International Journal of Computer and Electrical Engineering, Vol. 3, No. 2, April, 2011
1793-8163

Modeling IPv4 and IPv6 Performance in Ethernet Networks

Eric Gamess and Neudith Morales

Abstract—**IPv4 address space will run out in the next two years. Therefore, it is necessary to deploy the new version of the Internet Protocol (IPv6) which dramatically expands the address space. One important issue for IPv6 to gain acceptance, is its end-to-end performance. For this reason, it is important to offer the Internet community some tools to compare IPv4 and IPv6 performance. In this paper, we present some models to evaluate the performance of IPv4 and IPv6. Our models can be used for any variant of Ethernet technologies (10 Mbps, 100 Mbps, or 1000 Mbps). We start by presenting a model for the one-way delay in IPv4 and IPv6. We also develop an upper bound model for IPv4 and IPv6 throughput. To validate our models, we do some experiments and compare the experimental values with the values given by the models. From this research, we can conclude that IPv4 has a better performance than IPv6 (higher throughput), but the differences are small.**

Index Terms—**IPv4, IPv6, Modeling, Benchmarks, Ethernet, Performance Evaluation, Throughput, One-Way Delay.**

I. Introduction

Internet has been facing serious problems in the last few years due to the lack of adequate IPv4 address space. LACNIC [1] (Latin American and Caribbean Internet Addresses Registry) has announced the expected exhaustion of IPv4 addresses in 2011. To face the problem, several proposals have been developed and implemented. A partial and popular solution is NAT [5] (Network Address Translation) that consists of hiding networks with private IPv4 addresses behind a NAT-enabled router with few public IPv4 addresses. NAT has drawbacks since hosts behind a NAT-enabled router do not have true end-to-end connectivity and cannot participate in some Internet protocols.

Another solution to the problem of the shortage of public IPv4 addresses that faces Internet consists to migrate to the new version of the Internet protocol [3][4][10] (IPv6), or the coexistence between both protocols [1][8] (IPv4 and IPv6). IPv6 fixes a number of problems in IPv4, such as the limited number of available IPv4 addresses. IPv6 has 128-bit addresses while IPv4 has 32-bit addresses. IPv6 also adds many improvements to IPv4 in areas such as quality of service, routing, and network auto-configuration.

For IPv6 to gain acceptance, it is important that the network performance of user applications does not suffer a noticeable degradation. Some works have been proposed to compare the IPv4 and IPv6 performance (see Section II) and most of them concluded that the performance of IPv4 is a little better than the one shown by IPv6, but the difference is not significant. The majority of these works are based on benchmarks. In this paper, we present some models to evaluate the IPv4 and IPv6 performance. We also do some experiments to validate our models.

The rest of the paper is organized as follows. Related works are presented in Section II. A survey of Ethernet, IPv4 and IPv6 is made in Section III. The models to evaluate IPv4 and IPv6 performance are proposed in Section IV. We made some experiments in Section V to validate the proposed models. Conclusions and future work are discussed in Section 0.

II. Related Works

Even if IPv6 has been around for more than ten years now, just a few works have been presented to evaluate the performance of IPv6. Ettikan et al. [6][7] analyzed IPv6/IPv4 performance using simple applications (ping and FTP). They used computers with FreeBSD and the KAME[2] IPv6 protocol stack to simulate routers. They reported latency using the ping utility, and throughput using the FTP application.

Zeadally et al. [17][18] evaluated IPv6/IPv4 performance on Windows 2000 (Microsoft IPv6 Technology Preview for Windows 2000), Solaris 8 and RedHat 7.3. The authors experimentally measured throughput of TCP and UDP, latency, CPU utilization, and web-based performance characteristics.

Mohamed, Buhari and Saleem [13] evaluated IPv6/IPv4 performance on Windows 2003, FreeBSD 4.9 and RedHat 9. They measured throughput, round-trip time, socket-creation time, TCP-connection time, and number of feasible connections per second in three different testbeds. The first testbed consisted of a single computer and communication was limited to processes running in this computer using the loopback interface. In the second testbed, two computers were connected through an Ethernet hub. The Ethernet hub was replaced by a router in the third testbed. They used packets ranging from 1 byte up to the limits of an IP packet (which is typically around 65,535 bytes).

Manuscript received September 25, 2010. This work was supported in part by the CNTI (Centro Nacional de Tecnologías de Información) and the CDCH-UCV (Consejo de Desarrollo Científico y Humanístico de la Universidad Central de Venezuela) under Grant PI 03-00-6668-2007.

Eric Gamess is with the School of Computer Science, Universidad Central de Venezuela, Los Chaguaramos, Caracas 1040, Venezuela (phone: +58-212-6051296; e-mail: egamess@gmail.com).

Neudith Morales is with the DTIC (Dirección de Tecnología de Información y Comunicaciones), Universidad Central de Venezuela, Edificio El Rectorado, Los Chaguaramos, Caracas 1040, Venezuela (e-mail: neudith.morales@ucv.ve).

[1] http://www.lacnic.net

[2] http://www.kame.net

Gamess and Morales [8] setup two testbeds to evaluate IPv6/IPv4 performance stacks. In the first testbed, they connected two identical PCs with a point-to-point link to estimate the TCP and UDP throughput of Windows XP SP2, Solaris 10, and Debian 3.1, for IPv4 and IPv6. They inferred from this experiment that the IPv4 throughput is over the IPv6 throughput for both, TCP and UDP in all the three operating systems. In the second testbed, they used five identical routers. They connected two similar PCs through a chain of routers (0 to 5) to study the impact of intermediate devices over the TCP and UDP throughput for both protocols (IPv4 and IPv6) in large networks where hosts can be separated by a few intermediate layer-thee devices.

Gamess and Surós [9] presented an upper bound model for TCP and UDP throughput for IPv4 and IPv6 over Ethernet. Their model is limited to Ethernet for a full-duplex point-to-point connection. To validate the proposed model, they did some experiments and compared the maximum theoretical throughput with the experimental ones. Experiments were done with Windows XP SP2, Solaris 10, and Debian 3.1. The results showed that 10 Mbps Ethernet technology is already very mature, since it gave performance very closed to the maximum theoretical throughput. Experiments with FastEthernet (100 Mbps) showed a TCP and UDP throughput close to the maximum theoretical throughput, especially for large payload. In the case of GigabitEthernet (1000 Mbps), the differences between the model and the experiments are important.

None of the previous contributions report results for the IPv4 and IPv6 throughput or the OWD (One-Way Delay). Unlike the work of Gamess and Surós [9], in this paper we focus our research on the performance of the network layer and our models are not limited to point-to-point connections.

III. IPv4 and IPv6 over Ethernet

Our models for IPv4 and IPv6 performance are limited to Ethernet. Ethernet is the most widely available LAN technology. Different bandwidth for Ethernet had been proposed and developed. The 10 Mbps Ethernet variant (10Base2, 10Base5, 10BaseT) was the dominant technology of LANs a few years ago. Due to the immense demand for high speed networking, it has become almost obsolete and is replaced by 100 Mbps Ethernet (FastEthernet) and 1000 Mbps (GigabitEthernet).

Fig. 1 shows an Ethernet frame. The frame is composed of a header and a trailer. The header has three fields (*Destination Address*, *Source Address*, and *Type*) and is 14 bytes long. The trailer has one field (CRC) with a length of 4 bytes. It is used to detect error during transmission. The payload of Ethernet has a minimum length of 46 bytes and a maximum length of 1500 bytes, also called the MTU (Maximum Transmission Unit). If a higher layer protocol (such as IP) provides a packet that is smaller than 46 bytes, padding must be done by Ethernet to complete the minimum Ethernet data length.

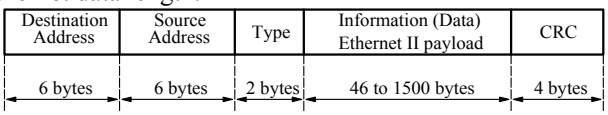

Figure 1. Ethernet frame.

The physical layer adds a header to the frame before sending it through the transmission medium. This header consists of two fields called preamble and SFD (Start Frame Delimiter). The preamble is a 56-bit (7 bytes) pattern of alternating 1 and 0 bits, which allows devices on the network to easily detect a new incoming frame and synchronize their clock with the clock of the sending device. The SFD is an 8-bit value (10101011) that separates the preamble of an Ethernet frame from the beginning of the data-link header.

Also, between two consecutive Ethernet frames, an idle period known as the IFG (Inter Frame Gap) must be present. The minimum IFG is a 96-bit times, that is the time necessary to transmit 96 bits (12 bytes) of raw data on the medium. In half-duplex Ethernet, collisions can occur and stations must retransmit the frame. The timing of retransmission is typically controlled using a retransmitting algorithm such as a Binary Exponential Backoff (BEB) algorithm. The BEB algorithm generally specifies "2" as a common small constant factor by which the time between making retransmission attempts is multiplied for each subsequent attempt. In full-duplex and switched Ethernet, there are no collisions, so there is no need of a BEB algorithm.

Fig. 2 shows the IPv4 header as defined in [16]. Since the header has a variable length (depending on options), the field IHL (Internet Header Length) is used to specify its length. IHL is a field of 4 bits and must be multiplied by 4 to compute the actual length of the IPv4 header in bytes. The minimum header length is 20 bytes (IHL=5), and represents an IPv4 packet without options. *Total Length* (16 bits) is the total length of the packet (IPv4 header length + IPv4 data length). This field allows the length of an IPv4 packet to be up to 65,535 bytes (including the header). That is, up to 65,515 bytes (65,515=65,535-20) of IPv4 data in a packet without options.

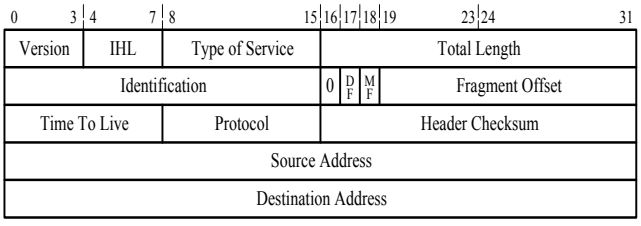

Figure 2. IPv4 header.

Fig. 3 shows the header of IPv6 as defined in [4]. The header length is fixed to 40 bytes. So the field IHL defined in IPv4 was eliminated in IPv6. Unlike IPv4, IPv6 manages options through extension headers (e.g. Hop-by-Hop Options Header, Routing Header, Fragment Header, Destination Options Header, Authentication Header, Encapsulating Security Payload Header) which are inserted between the IPv6 header and the upper layer protocol header. The field *Total Length* defined in IPv4 is now named *Payload Length* in IPv6. *Payload Length* represents the length of the IPv6 option headers and IPv6 data (the length of the rest of the packet that follows the IPv6 header). That is, *Payload Length* does not include the length of the IPv6 header (40 bytes). So an IPv6 packet can have up to 65,535 bytes of data when it does not have options.

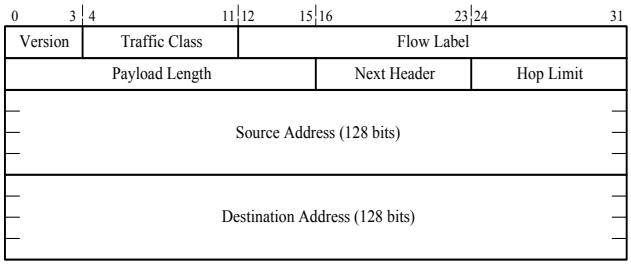

Figure 3. IPv6 header.

The fragment header (see Fig. 4) is used by an IPv6 source to send a packet larger than the path MTU [12] to its destination. Unlike IPv4, fragmentation in IPv6 is performed only by source nodes, not by routers along a packet's delivery path. When fragmentation is necessary, a fragment header is placed between the IPv6 header and the upper layer header. The fragment header is identified by a *Next Header* value of 44 in the immediately preceding header. The length of the fragment header is 8 bytes long.

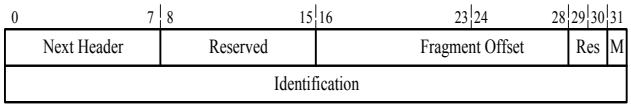

Figure 4. Fragment header.

IV. Modeling the IPv4 and IPv6 Performance

To compute the OWD (One-Way Delay), which can be considered as half the RTT (Round-Trip Time) in symmetric networks, we must consider (1) the processing time, (2) the serialization time, and (3) the propagation time. The processing time is the overhead introduced by the sender and the receiver to handle the frame for transmission and reception. Note that when a packet has to pass through intermediate devices (e.g. routers) in its way, they also introduce processing time. The serialization time is the amount of time the sender takes to emit all bits into the medium. It can be calculated by dividing the number of bits transmitted by the bandwidth. The propagation time is the amount of time it takes a bit to traverse the link. It can be calculated by dividing the length of the medium by its propagation speed (propagation speed depends on the physical medium).

Let call *TTFS(n,B)*, the minimum time necessary for the serialization of *n* bytes of IP payload with a bandwidth of *B*. In the following subsections, we give the formula of *TTFS(n,B)* for IPv4 and IPv6. TTFS stands for Total Time For Serialization. *B* is in bps (bits per second).

A. Total Time for Serialization in IPv4

For reasons of space, we will only present the steps to compute *TTFS(n,B)* for a single IPv4 packet.

In the best case, the IPv4 header will be 20 bytes long (when there are no options). Since the Ethernet MTU is 1500 bytes, the maximum IPv4 data in one packet is 1480 bytes (1480=1500-20). Also, for small values of n ($1 \le n \le 26$), padding will be necessary to fulfill the minimum Ethernet data size (46 bytes). So we can divide the transmission of one IPv4 packet in two cases:

First case: If $n \in \{1,\ldots,26\}$, Ethernet padding will be necessary. So to send *n* bytes of IPv4 data (IPv4 payload), the minimum number of necessary bytes is:

IFG	12 bytes
Preamble	7 bytes
SFD	1 byte
Ethernet	18 bytes
IPv4	20 bytes
Data	*n* bytes
Padding	26-*n* bytes
Total:	84 bytes

Then, for this case: $TTFS(n,B) = \frac{84 \times 8}{B}$.

Second case: If $n \in \{26,\ldots,1480\}$, no Ethernet padding is necessary. So to send *n* bytes of IPv4 data (IPv4 payload), the minimum number of necessary bytes is:

IFG	12 bytes
Preamble	7 bytes
SFD	1 byte
Ethernet	18 bytes
IPv4	20 bytes
Data	*n* bytes
Total:	*n*+58 bytes

Then, for this case: $TTFS(n,B) = \frac{(n+58) \times 8}{B}$.

We can show that *TTFS(n,B)* can be generalized to any value of n ($1 \le n \le 65515$). The general formula is (1) when padding is necessary for the last fragment and (2) without padding in the last segment. $E(x)$ is the function that returns the integer portion of its argument.

If $1 + 1480a \le n \le 26 + 1480a$ then:

$$TTFS(n,B) = \frac{\left(1538.E\left(\frac{n-1}{1480}\right) + 84\right) \times 8}{B} \quad (1)$$

If $26 + 1480a \le n \le 1480 + 1480a$ then:

$$TTFS(n,B) = \frac{\left(58.E\left(\frac{n-26}{1480}\right) + n + 58\right) \times 8}{B} \quad (2)$$

B. Total Time for Serialization in IPv6

For reasons of space, we will only present the steps to compute *TTFS(n,B)* for a single IPv6 packet.

In the best case, the IPv6 header will be 40 bytes long (when there are no options). Since the Ethernet MTU is 1500 bytes, the maximum IPv6 data in one packet is 1460 bytes (1460=1500-40). Also, for small values of n ($1 \le n \le 6$), padding will be necessary to fulfill the minimum Ethernet data size (46 bytes). So we can divide the transmission of one IPv6 packet in two cases:

First case: If $n \in \{1,\ldots,6\}$, Ethernet padding will be necessary. So to send *n* bytes of IPv6 data (IPv6 payload), the minimum number of necessary bytes is:

IFG	12 bytes
Preamble	7 bytes
SFD	1 byte
Ethernet	18 bytes
IPv6	40 bytes

Data	n bytes
Padding	6-n bytes
Total:	84 bytes

Then, for this case: $TTFS(n,B)=\frac{84\times 8}{B}$.

Second case: If $n\in\{6,\ldots,1460\}$, no Ethernet padding is necessary. So to send n bytes of IPv6 data (IPv6 payload), the minimum number of necessary bytes is:

IFG	12 bytes
Preamble	7 bytes
SFD	1 byte
Ethernet	18 bytes
IPv6	40 bytes
Data	n bytes
Total:	n+78 bytes

Then, for this case: $TTFS(n,B)=\frac{(n+78)\times 8}{B}$.

We can show that *TTFS(n,B)* can be generalized to any value of n $(1\le n\le 65535)$. The general formula is divided in 4 cases. Note that (3) is the only case where padding is done. That is, there will be no padding in the fragments that result in sending an IPv6 packet with a payload greater than 1460 bytes, since the necessary headers (IPv6 header + Fragment header) will occupy 48 bytes, which is over the minimum to do padding (46 bytes).

If $1\le n\le 6$ then:

$$TTFS(n,B)=\frac{84\times 8}{B} \tag{3}$$

If $6\le n\le 1460$ then:

$$TTFS(n,B)=\frac{(n+78)\times 8}{B} \tag{4}$$

If $1461\le n\le 2900$ then:

$$TTFS(n,B)=\frac{(n+172)\times 8}{B} \tag{5}$$

If $n\ge 2900$ then:

$$TTFS(n,B)=\frac{\left(86E\left(\frac{n-5}{1448}\right)+n+86\right)\times 8}{B} \tag{6}$$

C. One-Way Delay Model

To compute the OWD (One-Way Delay) in a general case, we must consider two PCs connected through a chain of routers as depicted in Fig. 5. All connections are variants of Ethernet technologies, that is, all the links between the two PCs must be: Ethernet, FastEthernet, and GigabitEthernet.

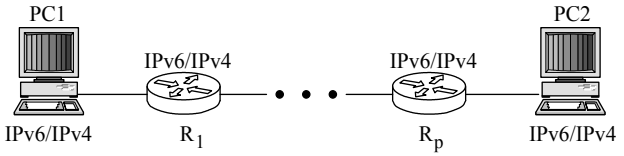

Figure 5. Connection of 2 PCs through a chain of routers.

Let denote $\pi(X, Y)$, the propagation time between X and Y, and $\phi(X)$ the processing time (overhead) introduced by X to process the packet. For a chain of p routers, we can infer that the OWD to send n bytes of IP payload from *PC1* to *PC2* is:

$$\begin{aligned} OWD(n) \;=\; & \phi(PC1)+TTFS(n,BPC1)+\pi(PC1,R1)\\ & +\phi(R1)+\Delta\big(TTFS(n,BR1)\big)+\pi(R1,R2)+\\ & +\phi(R2)+\Delta\big(TTFS(n,BR2)\big)+\pi(R2,R3)+\\ & \ldots\\ & +\phi(Rp)+\Delta\big(TTFS(n,BRp)\big)+\pi(Rp,PC2)\\ & +\phi(PC2)\end{aligned}$$

Where $\Delta(TTFS(n,B))$ is the time for the serialization of the fragments that are still in a device when the last fragment is received in this device. *BPC1* is the bandwidth of *PC1*, *BR1* is the bandwidth used by *R1* to send the data, *BR2* is the bandwidth used by *R2* to send the data, and so on. For small values of n (when fragmentation is not needed), Δ will be function identity.

D. Maximum Theoretical Throughput Model

The throughput is the amount of data transferred from one place to another in a specified amount of time. Typically, throughputs are measured in kbps, Mbps and Gbps. The maximum throughput that can be archived is:

$$throughput(n)=\frac{n\times 8}{TTFS(n,B_{\min})} \tag{7}$$

where $B_{\min}=\min(BPC1,BR1,BR2,\cdots,BRp)$, that is the minimum bandwidth of the path. The throughput of (7) can be obtained in an ideal network where the hypothetical devices have a processing time of 0 s. That is, devices where the overhead introduced to handle the packet for transmission and reception is 0 s. Even if such devices do not exist, the maximum theoretical throughput of IPv6/IPv4 is very useful since it can be compared to the experimental throughput.

1) Maximum Theoretical Throughput of IPv4 and IPv6 with Ethernet Technology

From (7), we can infer that the maximum theoretical throughput between two network devices that are separated by Ethernet, FastEthernet, and GigabitEthernet networks will be:

$$throughput(n)=\frac{n\times 8}{TTFS(n,10^7)} \tag{8}$$

since the minimum bandwidth in the network path is 10 Mbps (i.e. 10^7 bps). Fig. 6 shows the maximum theoretical throughput for IPv4 and IPv6 obtained in this case. We can observe that the maximum theoretical throughput of IPv4 (upper curve) is superior to the one shown by IPv6 (lower curve). This small difference is due to the header of IP (20 bytes in IPv4, and 40 bytes in IPv6).

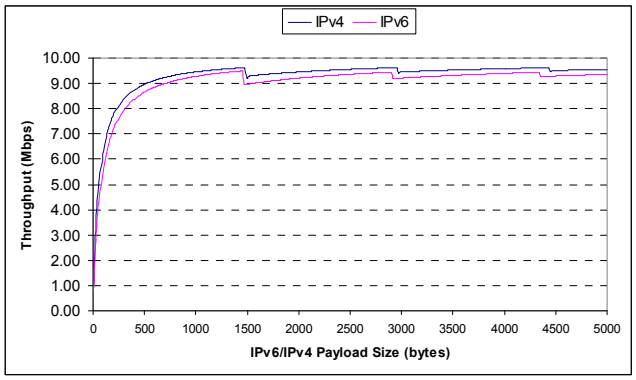

Figure 6. Maximum theoretical throughput with Ethernet.

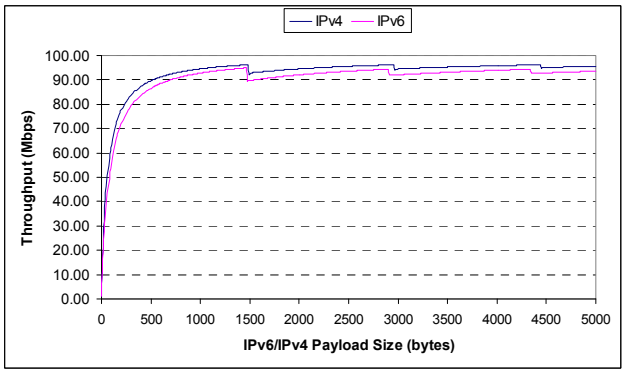

Figure 7. Maximum theoretical throughput with FastEthernet.

2) Interpreting the Maximum Theoretical Throughput of IPv4 with Ethernet Technology

From Fig. 6, we can infer that the maximum theoretical throughput of IPv4 increases with the size of the payload from 1 to 1480 bytes. For 1481 bytes, this maximum theoretical throughput falls and starts to increase again until the payload reaches 2960 bytes long. This pattern will appear every 1480 bytes. Let's give an explanation of this behavior. To send an IPv4 packet with a payload length less than or equal to 1480 bytes, a single IPv4 packet (without fragmentation) is necessary. The first fall is due to the necessity to divide the IPv4 packet in 2 fragments (we will have 2 fragments with an IPv4 payload between 1481 and 2960 bytes). The second fall is due to the necessity to divide the IPv4 packet in 3 segments, and so on every 1480 bytes. The maximum of the function is:

$$throughput(1480) = throughput(2960) = \cdots = \frac{1480}{1538} \times 10^7 \approx 9.62\ Mbps$$

So 9.62 Mbps is the maximum throughput of IPv4 for any length of payload if the path between the 2 devices has Ethernet technology.

3) Interpreting the Maximum Theoretical Throughput of IPv6 with Ethernet Technology

From Fig. 6, we can infer that the maximum theoretical throughput of IPv6 increases with the size of the payload from 1 to 1460 bytes. For 1461 bytes, this maximum theoretical throughput falls and starts to increase again until the payload reaches 2900 bytes long. For 2901 bytes, this maximum theoretical throughput falls and starts to increase again until the payload reaches 4348 bytes long. This pattern will appear every 1448 bytes. The explanation is similar to IPv4 and is due to fragmentation. The maximum of the function is obtained for 1460 bytes and its value is:

$$throughput(1460) = \frac{1460}{1538} \times 10^7 \approx 9.49\ Mbps$$

So 9.49 Mbps is the maximum throughput of IPv6 for any length of payload if the path between the 2 devices has Ethernet technology.

4) Maximum Theoretical Throughput of IPv4 and IPv6 with FastEthernet Technology

Fig. 7 shows the maximum theoretical throughput for IPv4 and IPv6 obtained when the two network devices are separated by FastEthernet and GigabitEthernet networks. In this case, the minimum bandwidth in the path followed by packets will be 100 Mbps. We can observe that the maximum theoretical throughput of IPv4 (upper curve) is superior to the one shown by IPv6 (lower curve). This small difference is due to the header of IP (20 bytes in IPv4, and 40 bytes in IPv6). The complete interpretation of Fig. 7 is similar to the one done for Fig. 6, that is, the falls are due to fragmentation.

V. Validation of the Models

We start the verification of our models with the validation of the throughput. To do so, we setup a testbed consisted of two identical PCs connected by a point-to-point link (see Fig. 8) with a crossover cable. Both PCs were equipped with a dual AMD Opteron 246 (2 GHz), 2 GB of RAM, a 72 GB hard drive, and a full-duplex 10/100/1000 Mbps PCI Ethernet adapter. In each PC, we installed Debian 5.0 (Lenny) with kernel version 2.6.26. We also setup both network protocol (IPv4 and IPv6).

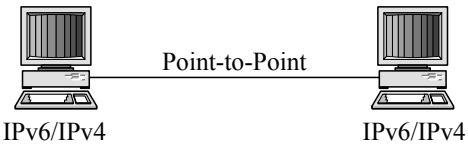

Figure 8. Testbed with of 2 PCs connected with a point-to-point link.

We did our own benchmark using the C programming language to measure the throughput. The benchmark is based on the client/server model. Basically, the client sends a specific number of packets of the same size at a specific rate to the server and the server counts the number of packets received. If the number of received packets is less than the number of sent packets, we are over the maximum throughput. So we reduce the transmission rate and rerun the test. The throughput is the fastest rate at which the number of packets transmitted by the client is equal to the number of packets received by the server. The experiment was repeated to get consistent measurements. We compiled the source code of the benchmark with the GNU C compiler (gcc) version 4.3.2. TABLE I shows the experimental (columns labeled with *Exp*) and the maximum theoretical (columns labeled with *Max*) throughput for IPv4 in the different technologies: Ethernet, FastEthernet, and GigabitEthernet. We can infer that our model is very closed to the reality,

excepting for small IPv4 payload sizes in GigabitEthernet. We can also observe the falls at 1481 and 2961 bytes due to fragmentation.

TABLE I: EXPERIMENTAL AND MAXIMUM THROUGHPUT FOR IPV4 (MBPS).

Payload (bytes)	Ethernet		FastEthernet		GigabitEthernet	
	Exp	Max	Exp	Max	Exp	Max
10	1.19	1.19	11.90	11.90	30.70	119.05
100	6.32	6.33	63.28	63.29	297.69	632.91
500	8.96	8.96	89.60	89.61	895.65	896.06
1000	9.44	9.45	94.50	94.52	945.10	945.18
1480	9.62	9.62	96.23	96.23	961.78	962.29
1481	9.12	9.13	91.30	91.31	912.99	913.07
1500	9.25	9.25	92.45	92.48	923.12	924.78
2000	9.44	9.45	94.51	94.52	945.12	945.18
2960	9.59	9.62	96.23	96.23	961.97	962.29
2961	9.36	9.37	93.68	93.70	936.95	937.03

TABLE II shows the experimental (columns labeled with *Exp*) and the maximum theoretical (columns labeled with *Max*) throughput for IPv6 in the different technologies: Ethernet, FastEthernet, and GigabitEthernet. We can infer that our model is very closed to the reality, excepting for small IPv6 payload sizes in GigabitEthernet. We can also observe the falls at 1461 and 2901 bytes due to fragmentation.

TABLE II: EXPERIMENTAL AND MAXIMUM THROUGHPUT FOR IPV6 (MBPS).

Payload (bytes)	Ethernet		FastEthernet		GigabitEthernet	
	Exp	Max	Exp	Max	Exp	Max
10	1.09	1.14	11.32	11.36	30.05	113.64
100	5.58	5.62	56.17	56.18	295.86	561.80
500	8.62	8.65	86.49	86.51	864.67	865.05
1000	9.25	9.28	92.73	92.76	927.63	927.64
1460	9.47	9.49	94.93	94.93	949.28	949.28
1461	8.92	8.95	89.46	89.47	894.25	894.67
1500	8.95	8.97	89.69	89.71	897.11	897.13
2000	9.17	9.21	92.06	92.08	920.79	920.81
2900	9.44	9.44	94.38	94.40	944.00	944.01
2901	9.17	9.18	91.82	91.83	918.25	918.33

For the verification of the OWD, some assumptions are used to simplify a complex model. Let consider that the processing time (ϕ) introduced by a device to handle the packet is negligible (we can make this assumption only in underloaded devices). So, the OWD for a uniform network (a network with a unique variant of Ethernet technology) of p routers (see Fig. 5) with connections of the same length (the propagation time is identical in all links) can be modeled as:

$$OWD(n) = TTFS(n,B) + p \times TTFS(m,B) + \pi \times (p+1) \quad (9)$$

where m is the size of the payload of the last fragment. In this hypothetical case, when the last fragment arrives in a router, the previous fragments have already been sent; so serialization is reduced to the serialization of the last fragment. Since there is no control over the order in which the sender is going to transmit the fragments, we will limit us to validate the OWD model for a single packet; in other words, limited to 1480 bytes of payload in IPv4 and 1460 bytes of payload in IPv6. In this case, (9) can be rewritten as:

$$OWD(n) = TTFS(n,B) \times (p+1) + \pi \times (p+1) \quad (10)$$

We did our own benchmarks using the C programming language to measure the OWD (One-Way Delay) for IPv4 and IPv6. The benchmarks are based on the client/server model. Basically, an IP packet (IPv4 or IPv6) of a fixed length is exchanged between the server and the client a number of times. We take a timestamp before and after the interchange. The difference of the timestamps is divided by the number of time the message was sent and received and by 2 to get the average OWD. The experiment was repeated to get consistent measurements.

Fig. 5 shows the testbed that we made for the experiments with the OWD. We connected two PCs (the same PCs of the first testbed) through a chain of routers (0 to 5). To do so, we used five identical routers (Cisco Systems 2811) with 256 MB of RAM. Each router had two FastEthernet interfaces (10/100 Mbps) that can be statically configured to 10 Mbps (command: *speed 10*) or 100 Mbps (command: *speed 100*). In each router, we installed Cisco IOS (Internetwork Operating System) version 12.4(9)T that has support for IPv6. We enabled CEF (Cisco Express Forwarding) in the routers; CEF is mainly used to improve packet switching speed, reducing the overhead and delays introduced by other routing techniques, increasing overall performance. We used category 5E UTP cables of 4 meters. The idea of this experiment is to study the impact of intermediate devices over the OWD.

TABLE III shows the results of the experiment for IPv4 using Ethernet technology. The first column represents the size of the IPv4 payload in bytes. The second column includes the OWD (experimental values are in the non-shaded row and values from our model are in the shaded row) for a point-to-point connection (0 routers). The third column includes the OWD for a connection through 1 router. The fourth column includes the OWD for a connection through 2 routers, and so on. OWD are giving is microseconds. For our model, we used a propagation time of 30 μs (π=30 μs) which was obtained experimentally.

TABLE III: OWD IN μS USING IPV4 AND ETHERNET TECHNOLOGY.

Payload (bytes)	Number of routers in the chain					
	0	**1**	**2**	**3**	**4**	**5**
10	104	184	275	362	446	533
	97	194	292	389	486	583
100	164	310	458	605	751	902
	156	313	469	626	782	938
400	406	796	1182	1580	1973	2367
	396	793	1189	1586	1982	2378
700	650	1291	1913	2552	3189	3831
	636	1273	1909	2546	3182	3818
1000	890	1764	2644	3526	4406	5294
	876	1753	2629	3506	4382	5258
1480	1288	2539	3790	5079	6350	7629
	1260	2521	3781	5042	6302	7562

TABLE IV shows the results of the experiment for IPv4 using FastEthernet technology. Values obtained with our model are very closed to the experimental values.

TABLE IV: OWD IN μS USING IPV4 AND FASTETHERNET TECHNOLOGY.

Payload (bytes)	Number of routers in the chain					
	0	**1**	**2**	**3**	**4**	**5**
10	40	69	101	135	169	199
	37	73	110	147	184	220
100	42	77	118	158	197	238
	43	85	128	171	213	256
400	72	131	188	289	300	367
	67	133	200	267	333	400

700	99	183	266	347	427	513
	91	181	272	363	453	544
1000	120	239	348	456	558	670
	115	229	344	459	573	688
1480	165	322	471	645	796	936
	153	306	459	612	765	918

TABLE V shows the results of the experiment for IPv6 using Ethernet technology. Values obtained with our model are very closed to the experimental values. Also, we can see that the OWD for IPv6 is a little over the one obtained in IPv4 (see TABLE III and TABLE V). This small difference is due to the header of IP (20 bytes in IPv4, and 40 bytes in IPv6).

TABLE V: OWD IN μS USING IPV6 AND ETHERNET TECHNOLOGY.

Payload (bytes)	Number of routers in the chain					
	0	**1**	**2**	**3**	**4**	**5**
10	109	206	313	405	512	617
	100	201	301	402	502	602
100	182	338	522	700	881	1002
	172	345	517	690	862	1034
400	430	841	1254	1662	2064	2487
	412	825	1237	1650	2062	2474
700	672	1342	1937	2636	3275	3933
	652	1305	1957	2610	3262	3914
1000	897	1774	2689	3582	4472	5322
	892	1785	2677	3570	4462	5354
1460	1298	2541	3785	5134	6322	7581
	1260	2521	3781	5042	6302	7562

TABLE VI shows the results of the experiment for IPv6 using FastEthernet technology. In this case also, values obtained with our model are very closed to the experimental values. Additionally, we can see that the OWD for IPv6 is a little over the one obtained in IPv4 (see TABLE IV and TABLE VI). This small difference is due to the header of IP (20 bytes in IPv4, and 40 bytes in IPv6).

TABLE VI: OWD IN μS USING IPV6 AND FASTETHERNET TECHNOLOGY.

Payload (bytes)	Number of routers in the chain					
	0	**1**	**2**	**3**	**4**	**5**
10	40	81	116	162	193	242
	37	74	111	148	185	222
100	45	82	135	193	240	283
	44	88	133	177	221	265
400	73	143	211	292	337	421
	68	136	205	273	341	409
700	101	193	289	381	474	545
	92	184	277	369	461	553
1000	124	251	317	492	599	731
	116	232	349	465	581	697
1460	168	331	472	645	776	937
	153	306	459	612	765	918

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we developed some models to evaluate the IPv4 and IPv6 performance for networks based in Ethernet technologies. To validate our models, we made some experiments and obtained experimental results very similar to the one given by our models.

Our experiments and our models show that the IPv4 performance is a little better that its equivalent in IPv6 (IPv4 has a better throughput and lower OWD). This is due to the difference in the IP headers (20 bytes for IPv4 and 40 bytes for IPv6), even if IPv4 has to compute the checksum which was eliminated in IPv6. By simplifying our models (ideal case), we inferred the best performance than can be archived in an Ethernet network for an end-to-end connection (connection between computers separated by several intermediate layer-thee devices). So our models give researchers and network administrators insights of the best performance that can be archived by an Ethernet network, that is a network with little load. These results can be used to model an upper-bound of the network performance of an application.

We plan to extend our models to other technologies such as IEEE 802.11 (WiFi) and MPLS (Multiprotocol Label Switching). We also want to further investigate the performance of transition mechanisms [14] such as 6to4 [2] and Teredo [11], which constitute a necessary step toward IPv6's full deployment.

REFERENCES

[1] M. Blanchet. Migrating to IPv6: A Practical Guide to Implementing IPv6 in Mobile and Fixed Networks. John Wiley & Sons, First Edition. January 2006.
[2] B. Carpenter and K. Moore. Connection of IPv6 Domains via IPv4 Clouds. RFC 3056. February 2001.
[3] J. Davies. Understanding IPv6. Microsoft Press, Second Edition. January 2008.
[4] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460. December 1998.
[5] B. Dutcher. The NAT Handbook: Implementing and Managing Network Address Translation. John Wiley & Sons, First Edition. January 2001.
[6] K. Ettikan. IPv6 Dual Stack Transition Technique Performance Analysis: KAME on FreeBSD as the Case. Technical report, Faculty of Information Technology, Multimedia Univ., Jalan Multimedia. October 2000.
[7] K. Ettikan, K. Gopi, and Y. Takefumi. Application Performance Analysis in Transition Mechanism from IPv4 to IPv6. IWS2000. Japan. February 2000.
[8] E. Gamess and N. Morales. Implementing IPv6 at Central University of Venezuela. In Proceedings of LANC'07 (the 4th International IFIP/ACM Latin America Networking Conference). San José, Costa Rica. October 2007.
[9] E. Gamess and R. Surós. An Upper Bound Model for TCP and UDP Throughput in IPv4 and IPv6. Journal of Network and Computer Applications 31, pp 585–602. 2008.
[10] S. Hagen. IPv6 Essentials. O'Reilly, Second Edition. May 2006.
[11] C. Huitema. Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs). RFC 4380. February 2006.
[12] J. Mogul and S. Deering. Path MTU Discovery. RFC 1191. November 1990.
[13] S. Mohamed, M. Buhari, and H. Saleem. Performance Comparison of Packet Transmission over IPv6 Network on Different Platforms. IEE Proceedings Communications 153(3): 425-433. June 2006.
[14] E. Nordmark and R. Gilligan. Basic Transition Mechanisms for IPv6 Hosts and Routers. RFC 4213. October 2005.
[15] C. Popoviciu, E. Levy-Abegnoli, and P. Grossetete. Deploying IPv6 Networks. Cisco Press. First Edition. February 2006.
[16] J. Postel. Internet Protocol. RFC 791. September 1981.
[17] S. Zeadally and I. Raicu. Evaluating IPv6 on Windows and Solaris. IEEE Internet Computing 7(3): 51-57. May 2003.
[18] S. Zeadally, R. Wasseem, and I. Raicu. Comparison of End-System IPv6 Protocol Stacks. IEE Proceedings Communications 151(3): 238-242.May2004.