International Journal of Current Trends in Engineering & Research (IJCTER)
e-ISSN 2455–1392 Volume 4 Issue 4, April 2018 pp. 1 – 11
Scientific Journal Impact Factor : 4.23
http://www.ijcter.com

Packet Sniffer for Users End Network Performance Monitoring using Python Programming

Adeyinka A. Adewale[1], Victor O. Matthews[2], Adebiyi A. Adelakun[3], Winifred Amase[4], Olaitan Alashiri[5]

[1,2,3,4,5]*Department of Electrical & Information Engineering¸ Covenant University, Ogun State, Nigeria*
corresponding author: ade.adewale@covenantuniversity.edu.ng

Abstract-Monitoring of data or information transmitted over network is becoming necessary with rapid increase in malicious intents on the global communication networks coupled with the continuous growth in size and complexity of the same. This research study presents the implementation of packet sniffer developed using python programming language to monitor the performance of the network at the users side on a campus local area network in terms of packet loss ratio or error rate. The evaluation was carried based on the model developed.

Keywords: Network traffic analysis, Packet sniffing, Python, Packet loss ratio rate

I. INTRODUCTION

Communication networks are more computerized than ever and rapidly increasing at a significant rate both in size and complexity. Monitoring the information or data transmitted over these networks is getting progressively difficult. As such, networks are subject to neglect and exposed to malicious attacks which could prove disastrous in protection and securing of data. Designing software that would have the capability to monitor and check/analyze the numerous data been transmitted over a computer network is one way to proffer solution to the challenge of monitoring systems. The software will make available the possibility of capturing data in terms of its formatted unit of data known as packet. The captured data which is in its binary format is the converted to readable format and displayed making it possible for network administrators or security professionals to monitor the information being transmitted over the network. Packet sniffer, also known as packet analyzer/ protocol analyzer/network analyzer is one of such software which has the ability to capture data passed over a network, converting it to readable format and reading of data content. It is known to have numerous versatile uses as the packet sniffer can be used legitimately by a network or system administrator to monitor and troubleshoot network traffic [1][3]. It is also a vital tool in identifying intruders on a network. It has had a large role in securing of network systems, resolving issues that arise in the network and identifying inconsistent connections. Packet sniffer has been defined in various ways. It has been defined as a tool used to monitor, intercept and decode data packets as they are transmitted across networks. It has been described as a tool that utilizes the process of gathering traffic from a network and storing them for later analysis [2].

The packet sniffer, in the present world has several significant uses that most security experts utilize in an effort to make network secure for data transmission without worry of an attack. It is relied upon to monitor and filter network traffic and it is an effective tool for testing protocols, diagnosing network problems, identifying configuration issues. Information technology- teams rely on packet sniffers to discover network misuse of any kind and also resolve bottleneck issues. In an intrusion monitoring system, packet sniffers have played a vital role as the monitoring of data is first carried out before detection and prevention of suspicious or malicious activity is performed.

Each machine on an immediate network has its own hardware address which differs from the other. When a data is sent across the network, it is sent in the form of packets. These packets are the chunks of data are directed to the certain designated system though will pass through some nodes on the network. Normally a particular system in a network is designed to receive and read only those data which are intended for it, so, for this reason, the network interface card (NIC) works in non-promiscuous mode and promiscuous mode. When a packet is received by a NIC, it first compares the destination address of the packet to its own. If the MAC address matches, it accepts the packet otherwise filters it. This operation mode where network card discards all packets that do not contain its own MAC address is called non promiscuous mode which basically means that each NIC reads only the frames meant for it [4][5]. Hence, packet sniffer captures the packets by setting the NIC card of its own system into promiscuous mode and when the packet arrive at the NIC, it is copied to the device driver memory, then passed to the kernel.

The rest of this article is divided into four sections. Literature review of the study was carried out in section II, while the materials and methodology of research is presented in section III. This followed by implementation and testing discussions in section IV. The paper is concluded in section V.

II. LITERATURE REVIEW

Packet sniffer is a program running in a network attached device that passively receives all data link layer frames passing through the device’s network adapter [5]. Using information captured by packet sniffer, an administrator can identify erroneous packets and use the data compiled to pinpoint bottlenecks and help maintain efficient network data transmission. Data addressed to other machines can be captured using the packet sniffer unlike standard network hosts that receive traffic sent specifically to them. Packet sniffers were formerly known to be very expensive dedicated hardware devices, but new advances in technology have allowed for the development of software network analyzers making it more convenient and affordable.

Packet sniffers can be standalone hardware device with specialized software, or it can simply be software that can be installed on a computer system [8]. They are available both free and commercially with the major difference depending on features such as the number of supported protocol that can be decoded, the user interface and statistical capabilities. A packet sniffer is a combination of hardware and software.

Analysis of a network can be carried out for either good or to harm a network and as such, packet sniffers which are tools, and like all tools they can be used to perform these nefarious acts. When used by malicious individuals, sniffers can present a significant threat to the security of a network [8]. On the contrary, packet sniffers can also be used in penetration testing which has been defined as the legal and authorized attempt to locate and successfully exploit computer systems for the purpose of making systems more secure from malicious attacks/threats. Penetration testing is known to determine how a system reacts to an attack, whether or not a system’s defense can be breached and what information can be acquired from the testing. The passive nature of the packet sniffer makes detecting them on the network difficult.

As a program, the packet sniffer runs in a network attached device that passively receives all data link layer frames that passes through the device’s network adapter. The packet sniffer as stated captures data addressed to other machines, saving it for later analysis. Over the years, several approaches have been taken into consideration to determine the best way to keep track of network. The packet sniffing method is one of such ways and has been used in several ways. There are different types of network sniffing tools depending on the network, application or protocols are available in markets. Here, the primary and most useful packet sniffer is considered.

Wireshark by Gerald Combsis a packet analyzer used for network troubleshooting, analysis. Originally named Ethereal, in May 2006 the project was renamed Wireshark due to trademark issues. It is cross-platform, using pcap to capture packets and runs on various Unix-like operating systems, Solaris, and on Microsoft Windows. It allows the user to put the network interfaces that support promiscuous mode into that mode, in order to see all traffic visible on that interface, not just traffic addressed to one of the interface's configured addresses and broadcast/multicast traffic [9][11]. Wireshark is software that "understands" the structure of different networking protocols. Thus, it is able to display the encapsulation and the fields along with their meanings of different packets specified by different networking protocols. It provides users the capability of capturing the packets traversing over an entire network on a particular interface at a particular time. However, wireshark has limitation of not reporting some malicious intents performed on a network [10][12].

Tcpdump by McCanne, Leres and Jacobson is a common packet analyzer that runs under the command line. It allows the user to intercept and display TCP/IP and other packets being transmitted or received over a network to which the computer is attached. It works on most Unix-like operating systems: Linux, Solaris, BSD, and Mac OS. In those systems, tcpdump uses the libpcap library to capture packets. The port of tcpdump for Windows is called WinDump and uses WinPcap- the Windows port of libpcap. It analyzes network behavior, performance and applications that generate or receive network traffic. It can also be used for analyzing the network infrastructure itself by determining whether all necessary routing is occurring properly, allowing the user to further isolate the source of a problem. It is also possible to use tcpdump for the specific purpose of intercepting and displaying the communications of another user or computer [9].

Capsa is a network analyzer for both LAN and WLAN which performs real-time packet capturing, 24/7 network monitoring, advanced protocol analysis, in-depth packet decoding and automatic expert diagnosis. It provides a comprehensive and high-level visibility to your entire network, helps network administrators or network engineers quickly pinpoint and resolve various application problems, and therefore enhance end user experience and guarantee a productive network environment. It can identify and analyze more than 300 network protocols, as well as network applications based on the protocols [10][12].

Table1: Characteristic Comparison of TCPdump, Wireshark, Capsa and EtherApe

S/N	**Property**	**Wireshark**	**TCPDump**	**Capsa**	**Ether Ape**
1.	OS Supported	Windows and Unix	Unix based	Windows	Unix based
2.	Disk Usage	81MB (Windows) & 449MB (Unix)	448KB	32MB	
3.	Cost	Free	Free	Charged	Free
4.	Open Source	Yes	Yes	No	Yes
5.	Display protocol in OSI Layer structure	Yes	No	Yes	Yes
6.	Libpcap used	Yes	Yes	Yes	Yes
7.	User Interface	GUI/CLI	CLI	GUI	GUI
8.	Creator	The Wireshark Team	The Tcpdump Team	Colasoft	Juan Toledo

Common types of sniffing methods in use are IP based sniffing, MAC based sniffing and ARP based sniffing which does not put the NIC in promiscuous mode. It is an effective method for sniffing in switched environment [10][13]. Several factors or parameters influence the performance of packet capturing. Packet size is one of such factors yet there is no right answer to the question of which packet size gives best performance [11][14]. Short packets are known to increase load on devices while long packets increase load on the network which means the less the long packets, the less stressed a network. Long packets means high ratio of packet payload and packet headers. Tools which support maximum medium sized packets are better tool on this benchmark. Moreover, throughput refers to the amount of data a system processes in bits per second (bps). So, the tool with higher throughput would give better performance but those of large range of rapidly changing throughput leads to random changes in the throughput are not good with respect to the network performance. On the other hand, tools with a range in a pattern are good for the network and show a consistent behavior [15].

Other performance indicators are packet drop rate and response time. Reasons for excessive packet loss could be due to lack of buffer so that all packets coming to the NIC could not be saved [12][13]. However, response time is the time taken to receive acknowledgements between the communicating nodes. Less response time indicates less number of retransmissions and less response time implies better performance. Many network administrators spend a lot of time trying to know what is degrading the performance of their network [14][16]. A typical solution to congestion problem could be an upgrade to the network infrastructure, that is, replace servers with high end servers and increase the bandwidth. However, this solution is expensive, short term and is not entirely scalable seeing that after the upgrade is performed; the congestion problem improves for a while and will later gradually deteriorate as the users change their behavior in response to the upgrade.

The alternative solution to this problem is to deploy a scalable network traffic monitoring and analysis system, in order to understand clearly the dynamics of the traffic and the changes in the internet together with the overall stability of the network. In addition to knowing the health status of the network, monitoring of network activity also has the benefits of detecting denial of service (DoS) and bandwidth theft attacks. In order to conduct analysis of wide range of network behaviors, it is necessary to collect network traffic on a continuous basis rather than as a onetime event which only captures transient behaviors that provides insight into network problems. Collection of long term network traffic data will provide valuable information for improving and understanding the actual network dynamics [17][18].

III. MATERIALS AND METHODS

The model design will not disturb the network traffic or make any changes to any of the packets but will only copy and analyze them. Data flow diagrams are as shown in the figures 1, 2 and 3.

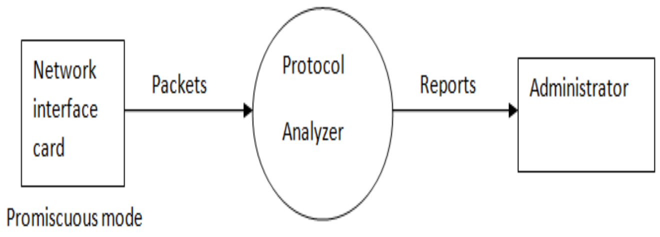


Figure 1: Level 0 DFD

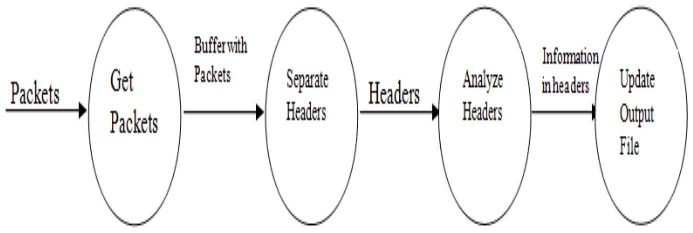


Figure 2: Level 1 DFD-Data flow diagram for packet sniffer

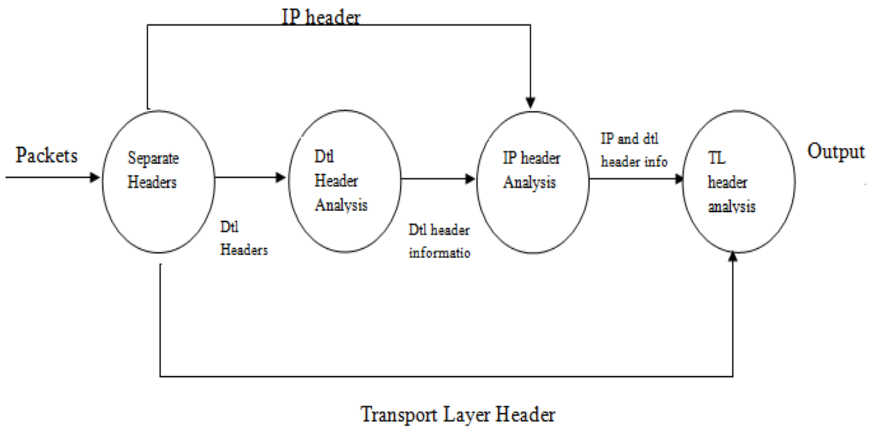


Figure 3: Level 2 DFD-Data Flow Diagram for the protocol Analyzer Process

In the level 0 (Figure 1) data flow diagram- the context diagram, illustrates that the inputs are the packets travelling across the network interface that is set to promiscuous mode and the output is the information contained in the packets in human readable form. The level 1 and level 2 DFD (Figure 2 and Figure 3 respectively) encompass the entire process. In the diagrams, the input is obtained as packets from the network interface by a ‘Get packets’ process. For this process to occur, a packet socket is defined and the raw packets are obtained from the network interface and it is stored in a buffer. The buffer which now contains the packets is passed to the ‘separate header’ process, which strips off various headers of the packet into Internet Protocol (IP) header and Transmission Control Protocol (TCP) header and passes them to ‘analyze headers’ process where they are analyzed and the information is passed on to the ‘update output file’ process. Here the output file will be updated with the latest information obtained from the later processes. The most abstract inputs are the stripped off headers and the most abstract output is the information in the headers in human readable form.

The Use case diagrams model the functionality of the system and use cases. It describes a sequence of actions that provide a measurable value to an actor and is drawn as a horizontal ellipse. An actor is a person, an organization or an external system that plays a role in one or more of the interactions with the system.

The Use Case diagram for the proposed software is as shown in the figure below:

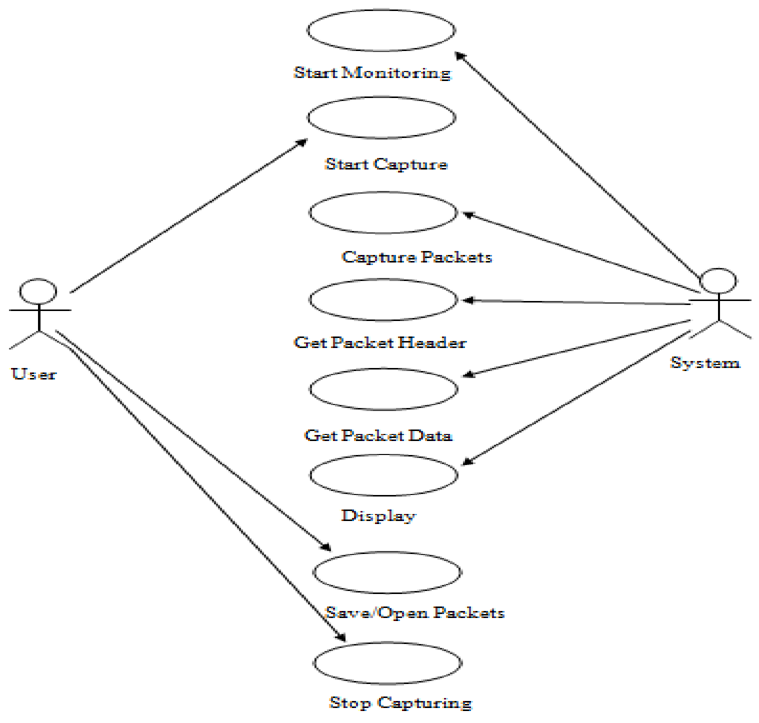


Figure 4: Use Case Diagram of Packet Sniffer

For the Use Case diagram above in a scenario, the system is first initiated to start monitoring. The user then presses the capture button enabling the system to capture packets. The system gets packet header and packet data and saves and opened the packet data. The user presses the stop capturing button.

One of such performance evaluation is that of calculating packet loss which occurs when one or more packets of data travelling across a computer network fail to reach their destination. A packet loss of 0.1% in TCP/IP protocols is generally tolerated since 1 lost packet in every 1000 packets is acceptable. However, anything higher is expected to have a negative impact and would need to be addressed. By definition from [1]; the packet loss ratio is calculated as;

$$\textit{Packet loss ratio} = \frac{\text{Number of Lost Packets}}{(\text{Number of lost packet} + \text{Number of packets received sucessfully})} \quad (1)$$

Throughput has previously been defined as the amount of data a system processes is throughput or bits per second (bps). It is traditionally defined as the ratio of an amount of data transmitted over the time required to transfer it. With the consideration that the data is successfully received, the throughput's general formula by definition from [16] is given as:

$$\textit{Throughput} = \frac{\text{Received Data}}{\text{Transmission Time}} \quad (2)$$

The theoretical throughput is calculated for a single user so that there is no collision in the network. A data size (D) was taken 1498 bytes (1478 UDP payload and UDP header plus 20 bytes of IP header). The general formula for throughput can then be used to prepare the theoretical throughput.

$$\textit{Throughput based on a data size} = \frac{D}{T} = \frac{1498\ x\ 8\ bits}{T} \quad (3)$$

Where;

D is size of packet

T is time taken to transmit the data with its associated headers until next transmission of another data frame.

A relationship however exists between the two properties, packet loss ratio and Throughput. This relationship is shown below:

Let,

$L_r = Packet\ loss\ ratio$

$N_s = Number\ of\ Packets\ sent$

$N_r = Number\ of\ Packets\ received$

$t = Transmission\ time$

Then packet loss ratio can be denoted as: $L_r = \frac{N_s - N_r}{N_s}$ (4)

Dividing through by N_s gives:

$$L_r = 1 - \frac{N_r}{N_s} \qquad (5)$$

Throughput based on the number of packets received will be:

$$\beta = \frac{Nr}{t} \qquad (6)$$

From (5) N_r is given as in (7)

$$N_r = N_s(1 - L_r) \qquad (7)$$

Put (7) in (6) gives (8) as

$$\beta = \frac{\mathrm{Ns}(1 - \mathrm{Lr})}{\mathrm{t}} = \frac{\mathrm{Ns}}{\mathrm{t}}(1 - Lr) \qquad (8)$$

Where,

$\frac{Ns}{t}$ denotes the transmission rate and

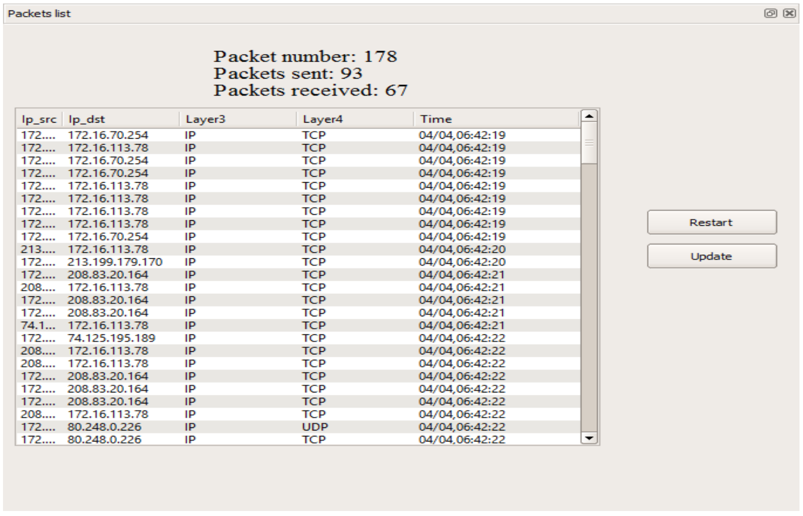
$(1 - Lr)$ is the ratio of packets that are not lost.

Then, it can be said that throughput is a product of two independent events that is transmission rate and the ratio of packets that are not lost.

IV. IMPLEMENTATION AND TESTING

The software requirements are: application environment: Microsoft Windows Vista/7/8/8.1; PyWin32 64bits: Necessary to access the network card list; Qt library for GUI; Impacket-0.9.9.9; numpy-1.9.2: Necessary to display number of packets and perform mathematical computations; Pyparsing-2.0.3: Necessary to parse packets when using Pycap; Matplotlib-1.4.3: Necessary in plotting graphs when using Python. Hardware Requirement: Pentium IV processor and above at 750MHZ or faster; Network Interface Card and Video Graphic Adapter (VGA). The Python version used to design this project is Python 2.7.8 and the capture library incorporated with this version of Python is the Pycap-0.10.5. The Winpcap software also installed allows for the capture of the packets by the capture library and the PyWin32 64bits is necessary to access the network card lists.

The packet sniffer can sniff as many packets as possible on the network. It was implemented with an analysis test on the network to determine the performance of the network by studying the packet loss ratio with the total number of packets that arrive on the network. Performance evaluation was carried out using the loss ratio graph and TCP loss ratio graph with the loss ratio graph comparing overall packet loss ratio to total number of packets and the TCP loss ratio graph comparing TCP packet loss ratio to total number of TCP packets. The two graphs are vital in showing how well the network performs and how efficiently packets are being delivered and received by their respective addresses. The packets are listed in the packet list window showing at a glance the number of packets captured at each instant of time and also displays the source and destination IP address together with the layer 3 and layer 4 protocols and the time at which the packet reaches its destination as shown in Figure 5.

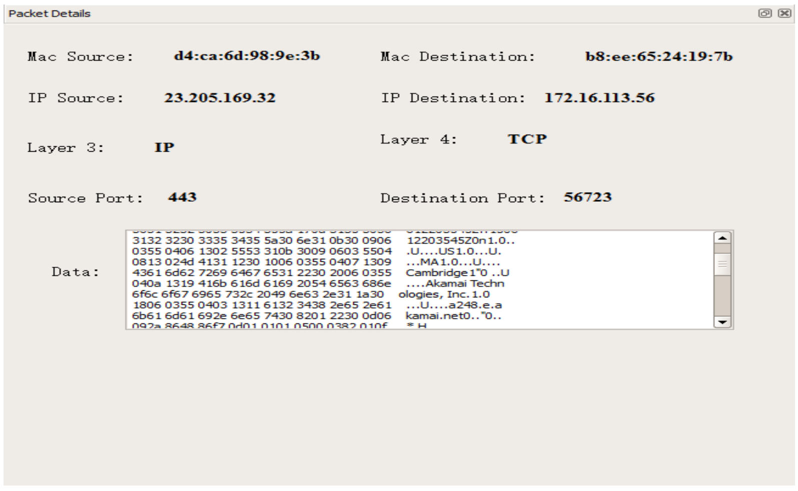


Packet number: 178
Packets sent: 93
Packets received: 67

Ip_src	Ip_dst	Layer3	Layer4	Time
172....	172.16.70.254	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.70.254	IP	TCP	04/04,06:42:19
172....	172.16.70.254	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.70.254	IP	TCP	04/04,06:42:19
213....	172.16.113.78	IP	TCP	04/04,06:42:20
172....	213.199.179.170	IP	TCP	04/04,06:42:20
172....	208.83.20.164	IP	TCP	04/04,06:42:21
208....	172.16.113.78	IP	TCP	04/04,06:42:21
172....	208.83.20.164	IP	TCP	04/04,06:42:21
172....	208.83.20.164	IP	TCP	04/04,06:42:21
74.1...	172.16.113.78	IP	TCP	04/04,06:42:21
172....	74.125.195.189	IP	TCP	04/04,06:42:22
208....	172.16.113.78	IP	TCP	04/04,06:42:22
208....	172.16.113.78	IP	TCP	04/04,06:42:22
172....	208.83.20.164	IP	TCP	04/04,06:42:22
172....	208.83.20.164	IP	TCP	04/04,06:42:22
172....	208.83.20.164	IP	TCP	04/04,06:42:22
208....	172.16.113.78	IP	TCP	04/04,06:42:22
172....	80.248.0.226	IP	UDP	04/04,06:42:22
172....	80.248.0.226	IP	TCP	04/04,06:42:22

Figure 5: Packet list window displaying packets

Each packet has a source and destination Mac Address, Ip Address and port number and the packet header is broken down to get the details of each packet clicked so analysis can be performed on each packet as shown in Figure 6.



Mac Source: **d4:ca:6d:98:9e:3b**   Mac Destination: **b8:ee:65:24:19:7b**

IP Source: **23.205.169.32**   IP Destination: **172.16.113.56**

Layer 3: **IP**   Layer 4: **TCP**

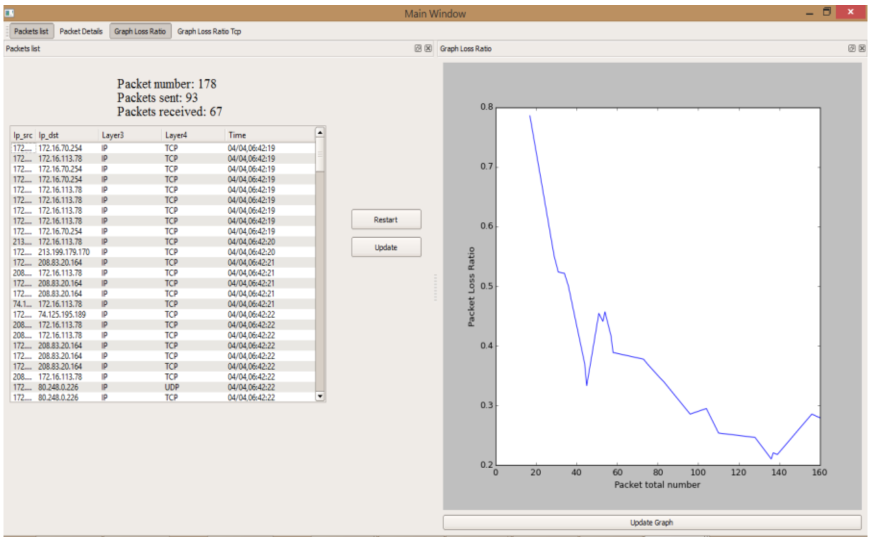
Source Port: **443**   Destination Port: **56723**

Data:

```
3132 3230 3335 3435 5a30 6e31 0b30 0906   1220354520n1.0..
0355 0406 1302 5553 310b 3009 0603 5504   .U....US1.0...U.
0813 024d 4131 1230 1006 0355 0407 1309   ...MA1.0...U....
4361 6d62 7269 6467 6531 2230 2006 0355   Cambridge1"0 ..U
040a 1319 416b 616d 6169 2054 6563 686e   ....Akamai Techn
6f6c 6f67 6965 732c 2049 6e63 2e31 1a30  ologies, Inc.1.0
1806 0355 0403 1311 6132 3438 2e65 2e61   ...U....a248.e.a
6b61 6d61 692e 6e65 7430 8201 2230 0d06   kamai.net0.."0..
092a 8648 86f7 0d01 0101 0500 0382 010f   *.H
```

Figure 6: Packet details of packet from 23.205.169.32

As data is being transmitted, one or more packets may fail to reach their destination and as previously stated, the graph plots the packet loss ratio against the total number of packets to monitor packet error rate in the network.

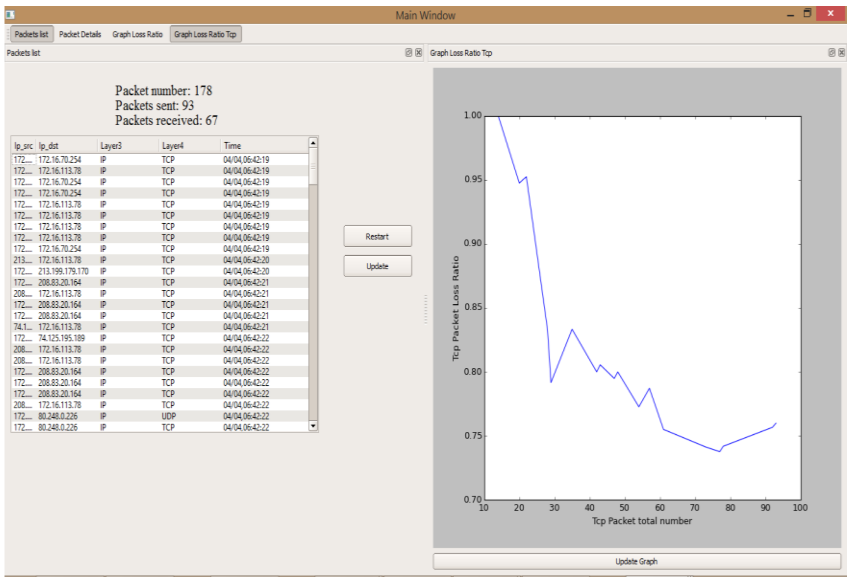


Packet number: 178
Packets sent: 93
Packets received: 67

Ip_src	Ip_dst	Layer3	Layer4	Time
172....	172.16.70.254	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.70.254	IP	TCP	04/04,06:42:19
172....	172.16.70.254	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.70.254	IP	TCP	04/04,06:42:19
213....	172.16.113.78	IP	TCP	04/04,06:42:20
172....	213.199.179.170	IP	TCP	04/04,06:42:20
172....	208.83.20.164	IP	TCP	04/04,06:42:21
208....	172.16.113.78	IP	TCP	04/04,06:42:21
172....	208.83.20.164	IP	TCP	04/04,06:42:21
172....	208.83.20.164	IP	TCP	04/04,06:42:21
74.1...	172.16.113.78	IP	TCP	04/04,06:42:21
172....	74.125.195.189	IP	TCP	04/04,06:42:22
208....	172.16.113.78	IP	TCP	04/04,06:42:22
208....	172.16.113.78	IP	TCP	04/04,06:42:22
172....	208.83.20.164	IP	TCP	04/04,06:42:22
172....	208.83.20.164	IP	TCP	04/04,06:42:22
172....	208.83.20.164	IP	TCP	04/04,06:42:22
208....	172.16.113.78	IP	TCP	04/04,06:42:22
172....	80.248.0.226	IP	UDP	04/04,06:42:22
172....	80.248.0.226	IP	TCP	04/04,06:42:22

Figure 7: Docked window displaying packet list and Graph loss ratio

The TCP graph loss ratio plots the TCP packet loss ratio to the total number of TCP packets.



Packet number: 178
Packets sent: 93
Packets received: 67

Ip_src	Ip_dst	Layer3	Layer4	Time
172....	172.16.70.254	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.70.254	IP	TCP	04/04,06:42:19
172....	172.16.70.254	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.113.78	IP	TCP	04/04,06:42:19
172....	172.16.70.254	IP	TCP	04/04,06:42:19
213....	172.16.113.78	IP	TCP	04/04,06:42:20
172....	213.199.179.170	IP	TCP	04/04,06:42:20
172....	208.83.20.164	IP	TCP	04/04,06:42:21
208....	172.16.113.78	IP	TCP	04/04,06:42:21
172....	208.83.20.164	IP	TCP	04/04,06:42:21
172....	208.83.20.164	IP	TCP	04/04,06:42:21
74.1...	172.16.113.78	IP	TCP	04/04,06:42:21
172....	74.125.195.189	IP	TCP	04/04,06:42:22
208....	172.16.113.78	IP	TCP	04/04,06:42:22
208....	172.16.113.78	IP	TCP	04/04,06:42:22
172....	208.83.20.164	IP	TCP	04/04,06:42:22
172....	208.83.20.164	IP	TCP	04/04,06:42:22
172....	208.83.20.164	IP	TCP	04/04,06:42:22
208....	172.16.113.78	IP	TCP	04/04,06:42:22
172....	80.248.0.226	IP	UDP	04/04,06:42:22
172....	80.248.0.226	IP	TCP	04/04,06:42:22

Figure 8: Docked window displaying packet list and TCP graph loss ratio

V. CONCLUSION

The packet sniffer software has been developed using Python programming language and has been implemented on a Windows platform since previous Packet Sniffer software designed with Python have only been implemented on Linux platform. The packet sniffer software precisely captures data packets. It converts and displays data into comprehensible format to enable reading of traffic and has minimized significantly the difficulty in navigating through previous one. The performance evaluation was carried out using packet loss ratio and TCP packet loss ratio monitor network performance from the users end.

REFERENCES

[1] PlallaviAsrodia, Vishal Sharma “Network Monitoring and Analysis by Packet Sniffing Method”, International Journal of Engineering Trends and Technology (IJETT)-volume 5-May 2013.

[2] Tabu S. Kondo, Leonard J. Mselle “Penetration Testing with Banner Grabbers and Packet sniffers”, Journal of Emerging Trends in computing and information sciences-volume 5, No 4-April 2014.

[3] S. Ansari, Rajeev S.G. and Chandrasekhar H.S., “Packet Sniffing: A Brief introduction”, IEEE *Potentials,* Dec 2002-Jan 2003, Volume: 21 Issue 5.

[4] PallaviAsrodia, Hemlata Patel, “Analysis of Various Packst Sniffing Tools for Network Monitoring and Analysis”, International Journal of Electrical, Electronics and Computer Engineering **1** (1): 55-58(2012).

[5] Awodele Oludele, Otusile Oluwabukola, “The Design and Implementation of a packet sniffer (PSniffer) Model for Network Security”, International Journal of Electronics Communication and Computer Engineering-volume 3, issue 6, ISSN (online): 2249-071X, ISSN (Print): 2278-4209.

[6] Ashish Kulkarni, Nimish Kate, Rohit Ghadshi, Rushikesh Date, “Signature Based Packet Sniffer”, International Journal of Advance Research in Computer Science and Management Studies, Volume 2, issue 3, March 2014.

[7] Rupam, Atul Verma, Ankita Singh, “An Approach to Detect Packets using Packet Sniffing”, International Journal of Computer Science and Engineering Survey (IJCES) Vol. 4, No. 3, June 2013.

[8] Mohammed Abdul Qadeer, Mohammad Zahid, Arshad Iqbal, MisbahurRahman Siddiqui, “Network Traffic Analysis and Intrusion Detection using Packet Sniffer”, 2010 Second International Conference on Communication Software and Networks.

[9] A. Dabir, A. Matrawy, "Bottleneck Analysis of Traffic Monitoring Using Wireshark", 4th International Conference on Innovations in Information Technology, 2007, IEEE Innovations '07, 18-20 Nov. (2007), Page(s): 158-162(2007).

[10] SB .A. Mohammed, Dr.S.M Sani, Dr. D.D. Dajab, “Network Traffic Analysis: A Case Study of ABU Network”, Computer Engineering and Intelligent Systems ISSN 2222-1719 (Paper) ISSN 22, Vol.4, No.4, 2013.

[11] Otusile Oluwabukola, Awodele Oludele, A.C Ogbonna, Ajeagbu Chigozirim, Anyeahie Amarachi, “A Packet Sniffer (PSniffer) Application for Network Security in Java”, Issues in Informing Science and Information Technology, Volume 10, 2013.

[12] Dr. Charu Gandhi, Gaurav Suri, Rishi P. Golyan, Pupul Saxena, Bhavya K. Saxena, “Packet Sniffer – A Comparative Study ”, International Journal of Computer Networks and Communications Security , Vol. 2, NO. 5, MAY 2014, 179–187, Available online at: www.ijcncs.org ISSN 2308-9830

[13] Lothar Braun, Alexander Didebulidze, Nils Kammenhuber, Georg Carle, “Comparing and Improving Current Packet Capturing Solutions based on Commodity Hardware”, Institute for Informatics Chair for Network Architectures and Services.

[14] A. Shah, D. Bhatt, P. Agarwal, and P. Agarwal, “Effect of Packet-Size over Network Performance”, International Journal of Electronics and Computer Science Engineering, Vol. 1, pp. 762-766, 2012.

[15] S. N. John, F. A. Ibikunle, Adeyinka A. Adewale, “Performance Improvement of Wireless Network Based on Effective Data Transmission” IET International Conference on Wireless, Mobile and Multimedia Networks, 11-12 January 2008, IEEE Xplore Digital Library Journal: ISSN 0537-9989, pp.134-137, @ IEEE Xplore Digital Library, 2008, USA.

[16] Adeyinka A. Adewale, Isaac Samuel, Awelewa A. A, Dike U. Ike, “Design and Development of a Microcontroller Based Automatic Switch for Home Appliances” International Journal Engineering Science Invention ISSN (Online): 2319-6734, www.ijesi.org Volume 2 Issue 10, October, 2013 pp24-31.

[17] Usha Banerjee, Ashutosh Vashishtha, Mukul Saxena, “Evaluation of the Capabilities of WireShark as a tool for Intrusion Detection”, International Journal of Computer Applications (0975 – 8887) Volume 6– No.7, September 2010.

[18] Md. Kamrul Hasan, A.H.M. Amimul Ahsan, and M.Mostafizur Rahman, “IEEE 802.11b Packet Analysis to Improve Network Performance”, JU Journal of Information Technology (JIT), Vol. 1, June 2012.