UDP, TCP, and IP Fragmentation Analysis and Its Importance in TOE Devices

Juan M. Solá-Sloan
Advisor: Isidoro Couvertier Ph.D.

Computer Information Science and Engineering
University of Puerto Rico, Mayagüez Campus
Mayagüez, Puerto Rico 00681-5000
juan_sola@yahoo.com
icouver@ece.uprm.edu

Abstract

In this document three different networks environments are analyzed using a TCP/IP offload engine point of view. Internet related protocols are studied in a real environment in order to assign candidates for a real TCP/IP offload engine implementation. The number of broadcast and unicast packets received are also analyzed. The test setup, network configurations, and the analysis of results are presented. A formula for calculating the number of wasted interrupts is presented. Also, tasks priorities imperative to implement in a TCP/IP offload device are established. This analysis improves the framework discussed on [Solá-Sloan02].

1. Introduction

As a net device speed increases the number of interrupts per unit of time that the operating system kernel must handle also increase. The Central Processing Unit (CPU) of a loaded multi-homed host could be overwhelmed by interrupt signals generated by the net device. A new trend of devices called TCP/IP Offload Engine (TOE) devices, is proposed for handling this problem [Yeh02]. TOE devices relieve TCP/IP processing load inside the typical computer network architecture. With TCP/IP offloading, CPU time used to handle TCP/IP is available for the application and socket layers [Solá02].

A TOE device proposes a similar offloading technique as the AGP 3D device [Couvertier02]. Currently the AGP 3D device is offloading the processor of real time three-dimensional mesh rendering. The 3D accelerator device is composed of hardware and software components that perform the 3D rendering tasks in real time. A TOE device, as described in the parallel framework presented in [Solá-Sloan02], will require hardware and software modules for handling TCP/IP efficiently.

As part of a broader study on TOE, an analysis of the datagrams behavior over different inter-network architectures was done. Datagrams were classified by their respective protocol and size. The purpose of this analysis is to refine ideas about a real TOE implementation.

2. Test Conducted

Two tests were conducted in different campuses within the University of Puerto Rico (UPR), one in Mayagüez and one in Bayamón. These networks were hybrid networks composed of mostly Microsoft networks (NT, 2000, XP) technology. The third test was conducted inside a different network architecture at a commercial site. This last intranet is composed mostly of Microsoft Windows workstations connected to a Novell Netware based network. For external transmission (SMTP, HTTP, FTP, etc.) TCP/IP traffic is used.

2.1 Test setup

Subjects used the Internet in different ways, mostly browsing through pages of heavy graphical content. Among these pages were picture galleries, email systems, and others. The typical 10/100Mbps Ethernet technology was used as the network technology at the workstation. Most of the pages browsed reside inside the American Continent. While the users were using the Internet, IPTraf [Gerard00] was used as the network monitor software. The data captured by IPTraf was

analyzed. IPTraf classified the packets by their protocol and size.

2.1.1 UPR-Mayagüez test setup

A PC running Linux Mandrake 8.0 was connected directly to a switch at the Ph.D. Computer Information Science and Engineering Laboratory at the Research and Development Center (R&DC) of the University of Puerto Rico at Mayagüez (UPRM). This switch is connected to the rest of the campus via a fiber optic cable installed between buildings in the R&DC, the Electrical and Computer Engineering (ECE) Building and other buildings within UPRM. A large percent of the workstations connected to this network use Microsoft Windows (NT, 2000, XP).

2.1.2 UPR-Bayamón test setup

A server running Linux Mandrake 8.0 was installed directly to a switch in the server room of the Computer Science Department at the University of Puerto Rico at Bayamón (UPRB). This switch was connected to the Campus Computer Center via a 10/100 Mbps connection. All Internet bound traffic had to pass through the Computer Center. This network was composed entirely of Microsoft Windows workstations and servers except for the Linux server used in the UNIX Operating System course.

2.1.3 New Port Sales, Inc. test setup

At New Port Sales (NPS), all Internet traffic had to pass through their Linux Box [Andrews00]. A separate Novell based file server provided the data required by the intranet users. The Linux Box was connected directly to a high speed ADSL modem with a 10Mbps network device and was connected to the rest of the network via a 10/100 Mbps network device, which was in turn directly connected to one of the switches. All the workstations used Microsoft Windows 9x or ME except the Linux Box.

3. Analysis of Results

This paper is part of a broader project on TOE devices. A detail analysis of the networks studied could be found in [Solá-Sloan02]. Figure 1 shows the packet distribution by size for the three networks. The data gathered by IPTraf was divided in four mayor groups based on packets ranging in size from: 1 to 375, 376 to 750, 751 to 1125, and 1126 to 1500 octets. The percent of packets received is shown along the y-axis.

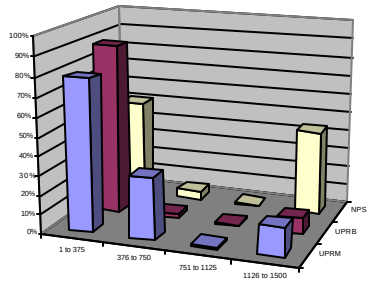

Figure 1. Packets Distribution by Size

We clearly see that in both UPRM and UPRB networks the number of small datagrams received is by far higher than the number of datagrams near the MTU size (1500 octets). Even in the NPS network architecture the percent of small packets received is higher (51%) than the percent of packets greater than 1126 octets (44%). This clearly shows that, even though the networks have a 1500 octets payload, the datagrams received are smaller 1500 octets. Therefore, as stated in [Comer01], small fragments traverse the Internet independently of the sender's or receiver's MTU.

Figure 2 shows the datagrams classified by their protocol. TCP was handled more than any other protocol. Close to 100% (99.5%) of the NPS network datagrams were TCP packets while the other networks received 78% .Approximately 12% of the packets handled by the UPRM and UPRB networks were UDP.

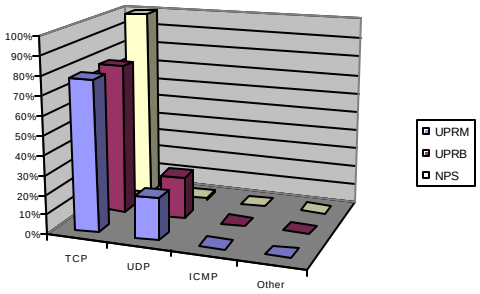

Figure 2. Datagrams classified by their protocol

The number of broadcast and unicast packets received by IPTraf in all three of the analyzed networks is depicted in Figure 3. UPRB and UPRM received approximately 85% unicast packets and a 15% of broadcast packets. The number of broadcast packets in the NPS network was 0.2%. Most of the broadcast packets received by the other networks were generated

by the network operating systems running in workstations and servers. Microsoft Windows network operating system (NT/2000,XP Pro) were generating broadcast datagrams containing UDP and TCP targeting port 137 and 138. These ports are reserved for NetBios. DHCP and Bootp were targeting ports 68 and 67 using UDP. NPS network configuration does not use DHCP, Bootp, and Netbios.

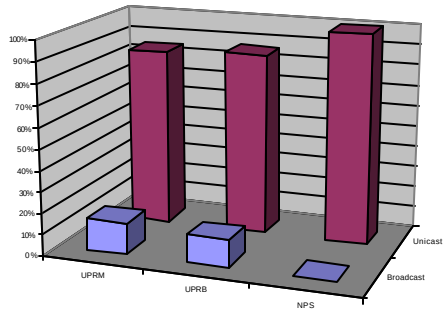

Figure 3. Broadcast vs Unicast

3.5 Intranet related packets and configuration issues

UDP and TCP packets generated by Microsoft operating systems targeting Netbios were captured by IPTraf. These packets affected our test in UPRB and UPRM but not at NPS. This could be seen in Figure 2 where the number of UDP packets at the UPR campuses are similar. UDP packets received by the NPS network were generated by the Domain Name Server (DNS) (port 53). Also the configuration on NPS is different than the configuration of the other networks analyzed. The Linux Box is connected directly to an ADSL modem via a network device. Another network device is connected to the network switch. The Linux Box is acting as a gateway masquerading packets along the way to the network. Also, the Linux Box is filtering IPX and SPX packets generated by Novell Netware client-server operating system.

4. Packet Wasted

Every time a packet is received a one to one relationship occurs between an interrupt and a packet reception. If we know the number of packets received we know the number of interrupts signals to the operating system for every packet received. First, we can calculate the number of octets wasted per received frame with the following equation:

$$w = \sum_{i=1}^{n} (a - O_i)$$

Here w is the number of wasted octets in n frames. A is the frame size of the network architecture, and O_i is the number of octets received within the packet. We can convert from wasted octets to wasted to packets with the following equation:

$$P = \left\lceil \frac{1}{a} W \right\rceil$$

Here p is the number of wasted packets. This also means that the number of wasted interrupts is equal to the number of wasted packets in the transmission because every time a packet is received an interrupt is generated. Therefore:

$$I = P$$

Where I is the number of interrupts generated. As a result, we can calculate the percent of packets wasted in our tests. This percents were: UPRM 60%, UPRB 53%, NPS 67%. Sixty percent conforms the mean of the average calculated by the interrupt wasted formula. More than half of the interrupts generated by packet reception were wasted.

5. Preparing the Ground for a TOE device

Its imperative, for a TOE device to handle fragmentation. Mainstream Internet implementations are done in uni-processor systems. A TOE hardware include in an Intelligent Network Interface Card as [Couvertier02] proposes, would offload the processor of TCP/IP handling. Base on the studies conducted, the following discussion targets issues about what should a TOE device should handle.

5.1 Issues that a TOE device should handle

We have analyzed that even in does filtered environments, the amount of small packets received from the Internet is greater than does packets that use the Ethernet payload size at maximum. Then, extending the payload size is not a substantial gain in performance if the packets will be fragmented along the way.

Broadcast packets generated inside the Intranet must be filtered in a real TOE device implementation in order to discriminate when it will interrupt the CPU and when it

will act upon the reception. At high speeds a high amount of these broadcast packets could be harmful if a TOE device does not handle them. If they are not capture, then they will generate a signal inside the operating system. Every TOE device must handle these interrupts in order to minimize the amount of times the main processor is interrupted.

More UDP packets were received than ICMP, IGMP or OSPF. In [Couvertier02], I have stated that a good TCP/IP and related protocols implementation must include ICMP. After this analysis, UDP is more important than ICMP. A significant amount of packets were received in UDP by the network monitors connected to their respective networks. Also, datagrams targeted to Netbios, Bootp, DHCP and DNS should be filtered or handled by the TOE device.

5.2 Primary candidate processes for offloading

IP has the highest priority. All other protocols that use the Internet infrastructure needs IP. In our test 100% of the packets were IP based. Fragmentation handling is a most in this layer. A TOE device must handle fragmentation at the receiving end. This will reduce the amount of interrupts generated to the CPU for TCP/IP processing. Therefore, fragmentation would not be harmful to our uni-processor architectures [Kent87].

Among the Transport Layer protocols, TCP has the highest priority as an offload candidate. In the non Microsoft configuration network (NPS) 99.8% of the IP packets encapsulated TCP. The other networks analyzed an approximate of 78% of the datagrams received contained TCP.

UDP must be process by the TOE device. In case UDP could not be processed at least a filter could be use to minimize the amount of interrupts generated to the CPU.

5.3 Secondary candidate processes for offloading

As stated before, ICMP is not more important than UDP. ICMP packets and other related protocols were not substantial. That's why they are selected as secondary candidates. Also if a TOE device will not handle these protocols it must discriminate when to interrupt the main CPU. A nice approach is to implement a way in which a TOE device could discriminate what protocols will process, what will process the main CPU, and which will be discarded without interrupting the CPU.

In the past, studies attempted to offload checksum verification. The amount of checksum errors were insignificant to the amount of packets received. That is why, checksum verification, is a secondary candidate and relies in the last paragraph of this section. Even though, checksum calculation is not so difficult to implement in hardware. All the packets needs to be checksum.

6. Conclusions

As [Solá02] proposed, maximizing the payload size in the frame carrier is not a feasible solution for datagram handling if these datagrams travel the Internet. Following fragmentation, each datagram is received encapsulated inside the frame carrier's payload. This produce a 1 to 1 relation for each datagram fragment [Couvertier02] received. Fragmentation is handle by software in uni-processor environments [Kent87]. A TOE device must handle fragmentation and de-fragmentation of datagrams in order to minimize the amounts of interrupts generated to the CPU. Also, a TOE device must discriminate when to generate these interrupts. An efficient TOE device would need to handle UDP and the way NetBios is handled by the CPU. All broadcast messages should be handled prior to CPU interruption. Also a TOE device should discriminate: which protocol will it processed, which processes will be handled by the main CPU, which protocols will be discarded.

7. Future Work

In [Solá02] a Parallel TCP/IP Offloading Framework is described. A UDP and Internet Related Protocols Framework should be design if maximal offloading needs to be accomplished inside a TOE device.
The test was conducted while browsing through image galleries in sites inside the American Continent. What happen if the test is conducted using a file-transfer protocol (FTP)? Does an FTP transfer simulates an ISCSI device receiving data? By the date of this article a feasible offloading solution is being designed on software.

References

[Andrews00] Andrews, Eric and Andrew Comech. 2000. "The Cheap Linux / Box," [online] http://www.ls.net/CheapBox.html

[Comer01] Comer Douglas. 2001. "Computer Networks and Internets," New Jersey: Prentice Hall

[Couvertier02] Couvertier Isidoro and Juan M Solá. 2002. "TCP/IP Offloading Framework for a

TCP/IP Offloading Implementation," Mayagüez, Puerto Rico: Computing Research Conference 2002. University of Puerto Rico. [online] http://mayaweb.upr.clu.edu/crc/crc2002/papers/Sola_Juan.pdf

[Gerard00] Gerard J. IPTraf Software. [online]http://cebu.mozcom.com/riker/iptraf/

[Kent87] Kent Cristopher A and Jeffrey C. Mogul. 1987. "Fragmentation Considered Harmful," [online] Palo Alto, California: Digital / Compaq, Western Research Laboratory. Available from the World Wide Web: [online] http://research.compaq.com/wrl/techreports/abstracts/87.3.html

[Solá02] Solá Sloan, Juan M. and Isidoro Couvertier. 2002. "A Parallel TCP/IP Offloading Framework for a TCP/IP Offloading Implementation," Grenoble, France: IP SOC Workshop 2002 Proceedings.

[Solá-Sloan02] Solá Sloan Juan M. 2002. "A Parallel TCP/IP Offloading Framework," [online] http://www.geocities.com/Juan_Sola/toe.html

[Yeh02] Yeh, Eric, Herman Chao, Venu Mannem, Joe Gervais and Booth Bradley. 2002. [online] "Introduction to TCP/IP Offload Engine (TOE)". 10 Gigabit Ethernet Alliance. [online] http://www.10gea.org