Development of a network packet sniffing tool for internet protocol generations

Article *in* International Journal of Cloud Computing · August 2020

DOI: 10.1504/IJCC.2020.109378

1 author:

Ruwaidah Albadri
Al-Furat Al-Awsat Technical University
5 PUBLICATIONS **4** CITATIONS

SEE PROFILE

Development of a network packet sniffing tool for internet protocol generations

Ruwaidah Fadhil Albadri

Information and Communications Technology ICT Department,
Technical Institute of Samawa,
Al-Furat Al-Awsat Technical University,
Al-Zahra Neighborhood, Highway, Al Muthanna, Iraq
Email: ins.rod@atu.edu.iq

Abstract: Packet sniffing is a way to take advantage of each packet as it flows across the network. One of the most complex problems that face the network administrators is the network analysing. The information provided by existing tools for network traffic analysis is very small and considered enormous data if they were all stored for later analysis which make it difficult to be analysed. This paper aims to propose a sniffing tool to capture the packets of both IPv4 and IPv6. The proposed tool works to access the captured packets by using socket class in visual studio. The first scenario is to analyse IPV4 by capturing the packets and identifying the used ports, protocols, and the packets. However, the second scenario is to analyse IPV6 in the same way. The results are varied from IPv4 to IPv6 by the number of captured protocols and the used ports for both source and destination.

Keywords: wide-area network emulator; protocols and length of data; logical connection; QoS configuration; TCP/IP sessions; TCP/IP based networks; IPv4 and IPv6; C# programming language.

Reference to this paper should be made as follows: Albadri, R.F. (2020) 'Development of a network packet sniffing tool for internet protocol generations', *Int. J. Cloud Computing*, Vol. 9, Nos. 2/3, pp.232–244.

Biographical notes: Ruwaidah Fadhil Albadri is an Assistant Lecturer in Electrical Department, Technical Institute of Samawa, Al-Furat Al-Awsat Technical University. She has Master's in Computer Sciences (MSc) from Jamia Hamdard 2012/2013 New Delhi, India. She has published a journal paper in *Journal of Contemporary Medical Sciences (JCMS)* 'Decision support detection system for lung nodule abnormalities based on machine learning algorithms'. Her research interests focus mainly on machine learning algorithms and decision support systems.

1 Introduction

A packet sniffing tool is used by the managers and administrators of networks to monitor the transmitted data over the network. Packet followers are used for network security and network management, and unauthorised users can use it to steal information from the network (Xu et al., 2016). Packet analyser can display a wide range of information sent over the network as well as the associated network (Singh and Kumar, 2018). There are

packet sniffers in the form of hardware or software, and they can capture both incoming and outgoing network traffic and use screen and username and passwords along with other sensitive information (Anu and Vimala, 2017).

A packet sniffer allows configuring the network interface to display all information transmitted over the network. When data sent or received through the network passes, it is possible to capture these data for analysis by using sniffing tools (Chauhan and Sharma, 2014). A sniffer objects transmitted data and captures the packets, then comparing it with the header formats in order to identify the standards parts and take the required parts like used ports, addresses and etc. There are a large number of tools for network analysis that can be used by everyone where the user's purpose cannot be determined whether it is for good purpose or useful or harmful purpose (Davis and Clark, 2011). For example, a program that can capture users' passwords can be used by the hackers, but the network administrator may use the same tool to find network statistics such as available bandwidth. Sniffer may also be useful for web filters or testing firewall or exploring client/server relationships (Elsen et al., 2015).

At present, network-related topics need practical labs to learn and understand protocol behaviour (Gandhi et al., 2014). In order to understand these protocols, we need to create an appropriate environment for testing the protocols used and verifying their behaviour under different circumstances. In this paper, we are going to implement an environment that enables to analyse network components, such as packet loss or delay. The best method to modify the conditions of these networks is by using a software tool, which allows to do so in a simple way; this tool is a wide-area network emulator (that is known as Sniffer) (Oluwabukola et al., 2013; Jaisinghani et al., 2017). Once this tool is programmed and setup, some tests will be performed to analyse network elements by analysing and distinguishing the protocols used in the transceiver operations. In addition, the packet will be analysed in detail to display the main message components by specifying the values of the variables. This will result in a complete network analysis to understand the behaviour of the network and users by identifying and collecting some statistics on the most used protocols and length of data. The proposed tool in this work helps to analyse the network and gives a clear live statistic about the network with no waste of storage.

2 Literature review

Nishanth and Babu (2014) proposed an approach of defence against 'hijacking attacks' at the transport layer level. A logical connection to the TCP layer must be created then the web server and the client can connect, it is known as a triple handshake. The parties synchronise their sequence numbers during this process. It depends on changing the sequence number throughout the connection. The idea is that the sequence number must be recalculated again each time a new packet is sent. This makes it more difficult for the attackers to predict or guess the sequence number during a 'hijacking attack'. The disadvantage of this method is that if the attacker can gather a large number of packets, he can break the algorithm to change the sequence number.

Poonkuntran and Arun (2014) suggested a method to detect fake access points by using the technique of honey pots. The honey pots will be an extraneous access points on the wireless networks. The honey pot will be used as a tool for gathering information or

evidence and for attack patterns that are used by attackers. The honey pots will be used to attract and redirect the attackers to the network trap system. The honey pots only detect the fake access points, while the identity of the regular stations will remain undetected.

Singh et al. (2012) proposed a (cross-layer IDS) intrusion detection system for wireless networks that combines the weight value of the received signal strength (RSS) and the time it takes (TT) to request transmission and transmit transmission packets. In technology, the TT and RSS values are captured and monitored on the server. During the process of detection, The IDS will raise the alarm when the combined weight of the station is greater than the specified limit. The technology depends on the value of the threshold to reach an accurate result. This method produces false negatives when the leg is legitimate silence.

Atlas (Qazi et al., 2013) is a collective outsourcing method for discovering granular applications from mobile agents. The trainer is then sent to an automated learning plane within the control plane for classification. Mekky et al. (2014) have proposed extensive application architecture have been proposed for SDN systems aware of the generalisation of redirection abstractions, including information from 4 to 7. To improve efficiency, their implementation increases application logic in converters. In addition, FlowQoS (Seddiki, 2014) is a reference design that implements application-based service quality by delegating application identification and QoS configuration to the SDN controller (Tsai et al., 2018).

3 The related works

In this section we explore some of related works in order to address the main points that we can improve in our work. The techniques that were presented in 2010 in Qadeer et al. (2010) focuses on the basics of sniffer package and its work, developing the tool on the Linux platform and using it for intrusion detection. It also discusses ways to detect the existence of such programs on the network and deal with them in an effective manner. Emphasis was also placed on analysing the emerging network bottleneck scenario, using this self-developed packet tool. Before the development of this original program, a careful observation was made about the work behaviour of the already existing sniffer program such as Wireshark (formerly known as ethereal), tcpdump, and snort, which serve as a basis for developing its sniffer program. To capture packets, a library known as libpcap was used. In order to minimise the risks to be detected and captured, an increasing number of hackers tend to use a springboard to launch their attacks. Because there are many sophisticated intrusion detection approaches proposed, many hackers avoid detection by processing TCP/IP sessions. The tool proposed in Yang et al. (2018), first introduces how to generate code for inhaling computer network traffic because most methods need to intercept TCP/IP packets to detect break-in intrusion. Unlike the use of packet sniffing tools, self-made code for sniffing network packets can be easily integrated into different detection methods. It helps detect intruders on the network by analysing packets sent over the network between two points and comparing the results after they are stored on the computer. In addition, sniffing tools are very useful in wireless sensor networks WSN. Because WSNs are typically deployed in a harsh environment, the number of sensors is large and a single node resource is limited, making WSNs vulnerable. Therefore, in order to support normal operation, WSNs need real-time monitoring tools. In this article (Zhao et al., 2012), a network monitoring and packet

sniffing tool for wireless sensor networks (NSSN) is presented and implemented. NSSN can capture radio packets from nearby nodes using NSSNer nodes, so they can monitor network status, find network problems, and improve network configuration without interfering with WSN networks. The functions applied by NSSN include network monitoring, protocol analysis and presentation, network diagnostics, performance measurement, data mining, and statistical analysis. Although NSSN has good performance, it cannot avoid common deficiencies in the sniffer technology, for example because of differences in RF circuit parameters and uncertainty in wireless communications, while the sniffer node succeeds in capturing a beam, it is not certain the adjacent sensor contract can receive the packets successfully and vice versa.

4 Methodology

The packet is a very large data carrier in TCP/IP based networks. In the packet exchange network, active information is broken down and encoded into packets. The source nodes send packets that include destination and source addresses to the access point. When the destination is acquired for packets, the decryption and aggregation is performed to obtain the expected data. In this paper we work on a suggestion tool for sniffing and network packet capturing. The tool captures the packet and analyses its contents for the purpose of understanding the network impediments that cause delays in transmission, reception or work in general. The delay in network operation may be due to congestion by packets on a given port. After the study and deep analysis of the data it will be easy for network administrators to propose alternative methods and techniques for the current method to solve the current problem. Even if there are no obvious problems in the network, as the size of the network expands and future work evolves, new problems will arise, so the main task of the proposed tool in this project is to analyse network behaviour for performance improvement or problem solving. Based on the analysis of previous studies, we have found that working on a tool that works on both IPv4 and IPv6 will be an important result that will directly help manage and improve networks both now and in the near future due to the start of IPv6 despite the low spread. The fundamental stage at this work is the implementation of the sniffing tool. This step will be proposed after doing some coding by using C# programming language. Firstly, the program provides the IP addresses of all the connected devices to this network. The selected IP address will be as an instruction to capture the packet and provide some information about it by identifying the direction of the packet whether it is a transmitted or received packet. Furthermore, the used protocol for this packet will be recognised. In addition to the protocol, the basic information of the packet will be explored like check sum, time to leave TTL, data length, source address, destination address, and the port.

4.1 Phase 1: simulation and data gathering

Simulation and data gathering is one of important phase where the real situation is simulated and data gathering of traffics are conducted for the project requirements of both IP versions 4 and 6. The simulation will run traffic with packet generator instead of Wireshark to capture the ideal traffic for pattern analysis. The result of pattern analysis

will be inserted to new sniffing tool in new approach in order to make those traffics monitored and captured by the new sniffing tool.

4.2 Phase 2: design and implementation

Design and implementation as the continuation of previous phase, this phase focuses on the design of the new Sniffing tool. Starting from the specification of system requirements that fulfil the users' needs continued with software engineering-related diagrams, implementations of specified requirements into the source code and compilation of the software. And the last thing is the testing of new sniffing tool with network simulation.

This paper explores a sniffing tool is fully programmed and built to work in accordance with the requirements and specifications prescribed and defined within the objectives of this project. This sniffing tool will be able to work well if the specific steps in the work methods are followed to reach the goal and the end result. The main and fundamental purpose of this tool is to analyse network data and capture network packets in order to get statistics and integrated network analysis. This analysis will directly contribute to improving the network management process by addressing and identifying weaknesses in the network after conducting the necessary analysis. The weakness in the network may not be completely interrupted, but it may be noted that there is a large pressure on one of the ports in the network, which would cause congestion, which leads to the weakening of the network or reduce the speed and thus poor performance and lack of efficiency.

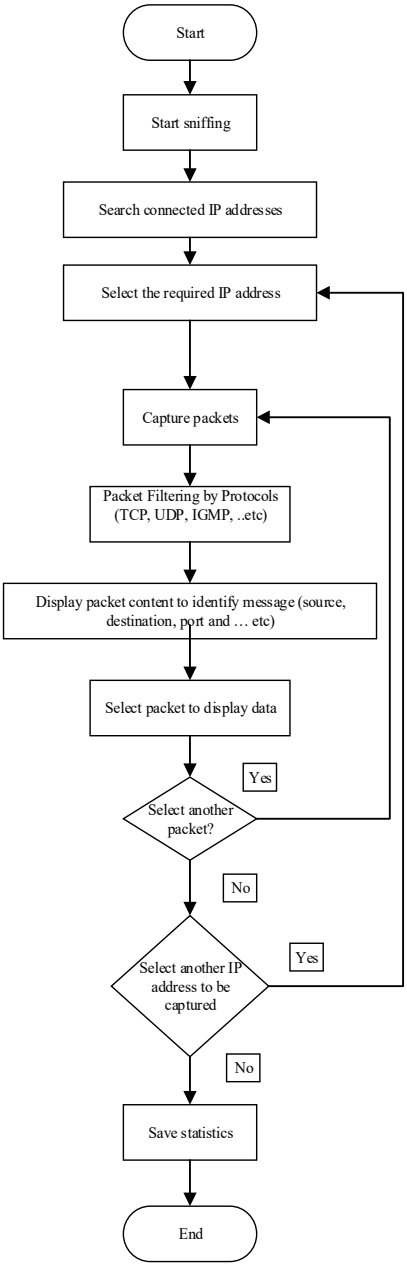
4.3 Phase 3: tool analysis

Project analysis will explain the analysis of traffic monitoring and capturing from previous simulation. The new and other two existing sniffing tools will log everything that it captures into a log file. The log file located within the program folder or home folder depends on the configuration. This file contains information what kind of traffic/packet in IPv4 and IPv6 network, and the source and the destination of traffic. The result within the log file can decide whether the new sniffing tools can monitor or capture the traffic as proposed or not. If not, then this tool must be fixed.

5 Proposed system

The new sniffing tool must capable to de-capsulate two types of traffic, IPv4 and IPv6 in network. The flowchart in Figure 1 is the process of sniffing steps. As shown in the flowchart in Figure 1, the process starts by doing a thorough search of all the devices connected to the network and listing them in a list to be selected by the user. After fetching IP addresses for all devices, whether version 4 or 6, the required IP address is selected and then the buffer size, which is the number of captured packets. The captured data is then analysed to display some information about each packet. This information will be necessary and very important in network analysis such as source IP address, source port, destination IP address, destination port, and protocol. This information will provide greater opportunities for network analysts to understand the network better. Then some statistics are provided by the proposed sniffing tool to display some information in friendly interfaces.

Figure 1 Flowchart of the whole procedure



In our packet sniffer, at user level the packets are copied from the kernel buffer into a buffer created by when a live capture session is created. A single packet is handled by the buffer at a time for the application processing before next packet is copied into it. The new approach taken in the development of our packet sniffer is to improve the performance of packet sniffer, using our proposed tool to use same buffer space between kernel space and application. As it is shown in Algorithm 1, the sniffing process is done

in live mode and gives live statistics about the active ports and the used protocols. The live mode sniffing is more efficient than the traditional approaches since it saves storage and time.

Algorithm 1: Capture selected IP address

Input: Desired IP address and buffer size to start capturing.

Output: Captured packets.

Start

Step 1: Capture all packets that related (received/sent) to the selected IP address.

Step 2: List all the captured packets in the list view.

Step 3: Identify the main contents of the captured packet (such as source IP address, destination IP address, port, and packet size) by comparing the captured packet with the header of packet to identify the required data and display them at the list view of the sniffing tool.

Step 4: If the number of captured packets equal to the buffer size Then breaks, otherwise go to Step 3.

Step 5: Store all the captured packets in the buffer temporarily and continue the live sniffing process.

End

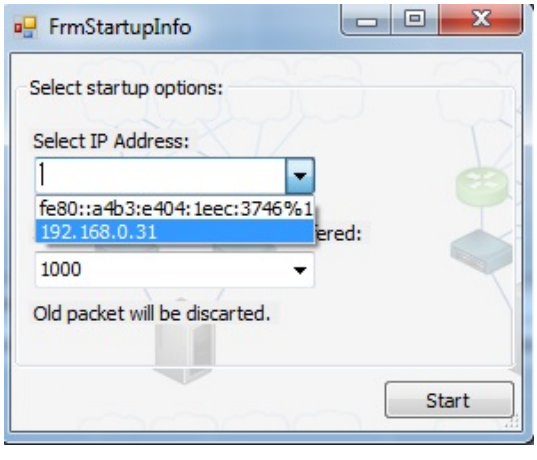
6 Results and discussion

In this section, two scenarios will be tested to discuss the results obtained when implementing the proposed tool.

6.1 Scenario A (IPV4)

In this scenario, IP address version 4 is tested by scanning all IP addresses connected to the network and then specifying the address to start the capturing and sniffing process. As shown in Figure 2, all network-related addresses are scanned and listed in a drop-down list for selection.

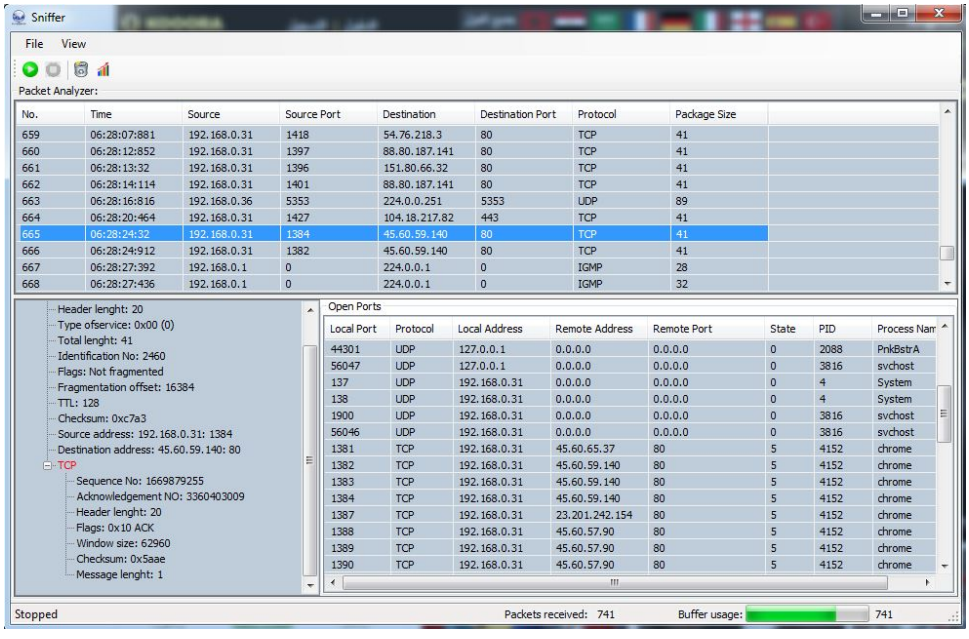
Figure 2 Desired IP address selection (see online version for colours)



As shown in Figure 2, the number of IP addresses on the network. After selecting the desired IP address, the size of the buffer should be specified, this buffer stores captured packets. In other words, the size of the buffer is the number of captured packets.

After selecting the desired basic information such as IP address and buffer size, pressing the start button in the capturing process will show the system's main screen with the beginning of the data for this IP address gradually when new packets are captured automatically displayed on the screen as it is shown in Figure 3.

Figure 3 Main sniffer interface with result of captured packets (see online version for colours)

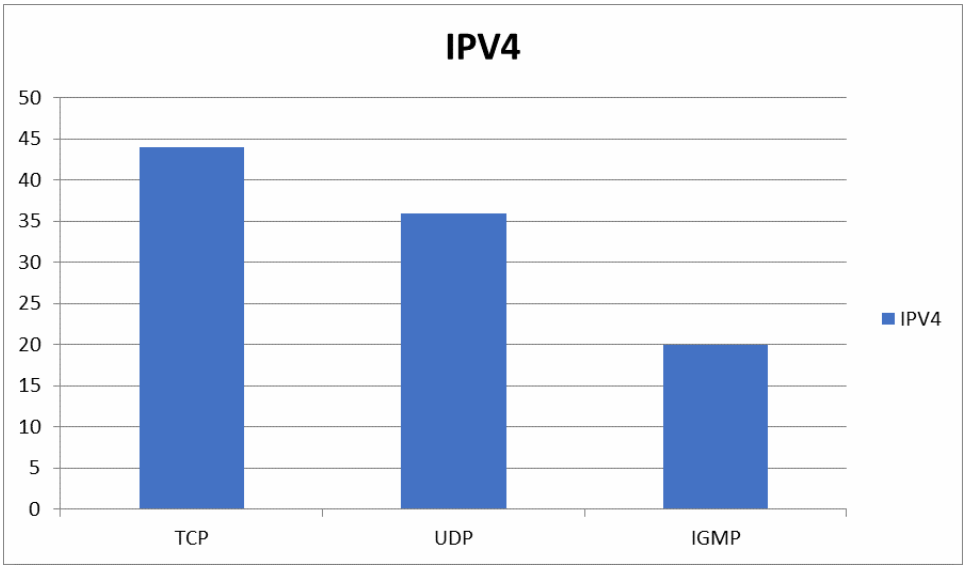


As shown in Figure 3, which represents the results of this scenario where the main interface will be divided into three main sections, each of which displays specific results in addition to the presence of auxiliary conditions such as status bar, which includes progress bar representing the number of packets to the buffer. The bulk of these three sections is the upper part called the ‘packet analyser’. The packet analyser list view contains eight columns, each with its function: (No, Time, Source IP address, Source port, Destination IP address, Destination port, Protocol, Package size). The data is in the form of consecutive rows. Each row represents a captured packet, simple and basic information about it is showed as mentioned previously. The number of captured packets is variable but depends entirely on the selected value of the buffer that is specified in the first interface before you start capturing. The second sub-interface in the program is activated when you press once on any of the captured packets. This interface includes some information that is more in depth and related to the selected packet and this information is very important for the purpose of analysing network information by system engineers. Of this information, for example the version of IP address if it is 4 or 6. In addition to IP packet format information like header length, type of service (TOS), total length, identification no, flags which indicate if the packet is fragmented or not,

fragment offset, TTL, checksum, source IP address, and destination IP address as it is shown in Figure 3.

In addition to this information, there is additional information about the protocol used by this packet either in sending or receiving it. The protocol used in this packet is TCP protocol, so there will be special information about this protocol, for example, sequence no, acknowledgement no, header length, flags, checksum, and message length. This information is variable by changing the type of protocol used by this packet. Our proposed system recognises and analyses four types of protocols: TCP, UDP, ICMP, and IGMP. For example, in this example, there is acknowledgment because the protocol used is TCP, which is a reliable protocol. When you click on any other packet, the data in this sub-interface of the detailed information about the captured packet automatically changes to the information of the new selected captured.

Figure 4 Statistics of the captured packets of IPv4 (see online version for colours)



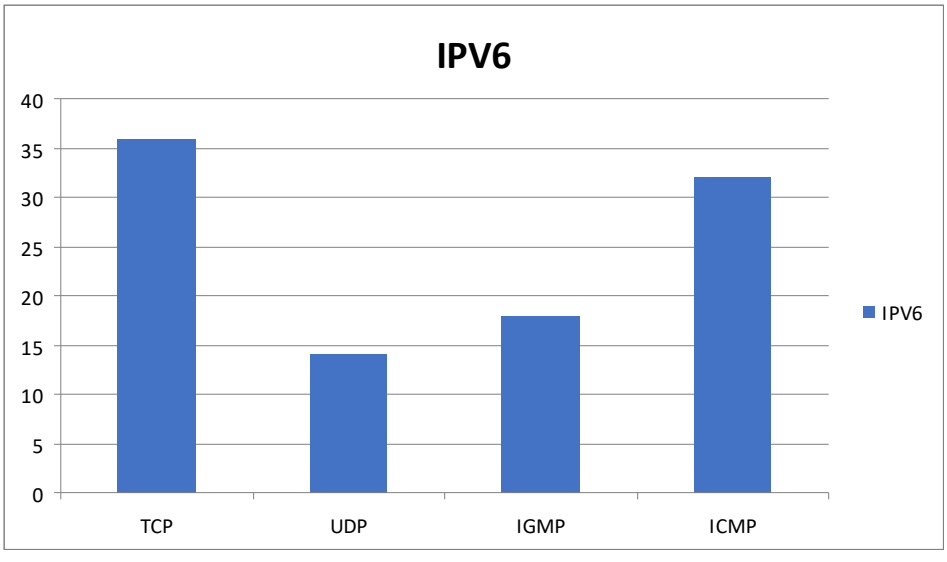
In addition to the used protocols, the system displays the most used ports in the network by displaying the most frequent source port. This is only for source ports from the sample taken to open the statistics window, as well as most frequent destination port. This is for destination ports only within the sample captured from the network packets until the statistics window is opened. This information is important in analysing the network and knowing the pressure areas on any port in the transmission and receiving, which may lead to problems of network congestion. As shown in Figure 4, there are three intercepted protocols that were identified from within the captured sample and estimated at 1,000 packets. These protocols are TCP, UDP, and IGMP where we note that the largest percentage is back to TCP and represented in blue and then followed by the UDP protocol. While the IGMP protocol was the least used within this sample taken from network packets. In addition, port 1901 represents the most frequent source port. However, the port 443 was the most frequent destination port in the captured sample of network packets. These percentages and clear values in the statistics are different and inconsistent. As they vary from one network to another and from time to time based on

the use of the network at that moment. Therefore, all protocols may appear, or some may be lost by use. The ratios may vary and vary significantly. Also, most frequent source port and most frequent destination port will certainly differ based on network usage and sample size taken from network packets.

6.2 Scenario B (IP v6)

Table 1 shows sample of statistics from network packets of selected IPv6. These percentages and clear values in the statistics are different and inconsistent. Where they vary from network to network and from time to time based on the use of the network at that moment so that all protocols may appear or may be absent by some use as well as ratios may be clearly different. Also, most frequent source port and most frequent destination port will certainly differ based on network usage and sample size taken from network packets.

Figure 5 Statistics interface for IPv6 (see online version for colours)



In the sample selected for this scenario the buffer size of 2,000 captured packages was selected. Within this selected sample we note the emergence of four protocols, namely TCP, UDP, ICMP, and IGMP, which have been highlighted by the system and have been used in the network operations both transmission and receiving. Figure 5 shows that the TCP protocol and ICMP protocol use is fairly close, the ICMP is represented by red colour and TCP is represented by light blue colour. While the use of IGMP protocol that is represented in dark blue colour was slightly more than the use of the UDP protocol that is represented in yellow colour.

The most frequent source port was 1742 and the most frequent destination port was 80 for the selected sample of the captured network packets of IPv6.

Table 1 Packet analyser for IPv6

No.	*Time*	*Source*	*Source port*	*Destination*	*Destination port*	*Protocol*	*Package size*
62	06:56:38:772	fe80::a4b3:e404:1eec:3746	1900	2001:e68:5414::0001:22cf	1900	IGMP	74
63	06:56:38:793	fe80::a4b3:e404:1eec:3746	1900	2001:a18:25:11::40	1900	UDP	74
64	06:56:38:854	2001:4860:4860::8888	1901	2001:4860:40::8114	1901	ICMP	90
65	06:56:38:915	fe80::a4b3:e404:1eec:3746	1901	2001:4860:40::8114	1901	IGMP	90
66	06:56:38:974	fe80::a4b3:e404:1eec:3746	1901	2001:4860:40::8114	1901	TCP	90
67	06:56:39:35	2001:4860:4860::8888	1740	2001:4860::25cc:877	80	TCP	133
68	06:56:39:93	fe80::a4b3:e404:1eec:3746	1740	2001:a18:25:11::40	80	ICMP	133
69	06:56:39:153	fe80::a4b3:e404:1eec:3746	1740	2001:4860::25cc:877	80	TCP	133
70	06:56:39:214	2001:4860:4860::8888	1741	2001:a18:25:11::40	80	ICMP	255
71	06:56:39:275	fe80::a4b3:e404:1eec:3746	1741	2001:e68:5414::0001:22cf	80	IGMP	255
72	06:56:39:335	fe80::a4b3:e404:1eec:3746	1741	2001:e68:5414::0001:22cf	80	UDP	255
73	06:56:39:395	2001:4860:4860::8888	1742	2001:a18:25:11::40	80	TCP	180
74	06:56:39:455	fe80::a4b3:e404:1eec:3746	1742	2001:e68:5414::0001:22cf	80	UDP	180

7 Conclusions

In conclusion, we can note the great difference in the apparent results between IPv4 and IPv6 of network packets. The results of each scenario were reviewed separately and discussed separately to find out the lessons derived from this scenario and then a discussion was held about the results of the scenarios with some. A difference in results was observed between the two scenarios. This discrepancy represents the difference in the values and protocols used. In Scenario A, only three protocols were captured TCP, UDP, and IGMP since the ICMP protocol was not captured. However, in Scenario B four protocols have been used and captured: TCP, UDP, ICMP, and IGMP. Some of the protocols used in Scenario B for IP version 6 that were not visible and did not occur in IP version 4 were found. For example, the ICMP protocol appeared as a difference between the two scenarios. In addition to the protocols taken, the ports also differed between the two scenarios or, more precisely, the most frequently used ports in the network differed. This difference is due to several factors that directly affect the results, including the captured IP packet if version 4 or 6 and the size of the captured sample. In addition, the use of the network at the time of implementation of the system greatly affects the results both in terms of protocols captured or through ports the most frequently used network.

References

Anu, P. and Vimala, S. (2017) 'A survey on sniffing attacks on computer networks', *2017 International Conference on Intelligent Computing and Control (I2C2)*, pp.1–5 [online] http://ieeexplore.ieee.org/document/8321914/ (accessed 21 May 2019).

Chauhan, D. and Sharma, S. (2014) 'A survey on next generation internet protocol IPV6', *International Journal of Electronics and Electrical Engineering*, Vol. 2, No. 2, pp.143–146.

Davis, J.J. and Clark, A.J. (2011) 'Data pre-processing for anomaly based network intrusion detection: a review', *Comput. Secur.*, Vol. 30, Nos. 6–7, pp.353–375.

Elsen, L. et al. (2015) 'goProbe: a scalable distributed network monitoring solution', *2015 IEEE International Conference on Peer-to-peer Computing (P2P)*, pp.1–10, Boston, MA, USA.

Gandhi, C., Suri, G., Golyan, R., Saxena, P. and Saxena, B. (2014) 'Packet sniffer – a comparative study', *International Journal of Computer Networks and Communications Security*, Vol. 2, No. 5 pp.179–187.

Jaisinghani, D., Naik, V., Kaul, S.K. and Roy, S. (2017) 'Sniffer-based inference of the causes of active scanning in WiFi networks', *2017 Twenty-third National Conference on Communications (NCC)*, Chennai, pp.1–6.

Mekky, H. et al. (2014) 'Application-aware data plane processing in SDN', in *ACM Workshop Hot Topics Network*, pp.13–18, Chicago, IL, USA.

Nishanth, N. and Babu, S.S. (2014) 'Sequence number alteration by logical transformation (SALT): a novel method for defending session hijacking attack in mobile ad hoc network', *International Journal of Computer and Communication Engineering*, Vol. 3, No. 5, pp.338–342.

Oluwabukola, O., Awodele, O., Ogbonna, C., Chigozirim, A. and Anyaehie, A. (2013) 'A packet sniffer (PSniffer) application for network security in Java', *Proceedings of the Informing Science and Information Technology Education Conference*, Informing Science Institute, pp.389–400.

Poonkuntran, S. and Arun, A.M. (2014) 'Study of Honeypots: analysis of WiFi Honeypots and Honeypots tools', *AENSI Journals on Advances in Natural and Applied Sciences*, Special Edition, Vol. 8, No. 17, pp.48–59.

Qadeer, M.A., Iqbal, A., Zahid, M. and Siddiqui, M.R. (2010) 'Network traffic analysis and intrusion detection using packet sniffer', *2010 Second International Conference on Communication Software and Networks*, Singapore, Singapore.

Qazi, Z.A. et al. (2013) 'Application-awareness in SDN', *ACM SIGCOMM Comput. Commun. Rev*., Vol. 43, No. 4, pp.487–488.

Seddiki, M.S. (2014) 'FlowQoS: QoS for the rest of us', in *Workshop Hot Topics Softw. Defined Network*. pp.207–208, Chicago, IL, USA.

Singh, J., Gupta, S. and Kaur, L. (2012) 'A cross-layer based intrusion detection technique for wireless networks', *International Arab Journal of Information Technology*, Vol. 9, No. 3, pp.201–207.

Singh, R. and Kumar, S. (2018) 'A comparative study of various wireless network monitoring tools', *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, Jalandhar, India, pp.379–384.

Tsai, P-W. et al. (2018) 'Network monitoring in software-defined networking: a review', *IEEE Systems Journal*, Vol. 12, No. 4, pp.3958–3969.

Xu, J., Liu, W. and Zeng, K. (2016) 'Monitoring multi-hop multi-channel wireless networks: online sniffer channel assignment', *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, Dubai, pp.579–582.

Yang, J., Zhang, Y., King, R. and Tolbert, T. (2018) 'Sniffing and chaffing network traffic in stepping-stone intrusion detection', *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, Krakow, Poland, pp.515–520.

Zhao, Z., Huangfu, W. and Sun, L. (2012) 'NSSN: a network monitoring and packet sniffing tool for wireless sensor networks', *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Limassol, Cyprus, pp.537–542.