PERFORMANCE EVALUATION OF LAYER 4 SWITCHING IN IPv6 VERSUS IPv4

Daniel ZINCA, Virgil DOBROTA, Cristian Mihai VANCEA
Technical University of Cluj-Napoca, Department of Communications,
26 Baritiu Street, 3400 Cluj-Napoca, Romania, Tel/Fax: +40-64-197083
e-mail: {Daniel.Zinca, Virgil.Dobrota, Mihai.Vancea}@com.utcluj.ro

Abstract: This paper is focused on a Layer 4 switching experiments of IPv6 over Fast Ethernet, running under Windows 2000 Professional. Prior to our studies on IPv4 over ATM with TCP relaying, firstly introduced at IEEE LANMAN'99, we are trying to evaluate the performances at the interface between the applications and the nonblocking stream-oriented sockets in TCP/IP. The first major objective is to get consistent results for a IPv6 versus IPv4 debate, even the implementation phase of this new version of the Internet Protocol is under progress. The second objective is to propose a method for measuring the round-trip-time RTT at the Application Layer. It will provide 1 nanosecond - accuracy, that is requested by the new technologies such as Gigabit Ethernet.

Keywords: *burst traffic, Fast-Ethernet, Gigabit Ethernet, IPv4, IPv6, Layer 4 switching, NTP, SNTP, TCP/IP*

I. INTRODUCTION

To help the industry-wide conversion from IP version 4 (IPv4) to IP version 6 (IPv6), Microsoft has announced a four-phase execution plan. The current phase (delivery of a preview IPv6 stack for applications conversion) will be followed by the delivery of a pre-production version for laboratory testing and, finally, the production release to be deployed [9].

All the experiments, concerning both IPv6 and IPv4 presented herein, involved Fast Ethernet technology and Windows 2000 Professional. An updated version of the software tool from [2],[3] offered the facilities to evaluate the sending time, the receiving time and the elapsed time at the interface between the application and the non-blocking stream-oriented socket in TCP/IP. Obviously each Layer 3 protocol version requested its own TCP implementation on top of it. Some of the preliminary results published herein were previously presented in [1].

Although the types of models applied have a great influence on the overall performances, it is not the subject of this paper to present the reasons to choose them. The reader is kindly advised to read [3],[4] for more details.

To understand the experiments carried out by comparing IPv6 implementation to currently running IPv4, under the same physical and data link testing conditions, a short note is necessary.

According to the approved commentaries regarding the OSI Reference Model, transport relays could not guarantee the transport service, except under very constrained circumstances [8]. Therefore it was generally agreed that the Layer 4 switching is prohibited. Despite of this statement, the results presented in [2],[3] demonstrated that the TCP/IP environment should be exploited by involving scheduling and relaying mechanisms even at the transport layer.

Recently, Cisco Systems Inc. has introduced its own concept of Layer 4 switching, rather different from that one we are using in this paper. They have implemented a Server Load Balancing (SLB) over Layer 3 switching for their Fast-Ethernet/Gigabit-Ethernet Catalyst 4840G. Cisco's Layer 4 switch is a re-distributor of the requests and hits from clients evenly among all the server in the server farm, in order to achieve a balanced load for each server. It was mainly designed for increasing Web traffic and access reliability of multiple Web servers, offering the appearance of one virtual server, with one IP address and a single Universal Resource Locator (URL) for an entire server farm [10].

In this paper, we are trying to obtain better results in TCP/IP for burst traffic by involving departure schedules for TPDUs (Transport Protocol Data Units). This means that the applications should not send the information directly to the sockets without taking into account the non-linear behaviour of the TCP/IP entities within a broadband network. On the other hand, at the server site, a pure Layer 4 switching will be performed, by redistributing the TPDU from the incoming socket to the outgoing socket, as faster as possible, without any additional scheduling or checking. As soon as the optimum model, i.e. a frame departure schedule, will be determined for a given application, under a given network, it is for sure that a Layer 4 switching schedule (or at least a QoS mechanism) has to be added at the server site, too.

II. TESTING CONFIGURATION AND FILES

Due to the fact that Microsoft's IPv6 implementation is based on Windows 2000 technology only and the available ATM cards drivers (VIRATAlink) were written for Windows95/98/NT only, we were forced to perform the experiments on Fast Ethernet, instead of ATM.

Let us suppose again, as in [3], the most favorable networking conditions, i.e. there will be no other workstations connected, except those involved in trial. The entire bandwidth is at our disposal, without unexpected collisions or congestion. Therefore the results presented herein could be considered as the maximum we can get from the network.

The testing configuration in *Figure 1* included three workstations connected to the 100 Mbps ports of HP ProCurve hub. The most powerful station within the tested network was based on Intel's Pentium II/400 MHz, running the server and acting as a Layer 4 switch. The client software was installed on two different workstations (with Celeron 366 MHz and Pentium 233 MHz MMX). Note that by the time this experiments were done, more powerful machines would have been available, but it was decided to keep the same testing configuration as in [2],[3] for IPv4 over ATM. The *Testfile1* has 7990 bytes, whilst *Testfile 2* has 240,118 bytes.

The evaluation's accuracy of the proposed software tool (client and server) is given by the clock period of the CPU (2.5 ns. at Pentium II/400 MHz). The measurement is also dependent on RDTSC *(Read Time Stamp Counter)* and other instructions included in the loop. Obviously the processes are guided by the TCP/IP entity, as we rely on the Windows Sockets *select* function to determine the status of the sockets and to perform synchronous I/O [3].

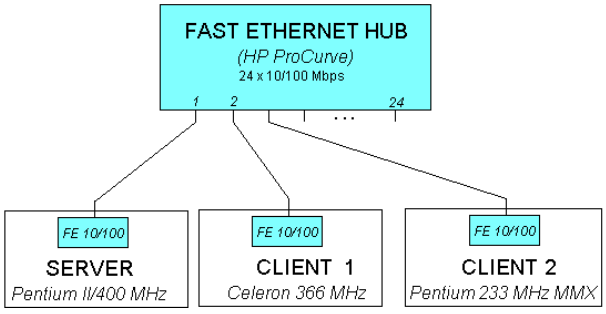

Figure 1. The testing configuration

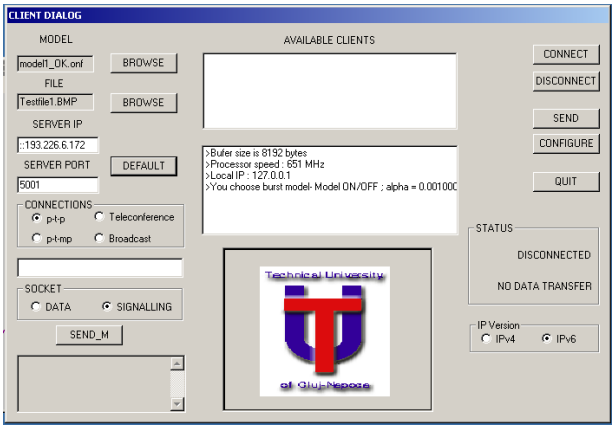

Figure 2. Screen capture of the client's GUI used as Testfile2 (240,118 bytes)

III. PROTOCOL DESIGN FOR MEASURING ROUND-TRIP-TIME IN APPLICATION LAYER

The time synchronization protocols currently in use for TCP/IP are NTP *(Network Time Protocol)* and its simplified version, SNTP *(Simple Network Time Protocol)*. They involve two synchronization processes. The first is the synchronization with a reference time device (cesium clock, GPS receiver, etc.), offering a theoretical accuracy of less than 1 nanosecond. The second process is the synchronization between the external reference and the computer's internal clock (using standard operating system calls). Unfortunately the existing implementations are providing about 1 millisecond accuracy only, determining the overall value. Note that NTP is using absolute time synchronization between servers and clients in order to distribute time across the Internet [5]. In many other applications, we need rather a relative synchronization between the hosts. Therefore in this paragraph we are proposing two methods for improving the overall accuracy used for the timestamps. The first method is devoted to SNTP and the second one is an original proposal of a new protocol. Both of them are based on Intel's microprocessors family and RDTSC instruction [6]. The number of clock periods since the computer was powered on is the resulting 64-bit timestamp.

The first method calls the RDTSC instruction, the timestamp being multiplied by the microprocessor's clock. The least significant 32 bits represent subdivisions of the current second, to be used by SNTP.

The second method envisaged describes a new protocol for measuring the RTT *(Round Trip Time)* at the Application Layer. As in [7], no corrections are performed against the local clocks. The timestamps are used to synchronize the sending moments according to the packet delays on the network at that time. The main requirements of this protocol are the following:

- accuracy of hundreds of picosecond up to 1 nanosecond, depending on the microprocessor's clock
- minimum packet size of the packet will accommodate several technologies, including ATM
- IPv4/IPv6, TCP/UDP support
- multicast support
- socket-based implementation (Microsoft Windows/ Linux/UNIX operating systems)
- minimum time between timestamp reading operations and network operations
- minimum time to fill-in the fields

Byte 0	*Bytes 1 … 8*	*Bytes 9 and 10*	*Bytes 11 … 18*	*Bytes 19 and 20*	*Bytes 21 … 24*	*(Bytes 25 … 1023)*
Type	**Sending Timestamp**	**Sending Sequence Number**	**Receiving Timestamp**	**Receiving Sequence Number**	**Sending Processor's Frequency**	**PAD (optional)**
(8 bits)	**(64 bits)**	**(16 bits)**	**(64 bits)**	**(16 bits)**	**(32 bits)**	**(0 or 7992 bits)**

Figure 3. Application Protocol Data Unit (APDU) determining RTT at Layer 7

This protocol is carried out on the signalling socket, involving an exchange of APDUs, with the following significance of the fields presented in *Figure 3*:

- *Type* (8 bits) is similar to the field *Stratum* used by NTP. It also specifies a multicast (MSB=1) or unicast (MSB=0) transmission
- *Sending Timestamp* (64 bits) represents the timestamp before starting the sending of the packet (as ts_{12_i} in *Figure 4*) to the corresponding client
- *Sending Sequence Number (SSN)* (16 bits) represents a sequence number used by the transmitter The UDP-based applications use this value, whilst the TCP-based ones may ignore it.
- *Receiving Timestamp* (64 bits) represents the timestamp before starting the receiving of the last packet (as ts_{12_f} in *Figure 4*) from the corresponding client
- *Receiving Sequence Number (RSN)* (16 bits) represents the sequence number of the last received packet from the corresponding client. The UDP-based applications use this value, whilst the TCP-based ones may ignore it.
- *Sending Processor's Frequency* (32 bits) is given in MHz.
- PAD (0 or 7992 bits) represents additional 7992 bits of 00h for Ethernet frame-based technologies because the 1024-byte APDU gives a more accurate evaluation of the overall RTT throughput. The 25-byte APDU (0 bits of PAD) is recommended for technologies such as ATM.

In multicast transmissions, the server sends only the first three fields, because its microprocessor's frequency is known from previous messages.

There are two memory tables for each host that uses this protocol. The first table is dedicated to the sent messages and stores the timestamps: ts_{12_i} (initial moment of the sending from client 1 to client 2), with associated SSN_{12}; together with the returned value ts_{12_f} (the timestamp when the message was actually received by the client 2), with associated RSN_{12}. The second table is dedicated to the received messages and stores the timestamps: ts_{21_i} (initial moment of the sending from client 2 to client 1) with associated SSN_{21}, together with the returned value ts_{21_f} (the timestamp when the message was actually received by client 1), with associated RSN_{21}. Note that the microprocessor's frequency is stored for each host (f_{CPU1} for Client 1, f_{CPU2} for Client 2).

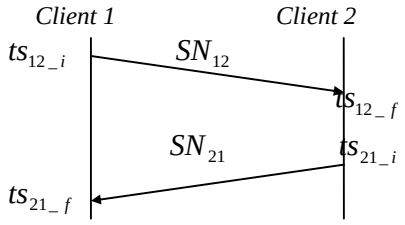

Figure 4. Four timestamps for measuring the RTT at the Application Layer

$$RTT = \frac{t_{21\_f} - t_{12\_i}}{f_{CPU1}} - \frac{t_{21\_i} - t_{12\_f}}{f_{CPU2}} \qquad (1)$$

On the data socket the elapsed time at the client could be evaluated only if the sending and receiving entities are on the same machine.

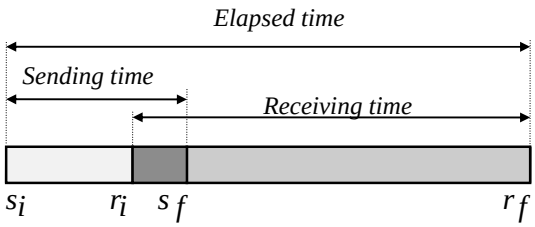

Figure 5. Four time stamps for measuring the sending, receiving and elapsed times at the client site

At the server site, the switching time is defined as the interval since the reception of the first TPDU started and the transmission of the last TPDU ended.

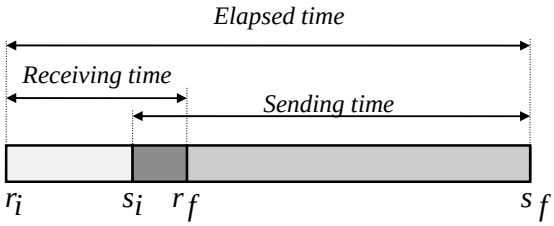

Figure 6. Four time stamps for measuring the sending, receiving and elapsed times at the server site

IV. EXPERIMENTAL RESULTS

The first experiments are dedicated to the influence of the application's buffer size (see *Table1* and *Table2*). All results were validated without employing any protocol described in Section III.

Application's buffer size [Bytes]	Average switching time *(IPv4)* [μ s]	Average switching speed *(IPv4)* [Mbps]	Average receiving time *(IPv4)* [μ s]
8192	111	575.84	217
5000	266	240.30	721
3000	333	191.95	1074
1500	617	103.59	2458
750	937	68.21	4945
375	1738	36.77	9502

Table 1. Testfile1, point-to-point, Client 1->server -> Client 1, without model. The average sending time/ throughput was 165 μ s/387.39 Mbps for IPv4.

The results for the application's buffer size of 8192 bytes are very important for evaluating the highest Layer 4 switching speed of about 575 Mbps (IPv4), respectively 1083 Mbps (IPv6). In general, supposing a theoretical transmission throughput of 100 Mbps (Fast Ethernet), it seems that the speed advantage is greater than 1 for a buffer size of at least 1500 bytes.

Due to the protocols stack the actual sending or receiving throughput at the lower layers cannot reach the upper bound of 100 Mbps.

Application's buffer size [Bytes]	Average switching time *(IPv6)* [μ s]	Average switching speed *(IPv6)* [Mbps]	Average receiving time *(IPv6)* [μ s]
8192	59	1083.38	102
5000	317	201.64	565
3000	486	131.52	1371
750	1249	51.17	5498
375	2601	24.57	10843

Table 2. Testfile1, point-to-point, Client 1->server-> Client 1, without model. The average sending time/ throughput was 264 μ s/242.12 Mbps for IPv6.

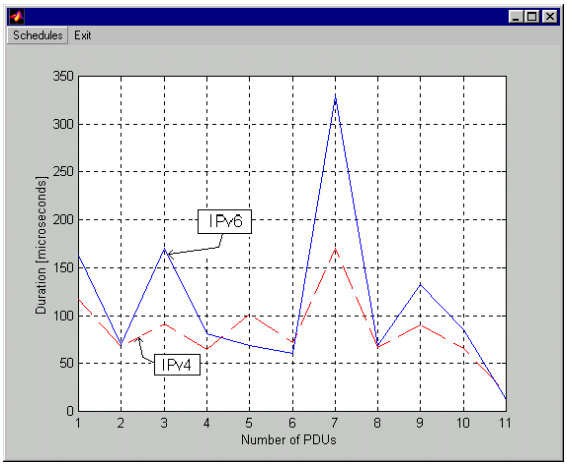

Figure 7. Switching time for application's buffer size of 750 bytes, Testfile1, point-to-point, Client 1->server -> Client 1, without model

Apparently, there is no advantage of using IPv6 instead of IPv4, as shown in *Figure 7*. This is a preliminary conclusion because several additional experiments should be performed. Let us involve now departure schedules (see *Figure 8*).

Note that there are two entities for each ON+OFF period. The first entity represents the number of bytes during the burst (for example 15582 bytes in *Model 1*, 209 bytes in *Model 2*, 62487 bytes in *Model 100*). The second entity is the total duration of the ON+OFF period (for example 0.005947 seconds in *Model 1*, 0.000016 seconds in *Model 2*, 0.005995 seconds in *Model 100*). Actually *Model 1* was designed for 25.6 ATM, as in

[2],[3], but it seems that it is also suitable for Fast Ethernet.

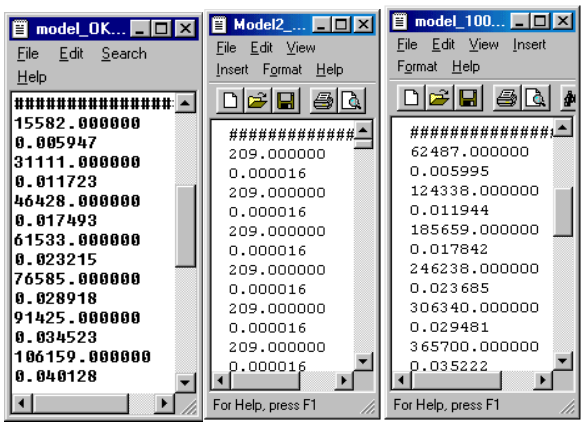

Figure 8. Model 1, Model 2, Model 100 for burst traffic

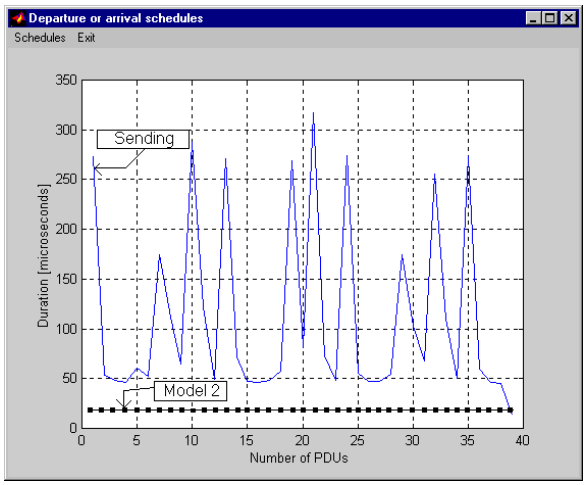

Figure 9. Model 2 for IPv6, Client 1 ->server-> Client 1, testfile1. The sending TCP entity cannot follow Model 2

Interval	Measured time [μs]	Throughput [Mbps]
$s_f - s_i$ SENDING	18543…18877 (CLIENT 1)	101.76..103.59
SWITCHING	74493…74643 (SERVER)	25.73…25.78
$r_f - r_i$ RECEIVING	82671…84942 (CLIENT 2)	22.61…23.23

Table 3. Model 100 for IPv4, point-to-point, Client 1 -> server-> Client 2, Testfile2, application's buffer size of 5000 bytes. The planned sending time/ throughput were 19209 μ s/100 Mbps without model and for Model2, respectively 22202 μ s/86.52 Mbps for Model100

Model 100 is the 100 Mbps-updated version of *Model 1*, but it is rather difficult to be accurately followed by the sending TCP entity.

Due to implementation difficulties, the experiments involving NTP, SNTP or the new proposed protocol are under progress and no preliminary results are considered to be included in this paper. However, we can estimate that more sophisticated trials still have to be performed. The APDU is encapsulated in a TCP segment, then in an IPv6/IPv4 datagram, then in a Fast-Ethernet frame.

According to [4],[9], we have to evaluate the following delays: access delay (the time requested to access the socket); packetization delay (the time needed to encapsulate the socket's data into a Layer 2 frame); transmission delay (to send the frame between Layer 2 and Layer 1, i.e. 13.6 microseconds in IPv4, respectively 16.8 microseconds in IPv6, at 100 Mbps); propagation delay (depending on the distance and on the type of media between the source and the destination); Layer 4 switching delay (at the server, see *Tables 1-3*); queueing delay and reassembly delay.

Whenever the RTT measured is different from the current value, the transfer rate for the sending model has to be proportionally adjusted. For example, let us suppose that the current 1024-byte RTT is 1437 microseconds. This means an average throughput of about 11.4 Mbps. The departure schedule will involve a 11.4 Mbps-based model (instead of a theoretical 100 Mbps-based model, such as *Model 100*). If the measured RTT is changing to 466 microseconds, the departures will follow a 35.15 Mbps-based model.

V. CONCLUSIONS

1. The preliminary comparison between IPv6 and IPv4 proved that there are no relevant permanent differences concerning the throughputs and the Layer 4 switching performances. This observation is valid for both Microsoft's Windows 2000 implementation (as from this paper) and Linux-based solution (as from our previous work).
2. It is more difficult to choose a departure schedule for Fast Ethernet comparing to equivalent conditions in ATM.
3. The Layer 4 switching performances could be improved by selecting the proper model at both sending client and the server.
4. Many users have an unrealistic expectation about the overall throughput, calculated within the interval since the first bit left the sender until the last bit reach the destination. The highest value, calculated at the application/Windows Sockets interface, is about 31-32 Mbps for 100 Mbps Fast Ethernet (for at least 366 MHz CPU's frequency).
5. Choosing a proper model for departure can generate better results (i.e. overall throughput, congestion avoidance etc.) than the expected ones obtained by

involving the classical one-block sending mechanism through sockets.

6. The involvement of the new protocol for measuring RTT at Layer 7 will limit the applicability of those models that are not supported by the network.
7. A dynamic adjustment of the transmission or switching schedule is becoming possible.

VI. FUTURE WORK

The next step is to determine the optimal model, depending on the specific application (burst traffic, voice, variable video streams etc). The overall performance of the Layer 4 switching is expected to be improved by running it on top of Layer 2/Layer 3 switches on the same machine. It is also for future work to evaluate the performances by involving time synchronization protocols (including the availability for other microprocessors family or other environments). Obviously it is for further study the optimum interval for RTT updating and the mechanism of switching from one sending model to another.

REFERENCES

[1] V. Dobrota, D.Zinca, C.M. Vancea, "Layer 4 Switching Experiments with IPv6 versus IPv4", *Proceedings of the International Conference Communications'2000,* Bucharest, Romania, December 7-9, 2000, pp.300-303

[2] V. Dobrota, D.Zinca, C.M. Vancea, A. Vlaicu - "Layer 4 Switching Experiments for Burst Traffic and Video Sources in ATM", *Digest of Papers. The 10th IEEE Workshop on LANMAN'99*, Sydney, Australia, 21-24 November 1999, pp. 66-69.

[3] V. Dobrota, D. Zinca, C.M. Vancea, A. Vlaicu, "Layer 4 Switching Experiments in a TCP/IP Environment for the ATM Sources", *ACTA Tehnica Napocensis*, ISSN 1221-6542, Vol.40, No.1, 2000, pp. 13-18.

[4] V. Dobrota, *Retele digitale in telecomunicatii. Volumul 2: B-ISDN cu ATM, Sistemul de semnalizare cu canal comun SS7*, Editura Mediamira, Cluj-Napoca 1998.

[5] D.L. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis ", *RFC 1305*, University of Delaware, March 1992.

[6] D. Zinca, V. Dobrota, M. Cosma, A. Vlaicu, "Software Traffic Analyzer and Frame Generator for IEEE 802.3u", *8th IEEE Workshop on Local and Metropolitan Area Networks LANMAN'96*, Berlin/Potsdam, Germany, August 25-28, 1996, pp. 243-248.

[7] J.M. Berthaud, "Time Synchronization over Networks using Convex Closures", *IEEE/ACM Transactions on Networking*, Volume 8, No. 2, April 2000, pp. 265-277.

[8] J. Day, "The (Un) Revised OSI Reference Model", *ACM/SIGCOMM Computer Communication Review*, vol.25, No.5, October 1995, pp.39-55.

[9] F. Tobagi, I. Dalgic, "Performance Evaluation of 10Base-T and 100Base-T Ethernets Carrying Multimedia Traffic", *IEEE Journal on Selected Areas in Communications*, Vol.14, No.7, September 1996, pp. 1436-1454.

[10] ***, *http://msdn.microsoft.com/downloads/sdks/platform/tcpip6.asp*

[11] ***, *http://www.cisco.com/univercd/cc/td/doc/product/l4sw/index.htm*