Framework for Ethernet Network Functionality Testing

Mirza Aamir Mehmood [1], Ahthasham Sajid[2] and Amir Shahzad Khokhar[3]

Department of Computer Science, Balochistan University of Information Technology, Engineering & Management Sciences Quetta, Pakistan

Abstract

Computer networks and telecommunication systems use a wide range of applications. Therefore, the power and complexity of computer networks are increasing every day which enhances the possibilities of the end user, but also makes harder the work of those who have to design, maintain and make a network efficient, optimized and secure. Ethernet functionality testing as a generic term used for checking connectivity, throughput and capability to transfer packets over the network. Especially in the packet-switch environment, Ethernet testing has become an essential part for deploying a reliable network. A platform and vendor independent framework is required to verify and test the functionality of the Ethernet network and to verify the functionality and performance of the TCP/IP stack. "NetBurst" is developed for Ethernet functionality testing.

Keywords: *Communication, Networking, Performance, Load, Portability, Throughput, TCP, IP, Protocol, Ethernet, Testing.*

1. Introduction

In the late 1990s, the spectacular growth of the Internet dramatically affected the evolution of computer networking. Numerous network techniques and technologies boomed, but quickly faded into oblivion. Others have stood the test of time. This growth is not possible without protocols and communications software. As a result, the end user is greatly empowered with numerous choices available to them. Contrary to this, the large number of transmission techniques and communication protocols had made it very difficult for the network engineers, who have to design, develop and maintain the performance and optimization of their network. Thus, with the power and complexity of computer networks raised, the need for tools to measure, analyze and test the functionality and performance of the network is greatly intensify.

2. Related Work

Number of researchers has worked on Ethernet and protocol testing. This section provides an insight of work done in relevant area.

There are two main methods to capture data from a network; the first is based on the use of dedicated hardware, while the second makes use of the hardware of a normal PC or workstation connected to the communication channel. In the second method, the network adapter of the computer is used to obtain the frames from the network, and the software performs packet capturing process. The software solution has usually the low performance as compare to hardware solution, particularly on slow machines, but it is cheaper, and easier to modify and upgrade. For this reason, it is widely adopted on the most used network architectures, where the performance of dedicated hardware is not needed. In network performance analyzing or Ethernet testing, ranges of different software and hardware tools and products are available for packet capturing, packet filtering and network monitoring [4, 5].

Following are the software and hardware products for capturing and analyzing network traffic.

2.1 Tcpdump

Tcpdump is a simple packet sniffer that uses command line interface, and allows a user to intercept and display TCP/IP and other packets being transmitted or received over a network. It is freeware software which works on most Unix-like operating systems (Linux, Solaris, BSD, Mac OS X, and HP-UX) and there is also a port of tcpdump for Windows called WinDump [6].

2.2 Wireshark

Wireshark is similar to tcpdump, but it provides graphical user interface (GUI) and many other options as well, like packet filtering and sorting. It is a free packet sniffing application. It works for different protocols with deep inspection of packets and uses decryption options for many protocols. It can run on Windows, Linux, Solaris, Mac OS and BSD [25]. Wireshark can be used for data analysis captured by NetBurst.

2.3 Snoop

Snoop is a freeware command line packet sniffer included as part of Sun Microsystems' Solaris Operating System. It is dedicated for the Solaris system, on a data-link or IP interface. Snoop captures packets and displays their contents. If the data-link or IP interface is not specified, snoop picks the first non-loopback data-link it finds [7].

2.4 TPTEST

TPTEST is another application used for measuring performance on Internet connection; TPTEST measures the throughput speed from various reference servers on the Internet. TPTEST measures the throughput of TCP/UDP incoming and outgoing packets and packet loss. The application is written in C++ and is portable for Windows, Mac OS, Linux and BSD [8].

2.5 Maxwell Network Emulator

Maxwell network emulator is a hardware appliance that helps network managers, software developers and testers learn how their products will perform in real-world production networks, including satellites and the Internet. It has both graphical and command line interface and also script driven interface for maximum flexibility. Moreover, it is fully customizable and programmable using C++ [9].

2.6 IxANVL™ - Automated Network Validation Library (ANVL)

Ixia's IxANVL (Automated Network Validation Library or ANVL) is another hardware device that is also known as the industry standard for automated network/protocol validation. Software developers and manufacturers of networking equipment and Internet devices use IxANVL to validate protocol compliance and interoperability. [10].

There are many software applications for packet sniffing for example TCP Dump, WireShark but they can only analyze traffic. They do not offer any option to inject customized packets. Applications that can test TCP for example TTCP [22], can send TCP packets, but they do not support custom packet creation. Applications like Snoop are platform dependent and can run on a specific platform.

The research work done so far also focused on certain environments or functionalities. Not a single research work provides us all the required functionalities that are sufficient to test an Ethernet node in our desired way. Since existing applications and research work do not fulfill all the requirements, a new application has to be developed that can fulfill all the requirements. We have adopted the technique that Douglas E. Comer and John C. Lin have demonstrated [21], i.e. active probing technique, to develop application named “NetBurst”.

3. Application Development Environment

The application has been developed in ANSI C using Eclipse IDE on Fedora core 10. External libraries used are Libnet and Libpcap on Linux where, as on the Windows platform, Winpcap and Libnet for Win32 are used, which are ported versions of early stated libraries.

3.1 Libnet

Libnet is a library for C programming used for packet construction and injection. Libnet is used to control every field of every header of every packet; large number of programs goes through a high-level interface in order to send traffic on the network. Occasionally, for security or hacking reasons, a program needs to construct its own network headers. The existing TCP/IP stack is unable to build these headers, and it must bypass it and go directly to the hardware drivers. Libnet is a library that makes custom packet generation easier [23].

3.2 Libpcap

Libpcap gives a portable structure for low level network monitoring. Libpcap can provide network statistics collection, security monitoring and network debugging. Every system vendor provides a different interface for packet capture. To overcome this problem, Libpcap was developed. This system-independent API makes it easier to port and alleviates the need for several system-dependent packet capture modules in each application [6].

3.3 Eclipse

Eclipse is an open source, integrated development environment. It comprises extensible frameworks, tools and runtimes for building, deploying and managing software. It provides plugins in order to provide all of its functionality on top of the runtime system; this is in contrast to some other applications where functionality is typically hard coded. This plug-in mechanism is a lightweight software component framework. Moreover, Eclipse allows extension using other programming languages, such as C and Python.

This plug in setup allows Eclipse to work with networking applications such as Telnet, and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management [24].

4. NetBurst Architecture

NetBurst is developed for Ethernet functionality testing. It is a vendor independent, cross platform application. It can be used to test load, performance and functionality of underlying TCP/IP stack. Functionality testing includes throughput, packet loss, latency, fragmentation, option processing etc.

4.1 Application Overview

The following figure provides an overall working model of NetBurst. The application works by sending customized packets, capturing and observing the response by sniffing the packets. The packet generator is sending packets to the machine whose protocol stack needs to be tested. Sniffer is running on machine under test (MUT) to capture and observe the response of MUT's protocol stack.

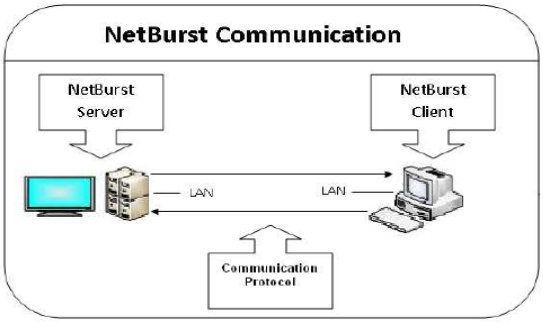

Figure 1. NetBurst Communication

Communicating nodes are connected in LAN and protocol being used is TCP/IP. Supported protocols are Internet Protocol and Transmission Control Protocol. NetBurst supports only Ethernet technology. An error will be generated if any other technology is being used.

4.2 Communication Mode

The application can be executed in two communication modes: synchronous and asynchronous. When executed in synchronous mode, the client is running on MUT, and the server is sending TCP or UDP packets. This mode is recommended only for test case designing, since each and every packet is processed to extract fields in the packet header. All incoming traffic displayed on the terminal. Since all incoming packets are processed, it may result in slower packet capturing and may lost many correctly received packets. Due to these reasons, this mode may produce false results. When executed in asynchronous mode, Packetizer and sniffer run independently. All captured packets are stored in a file for future processing.

4.3 Application Architecture

NetBurst has two main components: Packetizer and Sniffer. Packetizer has been built over Libnet and is controlled through configuration files. Sniffer is built over pcap and Winpcap libraries and is also controlled through configuration files. These configuration files are used to control the packet transmission rate, packet construction, and various other properties.

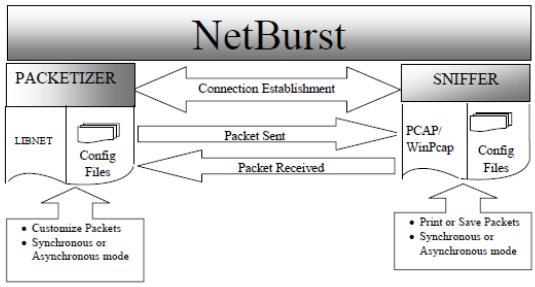

Figure 2. Application Architecture

As shown in the above figure, Packetezier can transmit random or fixed length packets. On the other hand, Sniffer records all incoming traffic and the response generated by MUT. The above figure depicts the application's execution mode.

4.3.1 NetBurst Configuration File

A configuration file is designed to control the application behavior. Following is an example of a configuration file containing different parameters. A user can interact with the application through this file to control application behavior. The structure of the configuration file is given in following figure.

```
time: 2
interval: 1
count: 3
ip_ver: 4
random: 1
cache: 0
protocol: 1
```

Figure 3. Configuration File

4.3.2 NetBurst Packetizer

This component is responsible for constructing and injecting packets. To inject a packet, NetBurst reads the packet constructed that is values of different fields of packet header from a specific file for each protocol header. The structure and purpose of these files are elaborated below.

4.3.3 TCP Configuration file

The following diagram depicts *tcp.opt* file which is used in Packetizer. The record parameter controls number of packet constructs provided in the file. The value of the source port and destination port is given in *sourc_port* and *destination_port* fields. The *window_size* takes the value for the window size, whereas *flag_urg* is used to take the value of the urgent flag which could be any numeric value. Sequence and acknowledge values are provided through *seuq* and *ackg***,** respectively. The user also has to provide the control flags (for example th_syn, TCP use this flag in its header to initiate communication) and this is done through providing the value in the *control_flag* parameter

```
record: 1
sourc_port: 1234
destination_port: 9876
window_siz: 77
flag_urg: 9
sequ: 11
ackg:8
```

Figure 4. Configuration File

4.3.4 IP Configuration File

The structure of the *ipv4.opt* file is depicted in figure 5. The application uses this file to construct a custom IPv4 packet. In *id*, the value of ID field of IP header is provided. The user provides the source and destination address in *source_ip* and *dest_ip* respectively. *Time_to_liv* takes the value for time to live, whereas *tos* takes value of type of service field of IPv4 header

```
record: 1
id: 2
source_ip: 192.168.1.1
dest_ip: 192.168.1.11
time_to_liv: 12
tos: iptos_lowdelay ¦
ipptos_throughput ¦
iptos_reliablity ¦
iptos_mincost!
```

Figure 5. IP Configuration File

5. Results

In this section, test cases and results are discussed to analyze the TCP functionality. Test cases are categorized in four groups: performance, load, socket settings and TCP functionality. For traffic analysis, Wireshark is used, which is standard traffic analysis tool.

5.1 Performance Testing

The performance of any communication system can be measured based on three main factors i.e. throughput, latency and packet loss. To analyze and measure these factors, a set of test cases are executed through NetBurst, and the performance of the underlying TCP Stack is observed. Following are the results of these test cases.

5.1.1 Test Case I: Throughput

To verify throughput, a burst of packets having packet size 1500 bytes is fired for 60 seconds. The following graph depicts the throughput.

Throughput is defined as number of bytes received divided by the transmission time. The figure 6 shows a normal test with no throughput problem. It is observed that the level of "fuzziness" of the throughput speed distribution is normal and may reflect a slight timing inaccuracy in the Ethernet Network Functionality Testing computer's clock. On analyzing the graph, a small gap is observed, which indicates a packet delay after a retransmission.

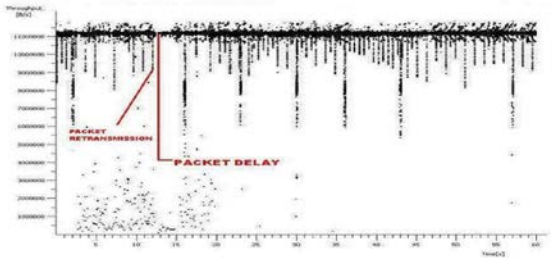

Figure 6. TCP Packet Throughput Graph

5.1.2 Test Case II: Latency

In continuation to test case I, a latency graph is also produced which is given below. In figure 7, packet number is given on x-axis and time (in seconds) given on y-axis

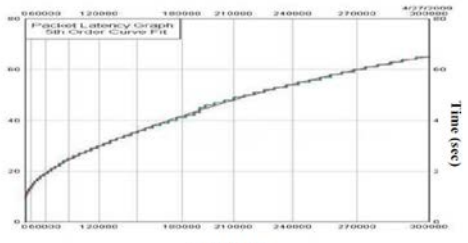

Figure 7. Packet Latency

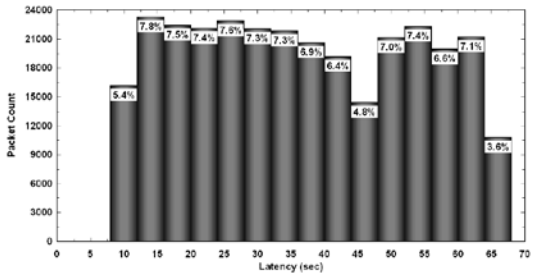

Figure 8. Packet Latency Histogram

The above graph shows latency during a burst of packet transmitted from NetBurst. We can deduce from the figure 7 that the latency gradually increases with the time. Likewise, in figure 8, latency histogram depicts the latency interval. It also shows that 7.8% packets were affected with highest latency. On the other hand, 3.6% of packets were affected by lowest packet latency, which occurred at the last packet transmitted. The variation of delay, known as jitter, occurred during transmission as depicted in figure 8.

5.1.3 Test Case III: Packet Loss

Packet loss occurs due to several reasons including the network conjunction. For analyzing packet loss, NetBurst has transmitted a total of 478861 packets. On the receiving node a total of 430550 packets were received, which shows that 11% of the packets were lost. For statistical analysis, the following graph shows the loss which occurred in the transmission.

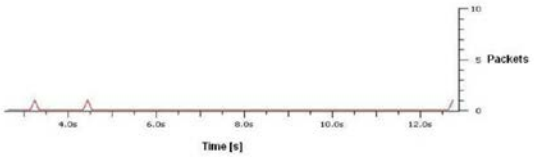

Figure 9. Packet Loss Graph

From the figure 9 it is clear that a small number of packets are lost during the whole period of packet transmission. In the above figure, small peaks depict packet loss with respect to packet sequence number.

5.2 TCP Functionality Testing

The objective of these test cases is to test the functionality of the TCP stack. A set of sample test cases were executed. These test cases, and their results, are discussed in this section.

TCP establishes connection using three way handshake mechanisms. This functionality is verified here.

5.2.1 Test Case I: Establishing Connection on Open Port

The following flow graph, shown in figure 7-6, describes how a "SYN" is acknowledged from the machine under test with a "SYN ACK".

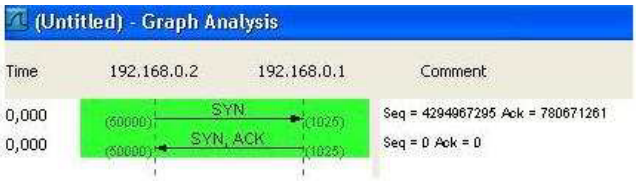

Figure 10. TCP Handshake

The above figure depicts that client has sent a packet with "SYN" flag. Server has acknowledged this "SYN" request through a TCP packet having control flags "SYN ACK".

5.2.2 Test Case II: Establishing Connection from Closed Port

In this test case, a "SYN" is sent from the client and then the port is closed. The server has acknowledged this "SYN" with a "SYN-ACK" packet. Since the port is closed from the client, an "RST" will have been sent to the server to inform it that the port is closed. The following graph in figure 11 depicts the same scenario.

Figure 11. TCP Connections.

5.3 Packet Parameters

Once the connection is setup, a desirable test is to see how the TCP stack deals with legal and illegal sequence numbers. Following test case verifies this functionality.

5.3.1 Test Case I: Sending Illegal Sequence and Acknowledgment Numbers

As shown in the below figure 12, the TCP stack should mark and ignore illegal SEQ and ACK number packets as out-of-order packets.

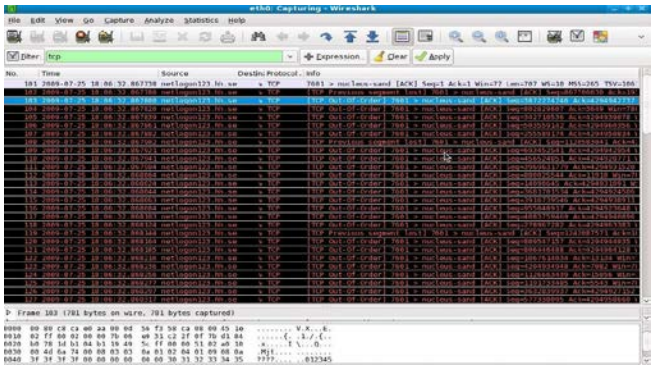

Figure 12. TCP Connection

5.4 TCP Option List

How the TCP stack processes the different TCP options is validated here. A set of test cases are executed to analyze different aspects of option processing

5.4.1 Test Case I: Valid options with legal values

An options byte string is sent with “SYN” packet. Option string is 20 bytes long and has the value "\003\003\012\001\002\004\001\011\010\012\077\077\077\077\000\000\000\000\000\ 00"

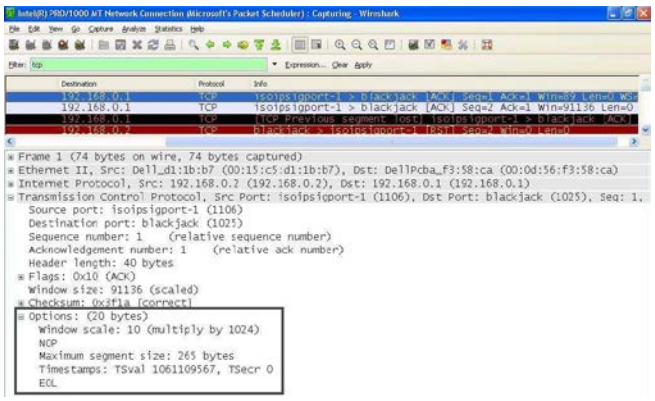

Figure 13. TCP Option Processing

5.4.2 Test Case II: Valid options with illegal and unusual values

An options byte string is sent with the “SYN” packet and the values "\003\003\012\001\002\004”**.** As shown in the figure 14 invalid options are detected and an error message “option goes past end of option” is displayed.

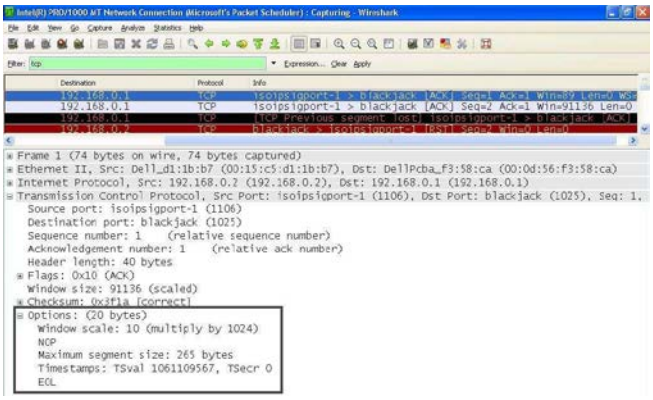

Figure 14. Bad Option Processing

In this section we have elaborated on TCP Functionality testing with different test cases for TCP handshake, Connection Establishment etc using “NetBurst” application. In the same way we can check the TCP functionality for TCP Packets, Payloads, and IP Fragmentation

6. NetBurst Portability

Due to resource constraints, the application portability is tested only for Windows and Linux systems. The application should work on VxWorks and Solaris, as the libraries used to develop this application claim to work on the above mentioned operating systems [23].

7. Conclusions

This paper introduces a cross platform and vendor independent farm work that uses active probing technique to test Ethernet functionality, comparatively, the NetBurst application gives an efficient result and a cost effective solution for network functionality testing.

There are many software applications to test the Ethernet functionality, like TCP Dump, WireShark but they can only analyze traffic. They do not offer any option to inject customized packets. Applications that can test TCP for example TTCP can send TCP packets, but they do not support custom packet creation. Applications like Snoop are platform dependent and can run on a specific platform. All these drawbacks are resolved in “NetBurst”.

Active probing technique was introduced by Douglas E. Comer and John Lin.Douglas E. Comer and John C. Lin’s work is focused only on RTO (retransmission time-out) estimation, retransmission interval and keep-alive functionality, and does not test any other functionality available in TCP

Our test cases have tested TCP/IP stack functionality and results are discussed in this paper. It is also explained in the report that the number of packets per second, and the latency, are directly proportional to each other. Latency will increase gradually as the number of packets increase. NetBurst also gives an output result of the network performance testing, such as load, packet loss, latency, and throughput.

8. References

[1] S. Bradner & J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", *RFC 2544*, Available at: http://www.ietf.org/rfc/rfc2544.txt. (Last accessed July 09, 2009).

[2] Douglas E. Comer, Internetworking with TCP/IP: Principles, Protocols, and Architecture, Volume 1 (4th ed.), Upper Saddle River: Prentice Hall, 2000, ISBN: 0130183806 (International Ed.)

[3] "Resolve IP Fragmentation, MTU, MSS, and PMTUD Issues with GRE and IPSEC", Available at: http://www.cisco.com/en/US/tech/tk827/tk69/technologies_white_paper 09186a00800d6979.shtml. (Last accessed July 1, 2009)

[4] "Ethernet Testing Software's" Available at http://www.allbusiness.com/technology/co puter-networking/831290-1.html. (Last accessed July 1, 2009)

[5] "Ethernet Testing Hardware" Available at: http://www.iol.unh.edu/services/ testing/ethernet/tools/. (Last accessed July 1, 2009).

[6] "TCP Dump", Available at: http://www.tcpdump.org/. (Last accessed July 1, 2009).

[7] "Snoop (1M) - capture and inspect network packets (man pages section 1M: System Administration Commands)", Available at http://docs.sun.com/app/docs/doc/819-2240/snoop-1m?&a=view&q=snoop. (Last accessed July 09, 2009).

[8] "TPTEST", Available at http://tptest.sourceforge.net/about.php. (Last accessed July 09, 2009).

[9] "Maxwell Network Emulator Information", Available at http://www.maxwelltester.com. (Last accessed July 09, 2009).

[10] "IxANVL™ - Automated Network Validation Library (ANVL)", Available at http://www.ixiacom.com/products/display?key=ixanvl#note1#note1, (Last accessed July 1, 2009).

[11] W. Hengeveld, "Protocol Testing: Using hardware techniques for software", *Seventh International Conference on Software Engineering for Telecommunication Switching Systems*, 1989. SETSS 89, July 1989

[12] Ethernet Network Functionality Testing Chanson, S.T. and Zhu, J, "A Unified Approach to Protocol Test Sequence Generation", *INFOCOM '93. Proceedings. Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future*. 28 March-1 April 1993 Page(s):106 - 114 vol.1

[13] Huang,C.M., Lin, Y.C. and Jang, M.Y, "An Executable Protocol Test Sequence Generation Method for EFSM Specified Protocols". IWTCS 95, Evry, 4-6 September.

[14] C.Bourhfir,R.dssouli,E. Aboulhamid, PI.Rico, "Automatic executable test case generation for extended finite state machine protocols", l0th IFIP IWTCS, 1997.

[15] Wu; Wang, "Internet protocol conformance testing by using a TTCN based protocol integrated test system," *Communications, 1999. ICC '99. 1999 IEEE International Conference*, vol.1, pp.646-650, 1999

[16] K Shemyak , K Vehmanen , "Scalability of TCP servers, handling persistent Connections", *Proceedings of the Sixth International Conference on Networking*.22-28, April 2007 on page(s): 89-89

[17] D Kassabian, A Albicki, "A Protocol Test System for the Study of Sliding Window Protocols on Networked UNIX Computers" *IEEE Transactions On Education, Vol. 38, No. 4. November 1995*

[18] X Xie; M Zheng; Kassabian, D.; Albicki, A., "Design and testing of a sliding window protocol in a protocol testing system," *Circuits and Systems, 1993., Proceedings of the 36th Midwest Symposium* on , pp.1144-1147 vol.2, 16-18 Aug 1993

[19] Y Lin, R E. Newman, H Latchman "A New TCP and UDP Network Benchmark Suite", *Proceeding of the 10th Communications and Networking Simulation Symposium* (CNS'07), March 2007.

[20] R J. Linn, Jr., "Conformance Evaluation Methodology And Protocol Testing", *IEEE Journal on Selected Areas In Communications*. Vol. 7. No. 7. September 1989

[21] D E. Comer , J C. Lin, "Probing TCP implementations", *Proceedings of the USENIX Summer 1994 Technical Conference on USENIX Summer*, p.17-17, June 06-10, 1994, Boston, Massachusetts. Available at: http://www1.bell-labs.com/user/johnlin/probing-TCP.pdf, (last accessed July 1, 2009)

[22] "Test TCP (TTCP) Benchmarking Tool for Measuring TCP and UDP Performance", Available at http://www.pcausa.com/Utilities/pcattcp.ht . (Last accessed July 09,2009).

[23] "Libnet", Available at: http://libnet.sourceforge.net/#whatis (Last accessed July 1, 2009)

[24] "Eclipse", Available at: http://www.eclipse.org/ (Last accessed July 1, 2009)

[25] "Wireshark", Available at http://www.wireshark.org. (Last accessed July 1, 2009).

[26] Eric Hall, Internet Core Protocols: The Definitive Guide Help for Network Administrators (1st ed.), 1005 Gravenstein Highway North Sebastopol, CA: O'Reilly Media, Inc., 2009, ISBN: 1565925726

[27] Douglas E. Comer, Internetworking with TCP/IP: Client-Server Programming and Applications, Linux/Posix Sockets Version, Volume 3 (4th ed.), Upper Saddle River: Prentice Hall, 2000, ISBN: 0130320714 (Paperback Ed.