicu, S. Zeadally. "Impact of IPv6 on End-user Applications", to appear at 10th International Conference on Telecommunications, ICT'2003, Tahiti Papeete, French Polynesia, February 23, 2003.

[17] R. P. Draves, et al. "Implementing IPv6 for Windows NT". Proceedings of the 2nd USENIX Windows NT Symposium, Seattle, WA, August 3-4, 1998.

[18] P. P. Xie. "Network Protocol Performance Evaluation of IPv6 for Windows NT", Master Thesis, California Polytechnic State University, San Luis Obispo, June 1999.

[19] S. Ariga, K. Nagahashi, A. Minami, H. Esaki, J. Murai. "Performance Evaluation of Data Transmission Using IPSec over IPv6 Networks", INET 2000 Proceedings, Japan, July 18th, 2000

[20] K. K. Ettikan. "IPv6 Dual Stack Transition Technique Performance Analysis: KAME on FreeBSD as the Case", Faculty of Information Technology, Multimedia University, Jalan Multimedia, October 2000

[21] K. K., Ettikan, et al. "Application Performance Analysis in Transition Mechanism from IPv4 to IPv6". Research & Business Development Department, Faculty of Information Technology, Multimedia University (MMU), Jalan Multimedia, June 2001.

[22] Microsoft Corporation, "Microsoft IPv6 Technology Preview for Windows 2000," December 12, 2000, http://www.microsoft.com/windows2000/technologies/communications/ipv6/default.asp