TCP/IP Networking Protocol Analysis

Pascal Donfack

Bachelor of Engineering

Telecommunication Engineering

Department of Engineering

Macquarie University

November 6, 2017

Supervisor: Professor Robert ABBAS

ACKNOWLEDGMENTS

I would like to express my gratitude to a number of people who helped me throughout the course of my degree and this project.

First and foremost, I would like to thank my supervisor who was patient in answering my questions and encouraging in reviewing my work.

Special thanks also to my friend Dr Sara Cotterall, who helped and encouraged me in all possible ways.

Special thanks to my wife Landresse, who looked after the children and our home whenever I was busy reading or working on an assignment. Next, I would like to thank my children for putting up with the times when I was too busy or too tired to play with them. Finally, I would like to thank my late mother, who instilled in me a belief in the importance of education and the need to always do my best.

STATEMENT OF CANDIDATE

I, Pascal Donfack, declare that this report, submitted as part of the requirements for the award of the Bachelor of Engineering in the Department of Electronic Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment in any other academic institution.

Student's Name: Pascal Donfack

Student's Signature:

Date: 6 November, 2017

ABSTRACT

Nowadays, Transmission Control Protocol/Internet Protocol (TCP/IP) is known to be the most reliable network communication protocol capable of handling retransmission, packet loss, congestion control and more. In TCP/IP communication, packet loss and congestion are likely to have an impact on bandwidth. Performance is affected even at low packet loss rates so with an enlarged rate of packet loss, a radical drop in bandwidth efficiency occurs. To identify the source of this unpredictable network performance, a thorough examination of TCP/IP traffic was conducted.

This thesis studied the behavior of the main protocols involved in the 4 layers of the TCP/IP stack. This project, which was primarily concerned with layers 3 and 4 of the ICP/IP stack protocol, carried out a comparison between the performance of two different congestion control algorithms (Cubic and Reno). Several experimental tests were conducted to determine when the connection experienced data loss and ACK loss, with the results plotted on individual graphs showing packet behavior.

Initially two different TCP congestion control algorithms were used to observe their influence on bandwidth rate. Subsequently the TCP variables - advanced window scaling and window scaling - were changed to observe their role in obtaining acceptable bandwidth rate with respect to packet drop and the retransmission rate.

The study revealed a significant reduction in performance during packet loss. Surprisingly, the result showed that the congestion control algorithms, Cubic and Reno led to the same outcome when the link was experiencing packet loss. However, ACK loss did not significantly affect performance, and up to 40% loss in ACKs could be tolerated with almost no reduction in performance.

While TCP still functions adequately when experiencing single segment loss, the challenge is to handle multiple packet loss. Future research, therefore, might investigate new algorithms for multiple packet loss in the network.

Contents

List of Figures

List of Tables

1 Introduction

This thesis presents several tools and approaches for displaying collected internet traffic packets, and studying some of the most commonly used communication protocols. The diagnosis of Transmission Control Protocol/Internet Protocol (TCP/IP) performance problems using these tools presents a more accurate result than those achieved by earlier tools and allows a much more detailed understanding of packet trace analysis. The software adopted in this study easily determines the characteristics of the network traffic by documenting and analysing all internet transmissions. This tool can also indicate optimal performance of the system.

The use of internet-dependent devices in the home has been increasing given that broadband connections in homes are increasingly common. The internet is used in homes with devices such as laptops, PCs, smartphones and smart televisions. Therefore, the volume of network traffic increases due to the large number of devices connected to a single home network. This research project analyses the network traffic recorded during one hour, one day and one week in a home network. *Wireshark* software, which is an open source platform for capturing network traffic, was the primary tool used in this project.

The simple structure of the network comprises any computer device such as a laptop or desktop computer, a router or access point and the internet. An investigation of the different types of internet traffic analyzer software available resulted in the selection of two types of software - *Wireshark* and *Microsoft Network Monitoring* - for this project.

The project seeks to answer the following questions by recording, analyzing and documenting the internet traffic on the home computer network:

1. Which applications contribute most to the total volume of internet traffic in a TCP/IP network each hour, day and week?
2. What patterns of security threats to the integrity of the TCP/IP network are detected in hourly, daily and weekly scans of internet traffic?
3. What are the major network Key Performance Indicators (KPIs) in the TCP/IP network each hour, day and week?

1.1 Project Goals

The project aims to investigate the performance of the home network traffic to which the network analyzer has access. This was expected to result in an understanding of the contents of the internet packets.

Secondly, the project aims to monitor and identify thread and network problems and troubleshoot any issues arising. It was hoped that this would result in a sound understanding of the use of packet sniffers such as *Wireshark* and *Microsoft Network Monitor* for planning, monitoring, optimizing and detecting illegal activities on the network.

The third main goal of this project is to understand the use of packet sniffers. The study set out to investigate the characteristics of the software and their usefulness.

1.2 Project Planning

The project was planned by carefully considering its scope, time frame, and associated costs. The project timeline comprised three important stages. First the proposal was drafted, submitted and approved. This involved searching the literature for recent relevant studies and analyzing and summarizing their findings. The results of this stage were summarized in a literature review which was submitted for assessment. The second stage consisted of designing and implementing the study by capturing TCP/IP internet traffic at hourly, daily and weekly intervals. The final stage involved analyzing the data obtained and summarizing the results in a series of tables and graphs and documenting the results.

1.3 Project Scope

Based on the project's goal of recording and analyzing internet traffic on the home network at hourly, daily, and weekly intervals and making recommendations about the best performance of internet traffic, an appropriate capturing location needed to be identified. This may vary from one network to another. In this project, a home network was chosen. To achieve the project's goals, the following stages were realized:

1. Choose and connect network
2. Capture packets
3. View and capture traffic analysis
4. Filter specific traffic and document the findings

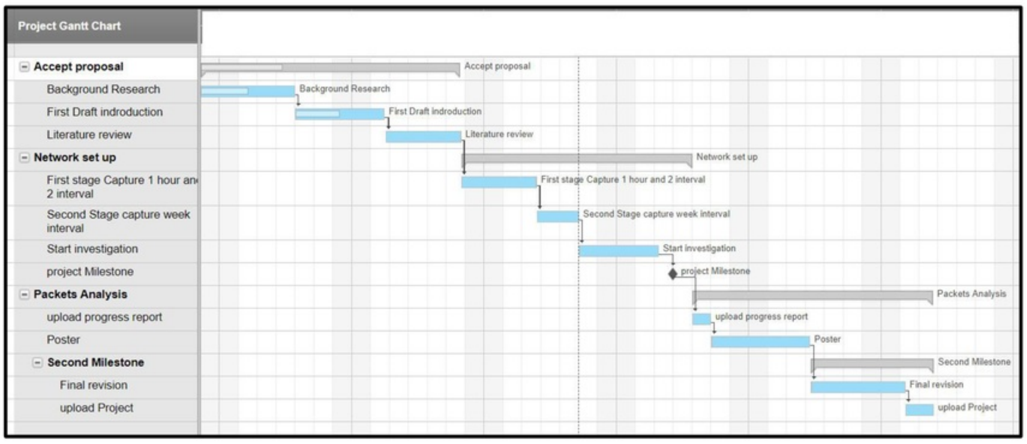


Figure 1 Project scope and planning

1.4 Time

To ensure that the project was successfully completed on time, Figure 1 (above) was drafted at the start of Week 2 of Semester Two (Tuesday August 7, 2017) and submitted to the project supervisor for approval. The capturing phase ran for a period of 3 days for three different durations according to the schedule shown in Figure 1. The data gathering period ran from August 7-26th with analysis starting in Week 6 of the semester on the 27th of August 2017.

1.5 Cost

The project was carried out at no financial cost. All software used was open source and downloaded free. The existing network equipment was already owned by the researcher. No further purchases were required.

2 Literature Review

Today, network monitoring is more important than ever due to the rapid growth of malware and internet threats. The recent increase in malicious content that can compromise specific networks [1] poses a major threat to network performance and efficiency. Experts [2] claim that maintaining network security represents a major challenge for network administrators.

According to Xu et al., (2013) [24], a traffic analysis platform is developed at first in which the incoming or outgoing traffic is collected, analysed and captured over a home network. Then the characteristics of the traffic are explained in detail on the basis of datasets, IP address and the time at which the traffic occurs. In this report, the author took two networks in the home known as Network A and Network B and compared them in terms of the incoming and outgoing network traffic over a time span of one month. Comparative results were produced using a traffic monitoring platform which collected and analysed the traffic over the home network.

According to Cecil (n.d.) [22], traffic monitoring is a significant means of troubleshooting the network to find security issues. The study reported here conducted the traffic analysis based on using a router and without using a router. SNMP, RMON and Cisco Net flow are the router based techniques; two recent methods of monitoring known as WREN and SCNM were used to find the traffic involved in the network. These router-based methods of analysis provide less flexibility whereas the non-router based monitoring methods include passive and active monitoring of the network traffic over a home network.

Liu, Liu and Ansari (2014) [23] argue that monitoring and analyzing network traffic can optimize the network's resources. Network traffic monitoring and analysis is also used for improving the experience of the network user. As the existing solutions are not ideal or scalable when analysis is required for large data traffic over a time span of more than one month, monitoring and analysis of traffic in their study was achieved using *Hadoop* [11]. *Hadoop* is an open source platform that can be used to obtain efficient feasible results when large data traffic monitoring is required.

The concerns of network administrators and traffic analysts are generally due either to their lack of appropriate tools to detect and resolve issues or their lack of relevant expertise in dealing with such threats [2]. Understanding the origin of the threat is the first step in taking appropriate action to protect and strengthen the network's overall performance.

Today different types of packet sniffer software are the most preferred troubleshooting tools employed when the network is under threat [4]. Sniffer software is used for listening to and capturing internet traffic. Such software can be extremely useful in detecting, tracing and analyzing network traffic.

The software that is now marketed under the name of *Wireshark* developed from an earlier, less sophisticated version called *Ethereal* [5]. However, Wireshark has been developed in various ways so that it is now a useful tool for analyzing wired and wireless network traffic. Analysts who operate within the laws and corporate policies that regulate the use of *Wireshark* can troubleshoot and secure their network efficiently and effectively [8].

Transport Control Protocol/Internet Protocol (TCP/IP) is a reliable transport protocol that is particularly suited for networks that consist of links with lower error rates [1]. The roots of the packet switching protocol can be traced back to 1969 when the Advanced Research Projects Agency (ARPA) funded a research project aimed at developing an efficient protocol for sending and transmitting packets in a network. It was first defined in 1980 by Request for Comment (RFC) 760 prepared by the Information Sciences Institute of the University of California for the US Department of Defense's ARPA.

TCP allowed specification of quality of service (QoS) using five parameters: viz stream versus datagram, precedence, reliability, speed, and speed versus reliability. The specific terminology for precedence was similar to the controls used in military command and control (CoC) networks. The transport protocol allowed multiple messages to be tagged using different precedence that impacted the order of delivery [3].

At this point, it is important to note that TCP is mitigated at the end nodes. It strives to minimize unreliable data found in multiple layers. The two end nodes perform different functions communicated in combination including numbering the packets before sending, monitoring the packets that arrive, reordering the packets if required, retransmitting packets that are lost, and then delivering the data to the end receiver [3].

The network protocol performs different treatments to packets as they are transmitted through the net. The protocol provides preferential treatment to network packets demanded by high priority applications [5]. This treatment to the packets is known as enhanced services.

The technological advancements in switches, fiber, routers, and other components have resulted in the provision of increased IP traffic speed [5]. Having said that, there is a possibility of potential for traffic congestion that can result in a delay in the transmission of packets.

Various researchers have proposed different practical schemes for controlling congestion [5]. A range of schemes have been presented to control IP traffic congestion in the overloaded network core. IP traffic congestion also happens because of variation in the speed of data from a fast LAN (local area network) through a slower WAN (wide area network) [6], and when different input streams reach a router that does not have the capacity to handle the inputs.

The Error Correction Mechanism (ECM) of the TCP/IP is one control mechanism that comes into play during network congestion [9]. When an out-of-order packet is received due to a network delay, the TCP may generate an acknowledgment due to the fast-retransmit algorithm [5]. This acknowledgment lets the other end of the network know that an out-of-order packet has been received. This is an example of how the ECM of the TCP/IP functions. The advantage of ECN is that it prevents an unnecessary drop in packets [7]. Another benefit of ECN is that it prevents delay in the transmission of packets, particularly from low-bandwidth TCP connections [11].

A number of proposals have been put forward to reduce packet loss rate due to network congestion, such as active queue management and its variations [9]. Whenever a connection is started, the network protocol attempts to increase the transmission rate by increasing the congestion window. However, to avoid excessive packet losses due to this fast-start stage, the sending node typically implements a congestion avoidance algorithm [7].

A congestion avoidance algorithm consists of monitoring the threshold value known as ssthresh (slow start threshold) that is approximate to the window size that can be supported by the network. In case the window size exceeds the threshold, TCP executes the congestion avoidance algorithm. During this period, if the network capacity cannot handle the transmission rate, it will result in a loss of packets. This loss of packets is perceived by TCP through the receipt of different duplicate acknowledgments that are issued by the receiving node [8].

In recent years, a number of efforts have been focused on designing and implementing more robust packet switching algorithms [10]. Multipath TCP (MTCP) is one such advance that has been proposed as an improvement to TCP [11]. According to the authors, MTCP can seamlessly and efficiently use the available bandwidth providing improved throughput. It also allows better fairness on several different topologies [11]. The benefit of using this algorithm is that it allows the user to regroup data centre networks as per the relationship between routing, transport protocols, and topology. The network protocol aims for improved topologies that are not possible for a single path TCP. As a concept of proof, the researchers introduced a dual homed version of the FatFree topology that is optimized on MPTCP. This results in better performance over several different workloads as compared to the existing TCP protocol.

Moreover, a number of researchers have developed a solution for networks that use programmable switches or a centralized coordinator to place flows on paths based on Fat Tree topology, as mentioned earlier [12].

Some researchers have introduced an improvement to the TCP network algorithm that sources network congestion and then utilizes source routing [13]. However, the problem with this approach is that the congestion changes quickly and the initial choice may not be the best route. A more practical solution involves spreading the connection over different paths that make the scheduling issue more traceable [14]. Routing packets over multiple paths is another solution [14]. However, there is a limitation to this solution in that it is not known how to apply back-pressure over multiple hops paths using various hardware that is the norm these days.

After the TCP connection has been made, it is not possible to modify any of the elements. MCTCP handles this problem by extending TCP to allow the end user to easily transfer data that belongs to one connection over multiple paths. In order to achieve this, MTCP combines different TCP connections known as sub-flows in the RFC 6824. This is done in a single TCP connection. The first sub-flow initiates within a three-way handshake [15].

At present, different independent interoperation implementations of MTCP exist. The most widely used include Linux and iOS platform [16]. The transport protocol is also supported by Solaris and FreeBSD platforms.

Finally, the input from Wi-Fi Assist allows the hand-over traffic to be transferred in the cellular interface. The deployment of advanced TCP such as MPTCP has contributed to a significant decrease in network errors. In some cases, for instance, in Apple Siri, the network error rate has decreased by as much as 80 percent [17].

The protocol design in the TCP/IP model does not depend on single hierarchical layering or encapsulation [19]. In fact, RFC 3439 consists of a section that considers layering as damaging to TCP/IP performance [20].

The networking model makes use of layers to help humans understand and categorize many different network functions. The network protocol has been popular with both users and website designers because of its inherent openness. Another reason for the popularity of the networking platform is that it can be renewed perpetually without any problem [21]. At the same time however, the openness of the system can also make it vulnerable to network attack. Hackers can make use of the loopholes to carry out network attacks. Preventing this kind of attack requires strict security algorithms to be implemented in order to minimize some of the vulnerabilities of the system.

While there is no overall regulating body for TCP/IP, control and improvements are made through a process of cooperation [10]. However, some organizations that have been established to provide standards for the network protocols, similar to ISO. One example is the Internet Society (ISO) that is managed and organized by the Internet Architecture Board (IAB).

This discussion makes it clear that the TCP/IP platform is still evolving. The evolution is being carried out to make the network protocol more rapid, robust and secure. Furthermore, several RFCs have been proposed that will greatly improve the reliability and speed of the network protocol in the near future.

3 Experimental Procedures

1.1 Introduction

Network operators continually monitor key performance indicators (KPIs) within their networks to identify performance problems and ensure customer satisfaction. These displays include quality of service, lost traffic, and other elements. To keep up to date with growing traffic, engineers are continually required to optimize network performance. Even the best network design cannot deliver performance if the physical infrastructure performs below expectations. This can lead to both operational and business challenges as customers notice poor network performance.

The first requirement is a thorough understanding of the network's specific requirements and the components available to help meet those requirements.

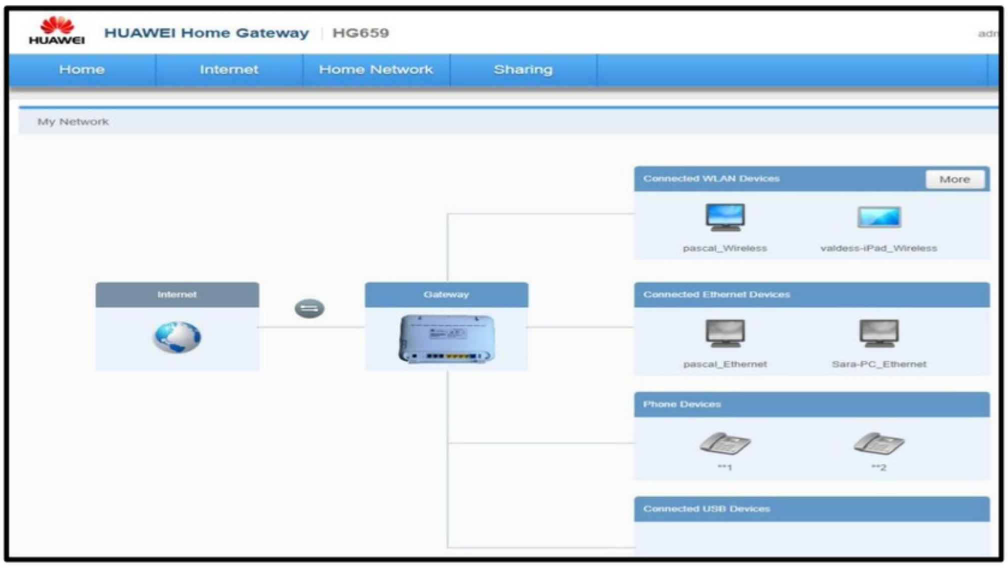


Figure 3.1 Network components

1.2 Network components

The basic building blocks of the experimental network are shown in figure 3.1. They include:

Access point (HUAWEI Home Gateway)

Access point comprises two directional antennas. In technical terms, an antenna is just a translator tool between guided and unguided media. Electromagnetic energy is transformed between free space and waveguide with the antenna help. Radiation and reception of this energy is the main function on the antenna. An antenna's performance measured using several techniques, with varying degrees of relevance to any application. These are the most important parameters:

- Frequency of operation of between 2.4 GHZ and 5 GHz refers to the operating frequency band—all antenna specifications are guaranteed within the frequency of operations. These frequency bands and channel arrangements are defined by ITU-R recommendations or ECC (error control check).
- Polarization which is the orientation of electric field driving the signal either vertical or horizontal.

Modulation and demodulation is one of the main role of the access point, the throughput can be increased either by adding more data channels used or by increasing the modulation scheme employed. Modulation schemes can range from low-order QPSK (quadrature phase shift keying) to higher-order 256 QAM (quadrature amplitude modulation) which is method of merging two amplitude-modulated (AM) signals into a single channel, thus doubling the effective bandwidth.

The Access point Ethernet RJ45 ports allows the connectivity of computers and phone to the network importantly. It's as a gateway to the internet.

One **Hewlett Packard** (device C). With the following characteristic

AMD Athlon Dual Core CPU 2.8 GHz (gigahertz) of frequency, 8 GB (Gigabits) of Memories (RAM), 32 bite operating System Windows 8, 500 GB hard disk of storage. TP-

LINK wireless network interface card and double high definition ASI screens. Genius keyboard and a mouse.

One **ASUS Laptop computer** (device B) with the following specifications:

Intel core i7 CPU 2.4 GHz, 2.4 of frequency, 16 GB (Gigabits) of Memories (RAM), 64bite operating System Windows 10, 1TB hard disk of storage. Intel Centrino wireless –N 2230 network interface card and high definition ASUS 18-inch screen.

Two tablets (1 Apple iPad 5 (device A), and 1 Samsung Galaxy Tab A with OS and Android operating system

Transmission media

The transmission media are the physical media connecting the computer and access point consisting of a substance (gas, solid or liquid) that can propagate energy waves. There are three main media types.

- Wireless media including radio frequencies, satellite, infrared and microwave.
- Copper cable such as shielded twisted-pair, unshielded twisted pair and coaxial
- Fiber optic cable (also known as optical fiber cable) consisting of glass thread used for data transmission. It consists of a bundle of glass threads, each with the capability of transmitting data modulated onto light waves.

Applications

Web browsers (Microsoft Internet Explorer, Mozilla Firefox, and Google Chrome) - software applications using HTTP protocol for repossessing, presenting and navigating information resources on the World Wide Web

- *Wireshark* (open Source Software) - a network protocol analyzer used to decompose the network structure at microscopic level enabling detailed examination of what is happening in the network.

- *Microsoft Network Monitor* is another network protocol analyzer used in this project.

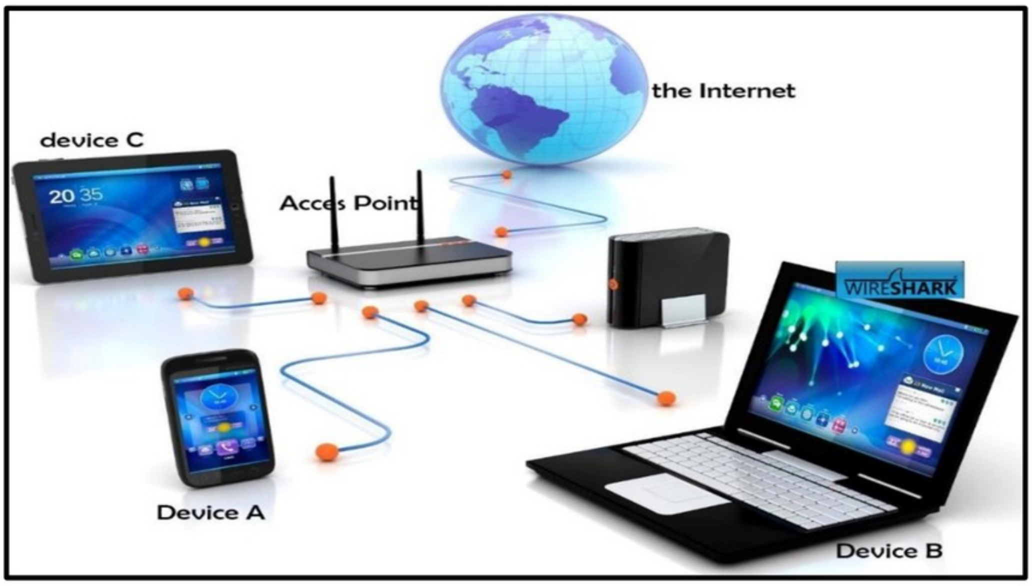


Figure 3.2 Basic network configuration

1.3 Network installation and configuration

No matter how well the network been planned or how much care has been taken in the selection of equipment, poor installation practices will expose the network reliability and deliver performance far below planned expectations. The location, the distance from the access point, the direction of the antenna, the presence of nearby equipment, and other prevailing local conditions mean each installation presents challenges requiring careful attention. While manufacturers provide detailed instructions on how to best assemble and install equipment, it's the expertise, skill and care of the network engineer to ultimately determine if the network fulfills its performance and reliability goals.

For the purposes of the project, figure 3.2 is considered to be the network diagram for the entire experiment. As shown in figure 3.2, the network analyzer (called *Wireshark*) is installed on Device B which, in turn, is connected to the access point Ethernet port. Devices A and C are also wirelessly connected to the same router to gain access to the internet. The goal of Device B is to use the software to capture the internet traffic and save it on the hard drive for further investigation.

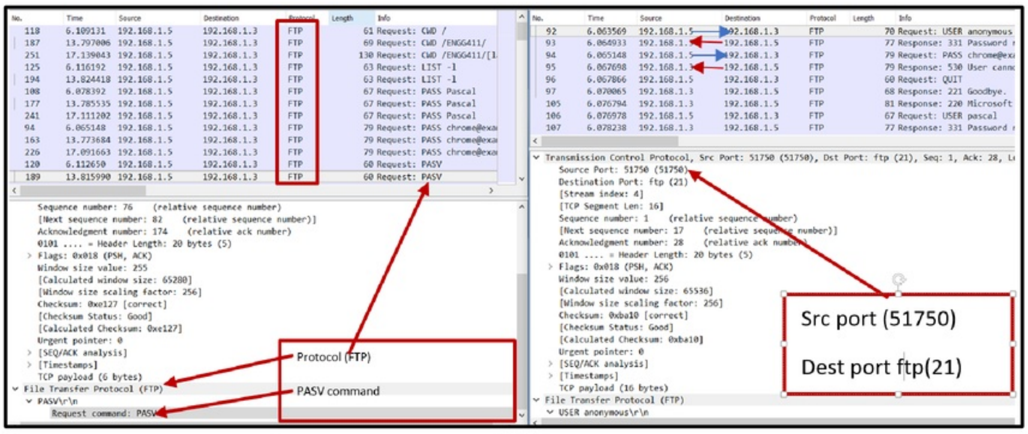
The basic steps taken in the experiment to capture internet packets using *Wireshark* are detailed below:

1. Create a folder in the hard drive to store the captured file and run the network protocol analyzer application.
2. Select the **capture > interfaces** menu option to list all available interfaces.
3. Choose the **active network card interface** (NIC) from the list shown in the pop up window.
4. Click the option to **set capture parameters**. In the option window, tick on the use of Promiscuous mode for all interfaces. If the promiscuous mode is not selected, frames with MACs other than the one the interface has are ignored.
5. Click on **capture filter** to allow only the target protocol (e.g. TCP only, or UDP port (80) to be captured.
6. Name the capture file (one_Hour_capture_dayTime) and link it to the created folder.
7. For multi file capturing, tick on **use multiple files** and set capture as desired. Indicate the number of files to be captured, desired size, time duration for each capture, the ring buffer and the time to terminate the capture.
8. On the Display option, choose **yes or no** to update the list of packets in real time
9. Decide on the name resolution during the capturing process (resolve MAC address, use external network name resolve).
10. Start the capture. (Files should automatically be saved on the hard drive and the application should stop capturing after the elapsed time.)
11. Open the browser and start surfing the internet.

1.4 Capturing mode selection

By choosing the promiscuous mode, a network card was enabled so that its driver could capture traffic that is addressed to another device in the network. Configuring the 802.11 network adapter to operate in this mode will ensure that it captures only SSID (service set identifier). SSID is the primary name associated with an 802.11 (WLAN) including home networks. To capture all the traffic, the adapter can receive, this adapter should be configured in monitor mode.

1.4.1 Capture on a specific application



```
No.   Time        Source        Destination   Protocol  Length  Info
118   6.109131    192.168.1.5   192.168.1.3   FTP       61 Request: CWD /
187   13.797006   192.168.1.5   192.168.1.3   FTP       69 Request: CWD /ENGG411/
251   17.139043   192.168.1.5   192.168.1.3   FTP      130 Request: CWD /ENGG411/[1
125   6.116192    192.168.1.5   192.168.1.3   FTP       63 Request: LIST -1
194   13.824418   192.168.1.5   192.168.1.3   FTP       63 Request: LIST -1
108   6.078392    192.168.1.5   192.168.1.3   FTP       67 Request: PASS Pascal
177   13.785535   192.168.1.5   192.168.1.3   FTP       67 Request: PASS Pascal
241   17.111202   192.168.1.5   192.168.1.3   FTP       67 Request: PASS Pascal
94    6.065148    192.168.1.5   192.168.1.3   FTP       79 Request: PASS chrome@exa
163   13.773684   192.168.1.5   192.168.1.3   FTP       79 Request: PASS chrome@exa
226   17.091663   192.168.1.5   192.168.1.3   FTP       79 Request: PASS chrome@exa
120   6.112650    192.168.1.5   192.168.1.3   FTP       60 Request: PASV
189   13.815990   192.168.1.5   192.168.1.3   FTP       60 Request: PASV

    Sequence number: 76    (relative sequence number)
    [Next sequence number: 82    (relative sequence number)]
    Acknowledgment number: 174    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
    Window size value: 255
    [Calculated window size: 65280]
    [Window size scaling factor: 256]
    Checksum: 0xe127 [correct]
    [Checksum Status: Good]
    [Calculated Checksum: 0xe127]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (6 bytes)
v File Transfer Protocol (FTP)
  v PASV\r\n
      Request command: PASV
```

```
No.   Time        Source        Destination   Protocol  Length  Info
92    6.063569    192.168.1.5   192.168.1.3   FTP       70 Request: USER anonymous
93    6.064933    192.168.1.3   192.168.1.5   FTP       77 Response: 331 Password
94    6.065148    192.168.1.5   192.168.1.3   FTP       79 Request: PASS chrome@exa
95    6.067698    192.168.1.3   192.168.1.5   FTP       79 Response: 530 User cann
96    6.067866    192.168.1.5   192.168.1.3   FTP       60 Request: QUIT
97    6.070065    192.168.1.3   192.168.1.5   FTP       68 Response: 221 Goodbye.
105   6.076794    192.168.1.3   192.168.1.5   FTP       81 Response: 220 Microsoft
106   6.076978    192.168.1.5   192.168.1.3   FTP       67 Request: USER pascal
107   6.078238    192.168.1.3   192.168.1.5   FTP       77 Response: 331 Password

v Transmission Control Protocol, Src Port: 51750 (51750), Dst Port: ftp (21), Seq: 1, Ack: 28, L
    Source Port: 51750 (51750)
    Destination Port: ftp (21)
    [Stream index: 4]
    [TCP Segment Len: 16]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 17    (relative sequence number)]
    Acknowledgment number: 28    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
    Window size value: 256
    [Calculated window size: 65536]
    [Window size scaling factor: 256]
    Checksum: 0xba10 [correct]
    [Checksum Status: Good]
    [Calculated Checksum: 0xba10]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (16 bytes)
v File Transfer Protocol (FTP)
  v USER anonymous\r\n
```

Figure 3.3 Windows presenting parts of an FTP packet.

To listen to a specific traffic conversation, a filter is performed on the port number of the application. In general, when we know the application port number, the capture can be set to look in the application traffic over UDP or TCP protocol. To create a packet capture on the specific application, port filtering is deployed. For example, on FTP (file transfer protocol), queries and responses run over port 21. The FTP zone transfer runs over the TCP on port 21. The result of setting a filter of FTP traffic over the TCP on port 21 is illustrated in figure 3.3. figure 3.3 shows a typical sample FTP packet produced by the *Wireshark* platform.

In the packet in Figure 3.3, Frame number 106 has the following characteristics:

- Capture Date: 26 August
- Capture Time: 18: 59 minutes
- Source Address: 192.168.1.5
- Destination server: IP address 192.168.1.3
- Establishing ftp connection
- Frame number 105 has the following characteristics:
- Capture Date: 26 August
- Capture Time: 18: 59 minutes
- Source Address: 192.168.1.3
- Destination server: IP address D 192.168.1.5

The same network shows an FTP (file transfer protocol) request for login information to access Microsoft files server services. Frames number 106 and 108 clearly show in the information column the user name (pascal) and Frame 108 clearly displays the password (***) to access the server. This demonstrates a security issue with applications using the FTP protocol.

The study of FTP in this session demonstrated that it is a TCP-based transfer application protocol. Figure 3.3 shows two separate communications used by FTP. The first connection is for commands and the second use for data transfer. Port number 21 is the most reliable for the FTP command channel, but other port number can still be configured to run FTP protocol. Through figure 3.3 the observation is made on the port 20 used for data channel, where Both FTP transfer Process type Such as active and passive mode are illustrated in the observed packet trace. Active mode uses PORT command for data transfer from FTP server to client, whereas Passive mode uses the PASV command to transfer data from FTP client to the FTP server. Wireshark Follow TCP streams option enable data transfer collection.

1.5 TCP/IP traffic capture management

This step involved launching the web browser (Google Chrome) and spending the next two hours visiting several websites including google.com, mq.edu.au and Yahoo.com. At the end of this two-hour period, the software automatically stopped capturing and storing each file on the computer's hard drive. In this scenario, the capture was done on Device B, with the wireless adapter set in promiscuous mode.

A similar scenario was completed for online screening of a *YouTube* video and TV program, also on Device B. Similarly, scenario was used when transferring files between the two computers in the network, listening to online radio (ABC radio and SBS radio) and sending email using client applications to the mail server application. Using the *Wireshark* filter option achieved good storage management of the files captured.

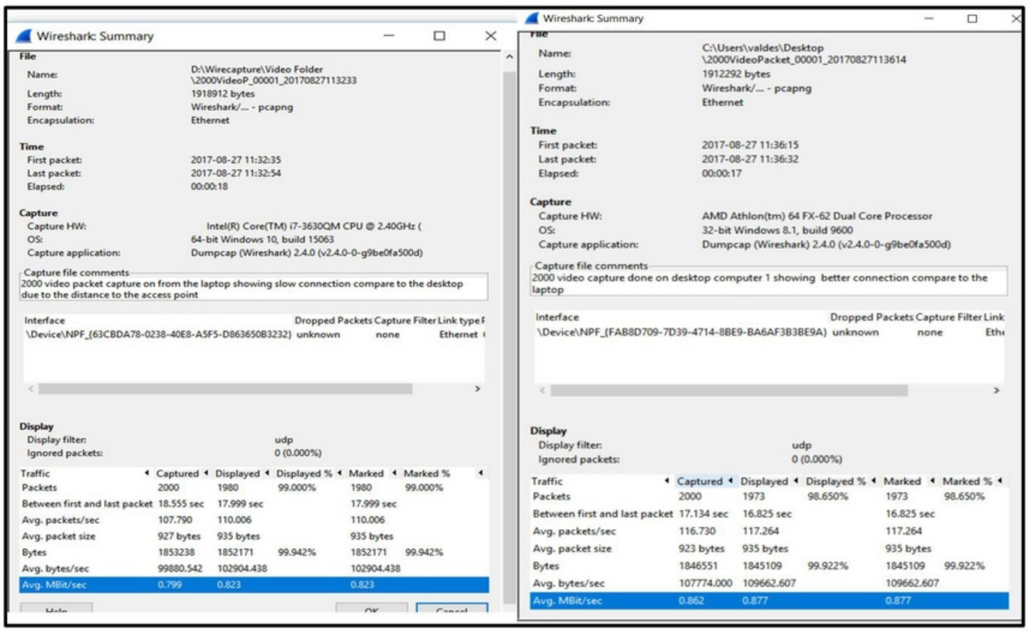
1.6 Analysis overview

1.6.1 TPC Packet overview description

Before the results of the software analysis are described, the components of each sniffer packet will be presented. Each default packet includes the following:

- 48 bits Ethernet Address example (60:36:dd:17:1a:0b)
- the Ethernet type 0X0806(ARP)
- No broadcast and No Multicast
- No ARP, IP, IP Address 192.198.1.5 (for the analyzer Laptop), TCP or UDP port example port 80 for (HTTP), HTTP TCP port

Figure 3.4 below summarizes the data provided on each packet filter including file length, time elapsed, the file format information, number of packets, average number of bytes and megabits per second. This summary is valuable when comparing appropriate network performance with problematic network performance.


Wireshark: Summary

```
File
  Name:                   D:\Wirecapture\Video Folder
                          \2000VideoP_00001_20170827113233
  Length:                 1918912 bytes
  Format:                 Wireshark/... - pcapng
  Encapsulation:          Ethernet

Time
  First packet:           2017-08-27 11:32:35
  Last packet:            2017-08-27 11:32:54
  Elapsed:                00:00:18

Capture
  Capture HW:               Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz (
  OS:                     64-bit Windows 10, build 15063
  Capture application:    Dumpcap (Wireshark) 2.4.0 (v2.4.0-0-g9be0fa500d)

Capture file comments
2000 video packet capture on from the laptop showing slow connection compare to the desktop
due to the distance to the access point

Interface                                           Dropped Packets Capture Filter Link type
\Device\NPF_{63CBDA78-0238-40E8-A5F5-D863650B3232}  unknown         none           Ethernet

Display
  Display filter:                         udp
  Ignored packets:                        0 (0.000%)

Traffic                      Captured    Displayed   Displayed %  Marked      Marked %
Packets                      2000        1980        99.000%      1980        99.000%
Between first and last packet 18.555 sec 17.999 sec               17.999 sec
Avg. packets/sec             107.790     110.006                  110.006
Avg. packet size             927 bytes   935 bytes                935 bytes
Bytes                        1853238     1852171     99.942%      1852171     99.942%
Avg. bytes/sec               99880.542   102904.438               102904.438
Avg. MBit/sec                0.799       0.823                    0.823
```

Wireshark: Summary

```
File
  Name:                   C:\Users\valdes\Desktop
                          \2000VideoPacket_00001_20170827113614
  Length:                 1912292 bytes
  Format:                 Wireshark/... - pcapng
  Encapsulation:          Ethernet

Time
  First packet:           2017-08-27 11:36:15
  Last packet:            2017-08-27 11:36:32
  Elapsed:                00:00:17

Capture
  Capture HW:             AMD Athlon(tm) 64 FX-62 Dual Core Processor
  OS:                     32-bit Windows 8.1, build 9600
  Capture application:    Dumpcap (Wireshark) 2.4.0 (v2.4.0-0-g9be0fa500d)

Capture file comments
2000 video capture done on desktop computer 1 showing  better connection compare to the
laptop

Interface                                           Dropped Packets Capture Filter Link
\Device\NPF_{FAB8D709-7D39-4714-8BE9-BA6AF3B3BE9A}  unknown         none           Eth

Display
  Display filter:                         udp
  Ignored packets:                        0 (0.000%)

Traffic                      Captured    Displayed   Displayed %  Marked      Marked %
Packets                      2000        1973        98.650%      1973        98.650%
Between first and last packet 17.134 sec 16.825 sec               16.825 sec
Avg. packets/sec             116.730     117.264                  117.264
Avg. packet size             923 bytes   935 bytes                935 bytes
Bytes                        1846551     1845109     99.922%      1845109     99.922%
Avg. bytes/sec               107774.000  109662.607               109662.607
Avg. MBit/sec                0.862       0.877                    0.877
```

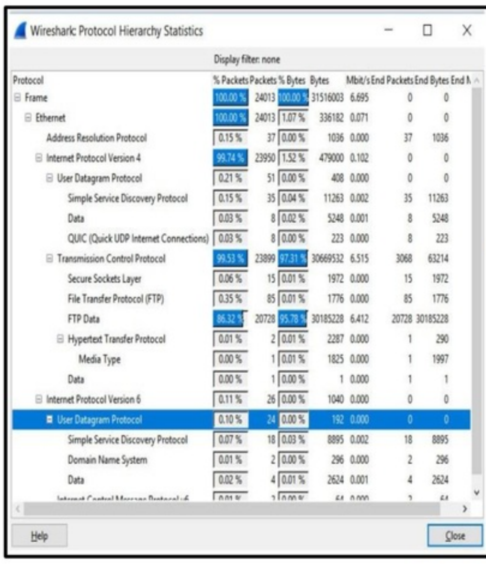
Figure 3.4 Summary window used to compare two set of data (UDP)

Figure 3.4 also compares two separate trace files. The left-hand window shows the slower download process when comparing the average packet per second and the average megabit rate. The right-hand window shows the faster download process with the higher data rate based on the average number of packets per second. This summary can be enhanced by filtering the number of values on TCP time _delta. Better visibility of the two instances can be obtained from the filtering result.

This delta time helps to measure the time between packets and specific reference points in the trace; it also helps identify gaps between consecutive packets. This difference in speed observed in figure 3.4 is due to the difference in distance from the access point. The further the device is from the router, the slower the connection.

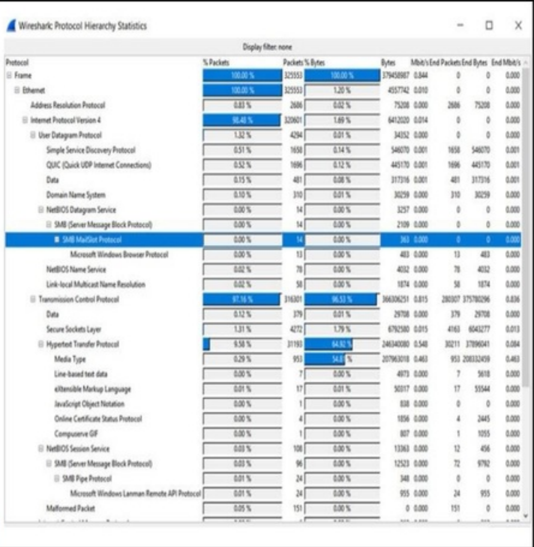
1.6.2 File interpretations

This section presents an interpretation of the data saved in the captured files based on the following statistical criteria: protocol hierarchies, end point and conversation, address and port, packet lengths, multicast stream and flow diagram.


Wireshark: Protocol Hierarchy Statistics

Display filter: none

Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End M
Frame	100.00 %	24013	100.00 %	31516003	6.695	0	0	
Ethernet	100.00 %	24013	1.07 %	336182	0.071	0	0	
Address Resolution Protocol	0.15 %	37	0.00 %	1036	0.000	37	1036	
Internet Protocol Version 4	99.74 %	23950	1.52 %	479000	0.102	0	0	
User Datagram Protocol	0.21 %	51	0.00 %	408	0.000	0	0	
Simple Service Discovery Protocol	0.15 %	35	0.04 %	11263	0.002	35	11263	
Data	0.03 %	8	0.02 %	5248	0.001	8	5248	
QUIC (Quick UDP Internet Connections)	0.03 %	8	0.00 %	223	0.000	8	223	
Transmission Control Protocol	99.53 %	23899	97.31 %	30669532	6.515	3068	63214	
Secure Sockets Layer	0.06 %	15	0.01 %	1972	0.000	15	1972	
File Transfer Protocol (FTP)	0.35 %	85	0.01 %	1776	0.000	85	1776	
FTP Data	86.32 %	20728	95.78 %	30185228	6.412	20728	30185228	
Hypertext Transfer Protocol	0.01 %	2	0.01 %	2287	0.000	1	290	
Media Type	0.00 %	1	0.01 %	1825	0.000	1	1997	
Data	0.00 %	1	0.00 %	1	0.000	1	1	
Internet Protocol Version 6	0.11 %	26	0.00 %	1040	0.000	0	0	
User Datagram Protocol	0.10 %	24	0.00 %	192	0.000	0	0	
Simple Service Discovery Protocol	0.07 %	18	0.03 %	8895	0.002	18	8895	
Domain Name System	0.01 %	2	0.00 %	296	0.000	2	296	
Data	0.02 %	4	0.01 %	2624	0.001	4	2624	


Wireshark: Protocol Hierarchy Statistics

Display filter: none

Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
Frame	100.00 %	325553	100.00 %	379458987	0.844	0	0	0.000
Ethernet	100.00 %	325553	1.20 %	4557742	0.010	0	0	0.000
Address Resolution Protocol	0.83 %	2686	0.02 %	75208	0.000	2686	75208	0.000
Internet Protocol Version 4	98.48 %	320601	1.69 %	6412020	0.014	0	0	0.000
User Datagram Protocol	1.32 %	4294	0.01 %	34352	0.000	0	0	0.000
Simple Service Discovery Protocol	0.51 %	1658	0.14 %	546070	0.001	1658	546070	0.001
QUIC (Quick UDP Internet Connections)	0.52 %	1696	0.12 %	445170	0.001	1696	445170	0.001
Data	0.15 %	481	0.08 %	317316	0.001	481	317316	0.001
Domain Name System	0.10 %	310	0.01 %	30259	0.000	310	30259	0.000
NetBIOS Datagram Service	0.00 %	14	0.00 %	3257	0.000	0	0	0.000
SMB (Server Message Block Protocol)	0.00 %	14	0.00 %	2109	0.000	0	0	0.000
SMB MailSlot Protocol	0.00 %	14	0.00 %	363	0.000	0	0	0.000
Microsoft Windows Browser Protocol	0.00 %	13	0.00 %	483	0.000	13	483	0.000
NetBIOS Name Service	0.02 %	78	0.00 %	4032	0.000	78	4032	0.000
Link-local Multicast Name Resolution	0.02 %	58	0.00 %	1874	0.000	58	1874	0.000
Transmission Control Protocol	97.16 %	316301	96.53 %	366306251	0.815	280307	375780296	0.836
Data	0.12 %	379	0.01 %	29708	0.000	379	29708	0.000
Secure Sockets Layer	1.31 %	4272	1.79 %	6792580	0.015	4163	6043277	0.013
Hypertext Transfer Protocol	9.58 %	31193	64.92 %	246340080	0.548	30211	37896041	0.084
Media Type	0.29 %	953	54.8[illegible] %	207963018	0.463	953	208332459	0.463
Line-based text data	0.00 %	7	0.00 %	4973	0.000	7	5618	0.000
eXtensible Markup Language	0.01 %	17	0.01 %	50017	0.000	17	55544	0.000
JavaScript Object Notation	0.00 %	1	0.00 %	838	0.000	0	0	0.000
Online Certificate Status Protocol	0.00 %	4	0.00 %	1856	0.000	4	2445	0.000
Compuserve GIF	0.00 %	1	0.00 %	807	0.000	1	1055	0.000
NetBIOS Session Service	0.03 %	108	0.00 %	13363	0.000	12	456	0.000
SMB (Server Message Block Protocol)	0.03 %	96	0.00 %	12523	0.000	72	9792	0.000
SMB Pipe Protocol	0.01 %	24	0.00 %	348	0.000	0	0	0.000
Microsoft Windows Lanman Remote API Protocol	0.01 %	24	0.00 %	955	0.000	24	955	0.000
Malformed Packet	0.05 %	151	0.00 %	0	0.000	151	0	0.000

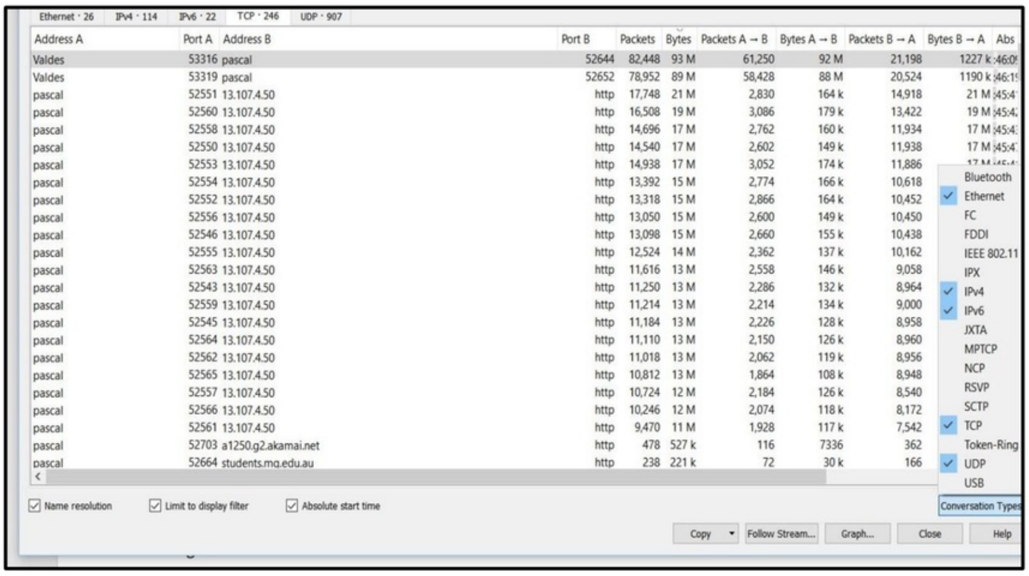
Figure 3.5 Protocol hierarchy information on the data transfer and web browser

Figure 3.5 displays from left to right (1) the protocol or application that was most used in the communication process, the percentage of number of packets (expressed in megabits and bits) and the bites and packet counts. According to the protocol, the figure 4.3 shows, 23899 packets (97.31% of the overall traffic) are TCP-based and only 51 packets used for DNS (Domain Name Service) requests and responses.

Figure 3.5 also shows that 85 packets were allocated to FTP (File Transfer Protocol), representing 0.35% of the traffic bytes which matched the number of packets after applying the FTP filter on the trace file. It therefore appears that most of the TCP traffic is file transfer traffic, with 953 packets being used for media type. The Host IP (Internet Protocol) address display filter is used to characterize all the applications and protocols used during the communication. The right-hand side of figure 3.5 represents the protocol hierarchy of the web browser traffic.

1.6.3 Active conversation identification in the trace file

In the context of computer networks, a conversation occurs when two devices exchange data. The term "end" is used to describe a single side of the conversation. Listening to the conversation between two hosts makes it possible to determine the most active conversation channel by examining the different packet rates and the duration of the conversations.


Ethernet · 26 | IPv4 · 114 | IPv6 · 22 | TCP · 246 | UDP · 907

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Abs
Valdes	53316	pascal	52644	82,448	93 M	61,250	92 M	21,198	1227 k	:46:0
Valdes	53319	pascal	52652	78,952	89 M	58,428	88 M	20,524	1190 k	:46:1
pascal	52551	13.107.4.50	http	17,748	21 M	2,830	164 k	14,918	21 M	:45:4
pascal	52560	13.107.4.50	http	16,508	19 M	3,086	179 k	13,422	19 M	:45:4
pascal	52558	13.107.4.50	http	14,696	17 M	2,762	160 k	11,934	17 M	:45:4
pascal	52550	13.107.4.50	http	14,540	17 M	2,602	149 k	11,938	17 M	:45:4
pascal	52553	13.107.4.50	http	14,938	17 M	3,052	174 k	11,886	[illegible]	[illegible]
pascal	52554	13.107.4.50	http	13,392	15 M	2,774	166 k	10,618		
pascal	52552	13.107.4.50	http	13,318	15 M	2,866	164 k	10,452		
pascal	52556	13.107.4.50	http	13,050	15 M	2,600	149 k	10,450		
pascal	52546	13.107.4.50	http	13,098	15 M	2,660	155 k	10,438		
pascal	52555	13.107.4.50	http	12,524	14 M	2,362	137 k	10,162		
pascal	52563	13.107.4.50	http	11,616	13 M	2,558	146 k	9,058		
pascal	52543	13.107.4.50	http	11,250	13 M	2,286	132 k	8,964		
pascal	52559	13.107.4.50	http	11,214	13 M	2,214	134 k	9,000		
pascal	52545	13.107.4.50	http	11,184	13 M	2,226	128 k	8,958		
pascal	52564	13.107.4.50	http	11,110	13 M	2,150	126 k	8,960		
pascal	52562	13.107.4.50	http	11,018	13 M	2,062	119 k	8,956		
pascal	52565	13.107.4.50	http	10,812	13 M	1,864	108 k	8,948		
pascal	52557	13.107.4.50	http	10,724	12 M	2,184	126 k	8,540		
pascal	52566	13.107.4.50	http	10,246	12 M	2,074	118 k	8,172		
pascal	52561	13.107.4.50	http	9,470	11 M	1,928	117 k	7,542		
pascal	52703	a1250.g2.akamai.net	http	478	527 k	116	7336	362		
pascal	52664	students.mq.edu.au	http	238	221 k	72	30 k	166		

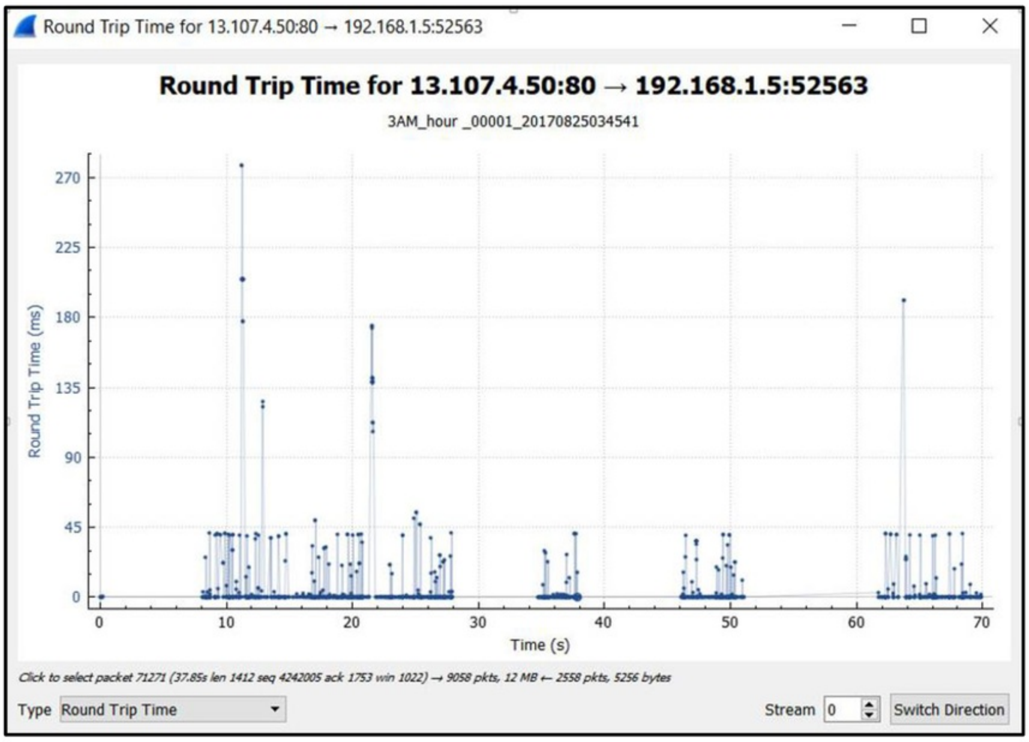
☑ Name resolution ☑ Limit to display filter ☑ Absolute start time

Bluetooth
✓ Ethernet
FC
FDDI
IEEE 802.11
IPX
✓ IPv4
✓ IPv6
JXTA
MPTCP
NCP
RSVP
SCTP
✓ TCP
Token-Ring
✓ UDP
USB

Figure 3.6 TCP conversation showing pair of hosts communicating with each other

Figure 3.6 shows that 26 Ethernet conversations took place, 114 IPV2 conversations, 22 IPV6 and 907 UDP conversations. Based on the number of bytes transferred between the two TCP hosts, the high rate (46 Megabytes) illustrated in the column Bytes A to B, represents the most active conversation in this file trace communication. This indicates that 114IPv2 remains the most utilized protocol in the communication network. 907 UDP explains what type of traffic was most used during the conversation.

Figure 3.7 (below) shows the time it took a packet to be transferred, plus the time it took the sender of the packet to receive the ACK of the last packet sent. The time delay that appears in figure 3.7 consists of the propagation time between the signal being transmitted and received. Being able to observe the round-trip time helps identify the cause of any inefficiencies in the network performance. The round-trip graph tracks the time between data being transmitted and the associated TCP ACK (acknowledgement).



Click to select packet 71271 (37.85s len 1412 seq 4242005 ack 1753 win 1022) → 9058 pkts, 12 MB ← 2558 pkts, 5256 bytes

Figure 3.7 TCP round trip time Host A to Host B

Transmission Control Protocol Traffic Analysis

1.6.4 The Purpose of TCP in internet communication

This section explores the process of investigating the collected packets to define the key performance indicators (KPI) in the transport layer, more specifically the (TCP). This is the most common means of determining whether a TCP connection is functioning efficiently or not. This section will demonstrate the problematic side of a TCP segment trace analysis and the difficulties arising. It also illustrates the conversion of data into a graphic representation that makes the study of packets much more manageable and readable. RFC-793 defines the TCP. It provides an unreliable datagram service and two directional reliable byte streams to the internet communication. TCP is a suitable protocol to investigate because its use is common in all aspects of networking communication.

1.6.5 TCP Packet trace

Multiple techniques allow the exploration of TCP performance testing, including manual analysis of appropriate collected TCP packets from network monitoring devices. This method uses timestamps and some important part of the packet for examination. The conversation within the packet can be reconstructed by examining the packet. By rebuilding the story of the trace from the raw data, it is often easy to understand the cause of a connection performing efficiently or otherwise.

It is challenging to understand how packets relate to each other in time sequence and acknowledgment numbers when using the manual method of performing trace analysis. Extracting these relationships manually is time-consuming. Previous experience indicates that it is possible to spend several hours trying to understand only one section of a TCP packet trace when dealing with performance issues. It is very time consuming for an analyst to reassemble and rebuild the purpose of each packet. This process can be illustrated by recognizing the segment with RST (Reset) acknowledgment set to one with a previous packet. This means of examining packet traces leads to a tedious task of recreating these relationships between TCP segments. However, expert knowledge can be used to capture and create an automated analysis tool to solve this problem.

Wireshark is used to scan TCP Packets, detecting and organizing common phenomena in the packet, mark them provides good description of the trace. This approach of sniffing internet packets, allows experts and non-analysts to detect and segregate performance issues in the connection. The use of *Wireshark* improves the ability to present information to users, without trying to implant knowledge about the analysis into the tools. By improving the form in which the packet trace is presented to the analyst and by providing some tools for its deployment, the problem of information overload dramatically decreases.

1.6.6 Detailed description of TCP

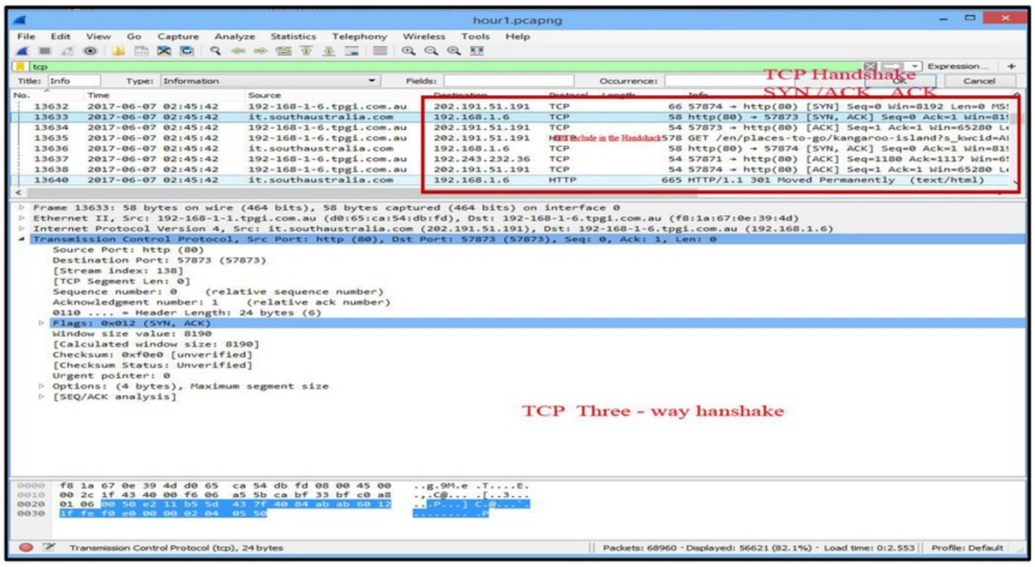
When using the internet protocol (IP), the delivery service is unreliable since the IP has no mechanism for controlling packet delivery. However, there is an expectation that packets will be reliably delivered in the internet protocol network. During transmission, the datagrams (IP) may be impaired by being corrupted, lost, disordered or duplicated. But with the use of a series of controllers such as sequence numbers, acknowledgments, checksums and windows, the TCP provides a reliable service with end-to-end flow control.

The TCP session offers a bidirectional connection, where the acknowledgment followed by the windows update in one direction is generally followed by the data moving in the opposite direction. The TCP protocol at each end of the link communicates by using an IP module to exchange TCP segment information to each other.

The data bytes (8 bits) in the stream are numbered in sequence so that each octet is represented by a 32-bit sequence number. A TCP header is provided with a 32-bit sequence number field, which contains the sequence number of the Reset data byte carried by the segment. In the absence of data to be sent, then the sender to the sequence number of the Reset byte not yet sent sets the sequence number in the TCP header. This type of packet with no data is used when control information needs to be transported to the other end of the connection and there is no data attached to it.

TCP is the standard that defines how to establish and maintain a network conversation. This protocol offers a connection-oriented transport between hosts. TCP supports the process of sending large numbers of data packets in sequence without waiting for acknowledgement from the receiver. During the three way–handshake, the size of the window is defined by the network activities. TCP is used by most file transfer protocols to ensure the reliability of data delivery. The TCP protocol offers transport for the following applications: HTTP, HTTPS, FTP, POP/SMTP, [32] and more.

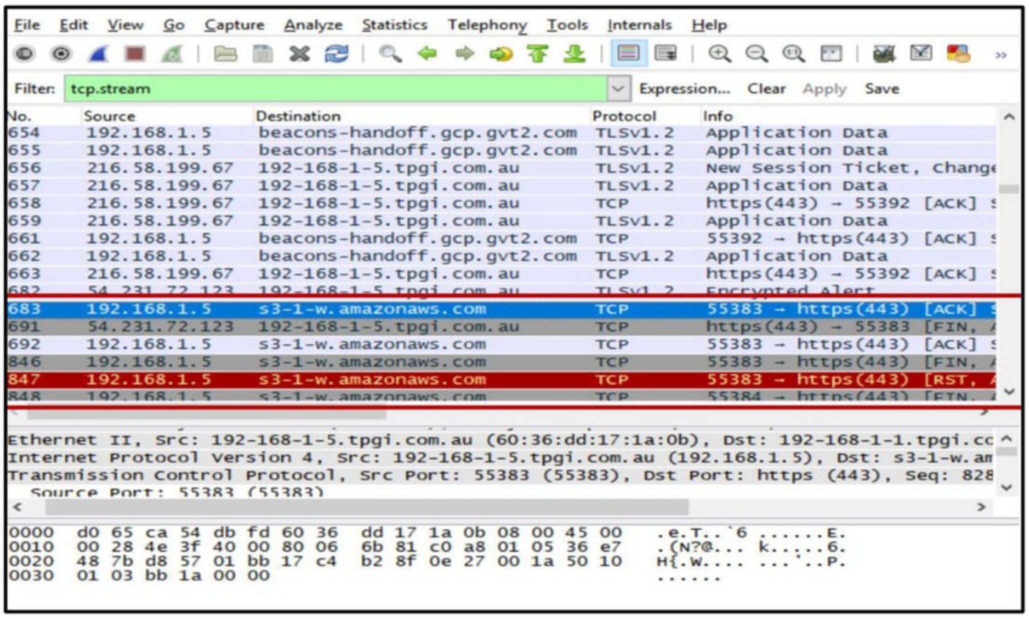
1.6.6.1 TCP packet analysis



```
No.    Time                 Source                      Destination      Protocol Length Info
13632  2017-06-07 02:45:42  192-168-1-6.tpgi.com.au     202.191.51.191   TCP      66 57874 → http(80) [SYN] Seq=0 Win=8192 Len=0 MS
13633  2017-06-07 02:45:42  it.southaustralia.com       192.168.1.6      TCP      58 http(80) → 57873 [SYN, ACK] Seq=0 Ack=1 Win=81
13634  2017-06-07 02:45:42  192-168-1-6.tpgi.com.au     202.191.51.191   TCP      54 57873 → http(80) [ACK] Seq=1 Ack=1 Win=65280 L
13635  2017-06-07 02:45:42  192-168-1-6.tpgi.com.au     202.191.51.191   HTTP    578 GET /en/places-to-go/kangaroo-island?s_kwcid=A
13636  2017-06-07 02:45:42  it.southaustralia.com       192.168.1.6      TCP      58 http(80) → 57874 [SYN, ACK] Seq=0 Ack=1 Win=81
13637  2017-06-07 02:45:42  192-168-1-6.tpgi.com.au     192.243.232.36   TCP      54 57871 → http(80) [ACK] Seq=1180 Ack=1117 Win=6
13638  2017-06-07 02:45:42  192-168-1-6.tpgi.com.au     202.191.51.191   TCP      54 57874 → http(80) [ACK] Seq=1 Ack=1 Win=65280 L
13640  2017-06-07 02:45:42  it.southaustralia.com       192.168.1.6      HTTP    665 HTTP/1.1 301 Moved Permanently  (text/html)

▷ Frame 13633: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface 0
▷ Ethernet II, Src: 192-168-1-1.tpgi.com.au (d0:65:ca:54:db:fd), Dst: 192-168-1-6.tpgi.com.au (f8:1a:67:0e:39:4d)
▷ Internet Protocol Version 4, Src: it.southaustralia.com (202.191.51.191), Dst: 192-168-1-6.tpgi.com.au (192.168.1.6)
◢ Transmission Control Protocol, Src Port: http (80), Dst Port: 57873 (57873), Seq: 0, Ack: 1, Len: 0
     Source Port: http (80)
     Destination Port: 57873 (57873)
     [Stream index: 138]
     [TCP Segment Len: 0]
     Sequence number: 0    (relative sequence number)
     Acknowledgment number: 1    (relative ack number)
     0110 .... = Header Length: 24 bytes (6)
   ▷ Flags: 0x012 (SYN, ACK)
     Window size value: 8190
     [Calculated window size: 8190]
     Checksum: 0xf0e0 [unverified]
     [Checksum Status: Unverified]
     Urgent pointer: 0
   ▷ Options: (4 bytes), Maximum segment size
   ▷ [SEQ/ACK analysis]

0000  f8 1a 67 0e 39 4d d0 65  ca 54 db fd 08 00 45 00   ..g.9M.e .T....E.
0010  00 2c 1f 43 40 00 f6 06  a5 5b ca bf 33 bf c0 a8   .,.C@... .[..3...
0020  01 06 00 50 e2 11 b5 5d  43 7f 40 84 ab ab 60 12   ...P...] C.@...`.
0030  1f fe f0 e0 00 00 02 04  05 50                     ........ .P

Transmission Control Protocol (tcp), 24 bytes    Packets: 68960 · Displayed: 56621 (82.1%) · Load time: 0:2.553    Profile: Default
```

Figure 3.8 TCP three -way handshake between hosts (192.168.1.6 and 202.191.51.191)

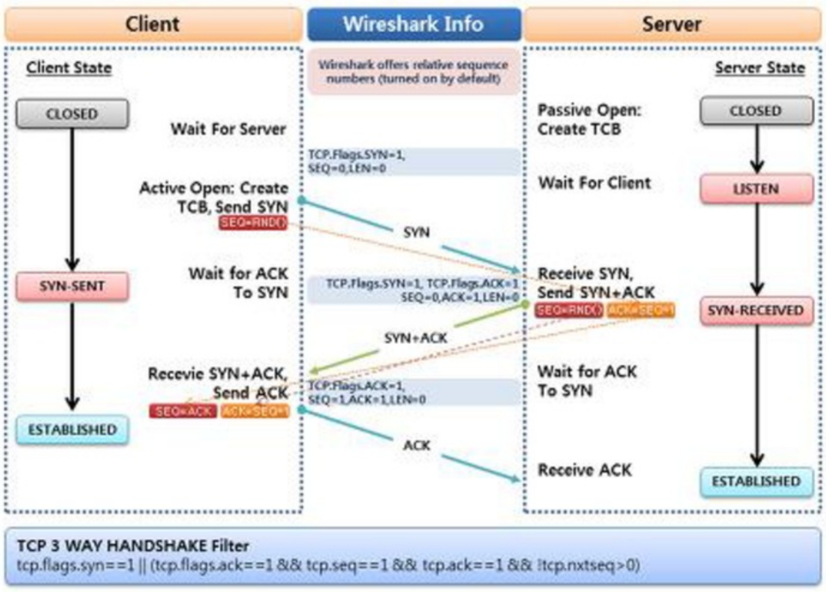
As mentioned above, the connection between the two hosts is established through the three-way handshake process, illustrated with SYN, SYN/ACK and ACK as shown in figure 3.8. In the handshake process, the SYN (synchronization) packet synchronizes the sequence number to establish the connection between the two hosts and to ensure that both end devices know each other's initial sequence numbers (ISN). Figure 3.8 illustrates the TCP three-way handshake used to establish the connection between host B (IP source IP address 192. 168.1.6 and the web server (destination IP address 202.191.51.191). The host with IP address 192.168.1.6 establishes the TCP connection to the Web server with IP address 202.191.51.191. Packet number 13632 contains the designation (SYN); the next packet number 13633 contains (SYN and ACK) and packet number 13634 lists (ACK). These three sequenced line numbers represent the three-way handshake pattern used to establish a connection. This connection is established, and the webserver sends back the requested website to view. In the same file trace, it was noticed that when the SYN does not obtain a response from the target destination, it automatically retransmits the SYN in another attempt to establish a connection.


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: tcp.stream
Expression... Clear Apply Save

```
No.   Source          Destination                    Protocol  Info
654   192.168.1.5     beacons-handoff.gcp.gvt2.com   TLSv1.2   Application Data
655   192.168.1.5     beacons-handoff.gcp.gvt2.com   TLSv1.2   Application Data
656   216.58.199.67   192-168-1-5.tpgi.com.au        TLSv1.2   New Session Ticket, Change
657   216.58.199.67   192-168-1-5.tpgi.com.au        TLSv1.2   Application Data
658   216.58.199.67   192-168-1-5.tpgi.com.au        TCP       https(443) → 55392 [ACK] S
659   216.58.199.67   192-168-1-5.tpgi.com.au        TLSv1.2   Application Data
661   192.168.1.5     beacons-handoff.gcp.gvt2.com   TCP       55392 → https(443) [ACK] S
662   192.168.1.5     beacons-handoff.gcp.gvt2.com   TLSv1.2   Application Data
663   216.58.199.67   192-168-1-5.tpgi.com.au        TCP       https(443) → 55392 [ACK] S
682   54.231.72.123   192-168-1-5.tpgi.com.au        TLSv1.2   Encrypted Alert
683   192.168.1.5     s3-1-w.amazonaws.com           TCP       55383 → https(443) [ACK] S
691   54.231.72.123   192-168-1-5.tpgi.com.au        TCP       https(443) → 55383 [FIN, A
692   192.168.1.5     s3-1-w.amazonaws.com           TCP       55383 → https(443) [ACK] S
846   192.168.1.5     s3-1-w.amazonaws.com           TCP       55383 → https(443) [FIN, A
847   192.168.1.5     s3-1-w.amazonaws.com           TCP       55383 → https(443) [RST, A
848   192.168.1.5     s3-1-w.amazonaws.com           TCP       55384 → https(443) [FIN, A

Ethernet II, Src: 192-168-1-5.tpgi.com.au (60:36:dd:17:1a:0b), Dst: 192-168-1-1.tpgi.cc
Internet Protocol Version 4, Src: 192-168-1-5.tpgi.com.au (192.168.1.5), Dst: s3-1-w.am
Transmission Control Protocol, Src Port: 55383 (55383), Dst Port: https (443), Seq: 828
  Source Port: 55383 (55383)

0000  d0 65 ca 54 db fd 60 36  dd 17 1a 0b 08 00 45 00   .e.T..`6 ......E.
0010  00 28 4e 3f 40 00 80 06  6b 81 c0 a8 01 05 36 e7   .(N?@... k.....6.
0020  48 7b d8 57 01 bb 17 c4  b2 8f 0e 27 00 1a 50 10   H{.W.... ...'..P.
0030  01 03 bb 1a 00 00                                  ......
```

Figure 3.9 TCP RESET used to close a TCP connection session

Terminating the connection can be achieved in several ways with the use of (TCP FIN) packets. When the FIN packet is deployed, a host sends a FIN packet and enters a FIN-WAIT state until its receipt is acknowledged. Within the TCP connection, the RESET can be used to explicitly end the connection. Figure 3.9 illustrates packet number 691. Source host with IP address 192.168.1.5 shows the HTTP based protocol connection created by the destination host with IP address (S3-1-W.amazonaaws.com) web server. In packet 847, the web server sent a packet containing the letters (FIN) indicating the end of the data packet. The host client sent the acknowledgement bit and the FIN bit in packet 846. Subsequently, the server in packet 847 sent a TCP Reset (RST) to terminate the connection.


Client
Wireshark Info
Server
Client State
Wireshark offers relative sequence numbers (turned on by default)
Server State
CLOSED
Wait For Server
Passive Open: Create TCB
CLOSED
TCP.Flags.SYN=1, SEQ=0,LEN=0
Wait For Client
LISTEN
Active Open: Create TCB, Send SYN
SEQ=RND()
SYN
SYN-SENT
Wait for ACK To SYN
TCP.Flags.SYN=1, TCP.Flags.ACK=1 SEQ=0,ACK=1,LEN=0
Receive SYN, Send SYN+ACK
SEQ=RND()
ACK=SEQ+1
SYN-RECEIVED
SYN+ACK
Recevie SYN+ACK, Send ACK
SEQ=ACK
ACK=SEQ+1
TCP.Flags.ACK=1, SEQ=1,ACK=1,LEN=0
Wait for ACK To SYN
ESTABLISHED
ACK
Receive ACK
ESTABLISHED
TCP 3 WAY HANDSHAKE Filter

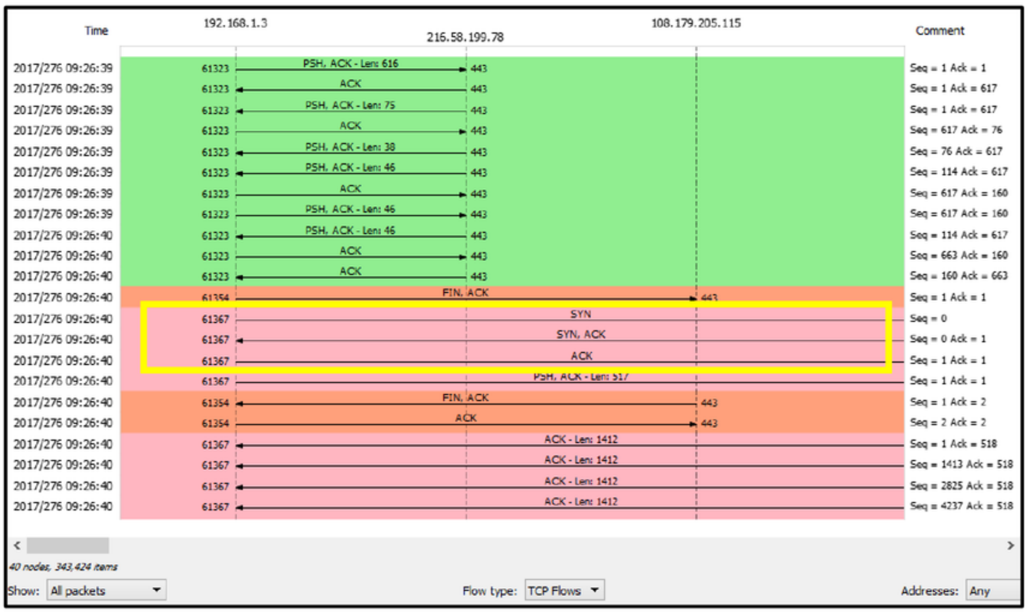
```
tcp.flags.syn==1 || (tcp.flags.ack==1 && tcp.seq==1 && tcp.ack==1 && !tcp.nxtseq>0)
```

Figure 3.10 TCP Protocol 3 way-handshake

1.6.7 TCP packet sequence tracking

During the handshake process, the transmitter and the receiver connection own their own initial sequence number. Each end host increases its sequence number according to the amount of data contained in its packet. Sequencing and acknowledgment process analysis is based on the following formula:

$$= \frac{+\ sequence\ Number\ in\ Byte\ of\ data\ recieved}{Acknowlegment\ \ number\ out}$$



```
Time                  192.168.1.3        216.58.199.78            108.179.205.115        Comment

2017/276 09:26:39     61323 --PSH, ACK - Len: 616--> 443                               Seq = 1 Ack = 1
2017/276 09:26:39     61323 <--ACK------------------ 443                               Seq = 1 Ack = 617
2017/276 09:26:39     61323 <--PSH, ACK - Len: 75--- 443                               Seq = 1 Ack = 617
2017/276 09:26:39     61323 --ACK------------------> 443                               Seq = 617 Ack = 76
2017/276 09:26:39     61323 <--PSH, ACK - Len: 38--- 443                               Seq = 76 Ack = 617
2017/276 09:26:39     61323 <--PSH, ACK - Len: 46--- 443                               Seq = 114 Ack = 617
2017/276 09:26:39     61323 --ACK------------------> 443                               Seq = 617 Ack = 160
2017/276 09:26:39     61323 --PSH, ACK - Len: 46---> 443                               Seq = 617 Ack = 160
2017/276 09:26:40     61323 <--PSH, ACK - Len: 46--- 443                               Seq = 114 Ack = 617
2017/276 09:26:40     61323 --ACK------------------> 443                               Seq = 663 Ack = 160
2017/276 09:26:40     61323 <--ACK------------------ 443                               Seq = 160 Ack = 663
2017/276 09:26:40     61354 ----------FIN, ACK-------------------------> 443           Seq = 1 Ack = 1
2017/276 09:26:40     61367 ----------------------SYN--------------------------------  Seq = 0
2017/276 09:26:40     61367 <---------------------SYN, ACK---------------------------  Seq = 0 Ack = 1
2017/276 09:26:40     61367 ----------------------ACK--------------------------------  Seq = 1 Ack = 1
2017/276 09:26:40     61367 ----------------------PSH, ACK - Len: 517----------------  Seq = 1 Ack = 1
2017/276 09:26:40     61354 <---------FIN, ACK-------------------------- 443           Seq = 1 Ack = 2
2017/276 09:26:40     61354 ----------ACK------------------------------> 443           Seq = 2 Ack = 2
2017/276 09:26:40     61367 <---------------------ACK - Len: 1412--------------------  Seq = 1 Ack = 518
2017/276 09:26:40     61367 <---------------------ACK - Len: 1412--------------------  Seq = 1413 Ack = 518
2017/276 09:26:40     61367 <---------------------ACK - Len: 1412--------------------  Seq = 2825 Ack = 518
2017/276 09:26:40     61367 <---------------------ACK - Len: 1412--------------------  Seq = 4237 Ack = 518

40 nodes, 343,424 items
Show: All packets            Flow type: TCP Flows            Addresses: Any
```

Figure 3.11 TCP sequence and acknowledgement number data exchange

The ACK number field has the value of the following sequence in the queue from the other end. This ACK number only increases when the data is received. The numbering sequence used in *Wireshark* can vary; to facilitate reading, the initial sequence number is set to 0. During the three-way handshake, the sequence increases by the value of 1. After the process is established, the SEQ (sequence number) only increases by the value of the current data sent. Figure 3.11 shows how the acknowledgment number and the sequence number increase with the data exchanged. The figure illustrates the connection between two end applications. The Client on port 6167 and server on port 443 which is used by QUIC (quick UDP internet connection). The first handshake is marked inside the yellow rectangle with sequence number = 0, sent by the client computer and ACK =1, the server replied and accepted the first contact by sending Seq =1 for the expected incoming segment and ACK=1 on the (SYN ACK). Finally, the client sent back another ACK to establish the connection with Seq=1 and ACK =1. When both ends agree on the sequence number to be used and the size of the scaling window, they start exchanging real data. This explains the fact that the acknowledgment of multiple data packets can be validated with a single ACK.

1.6.8 TCP concepts

This section presents a brief overview of various TCP congestion avoidance algorithms, flow control and different TCP variables evaluated later in this project work. The flow control and congestion avoidance mechanism on TCP prevent a sender within the connection from sending amounts of data that could overload the receiver. TCP likewise provides sliding window mechanism to maintain load balance between the sender and the receiver. The following section analyses the congestion control and flow control method (algorithm) observed in the TCP/IP network packet obtained from *Wireshark*'s file capture.

1.6.8.1 TCP congestion avoidance algorithms

"The TCP congestion avoidance algorithm is the primary basis for congestion control in the internet "[33]. The congestion avoidance algorithm detects congestion by observing retransmission timer expiration and the reception of duplicate ACKs. To remedy this situation, the sender decreases its transmission window, namely, the number of unacknowledged packets in transit, to one half of the current window size.

There are various types of TCP congestion avoidance algorithms. For example: Tahoe, Reno, New Reno, Vegas, BIC, CUBIC, H-TCP etc. In this project, three algorithms were chosen for KPI analysis: *CUBIC*, *Reno* and *H-TCP*.

In TCP, for each received ACK, the congestion window is normally increased by one segment per round trip time (RTT); this mechanism is called *Slow Start*. On the other hand, when packet loss occurs, TCP applies a mechanism called *Multiplicative Decrease Congestion Avoidance* (MDCA) which decreases the congestion window to half the round-trip time [27].

Timeout occurs when the sender does not receive the ACK within a given time. This initiates the retransmission of the lost segment. Fortunately, *Reno* has a fast retransmission feature which reduces the time a sender waits before retransmitting. But the MDCA mechanism does not work properly in *Reno*, so the fast-retransmitted packets also start dropping at the receiver end. Because the retransmission rate is not aligned with the receiver's receiving capability, this results in a large number of packet drops. Therefore, *Reno* only performs well with very small volumes of packet loss [27].

TCP variables

Several TCP variables are rumored to impact on TCP performance. Since TCP variables control different structures, it is believed that they should have some influence on network performance. To verify that, a combined TCP algorithm was investigated; this is reported on in the next section.

1.6.8.1.1 Studies of TCP_window_scaling Algorithm

TCP_window_scaling is the procedure that enables a TCP decision to adjust window size so it can accommodate what are called "Large Fat Pipes (LFP)". TCP could be subject to bandwidth failure. packets could be completely lost during transmission through large channels, as these channels are not fully occupied when expecting ACK's from segments previously transmitted. The use of the scaling factor is provided through the TCP window scaling feature; for oversized windows, this protocol guarantees all available bandwidth utilization.

The default setting of tcp_window_scaling is the flag true or the value one and set to zero when turning off [25]. This feature was studied to evaluate and expend system performance and packet loss while capturing large amounts of data through the internet network.

1.6.8.2 Testing TCP performance

A TCP connection was established between the two end hosts. The Network Interface Card (NIC) was configured for 100Mbit/s on Host A and 1000Mbit/s on Host C. Later, it was modified to be the same for all hosts. Figure 3.12 (below) presents a physical image of the network.

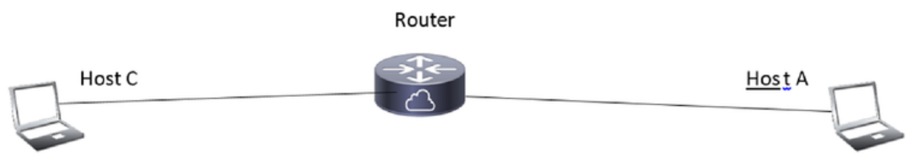


Figure 3.12 Test network

1.6.8.3 Measuring the impact of TCP drops

For the experimental study, TCP variables were filtered for different value. Wireshark was used to perform measurements on each variable. Observations indicated that performance decreased with the increase in the number of packets lost. Performance reached 98Mbits/s when no loss was observed, but reduced to 5 Mbits/s after experiencing 20% of packet loss.

1.6.8.4 Experiment when data link experienced data loss

In this investigation, the filter was set at the values shown in figure 3.13 below. As a result, the network experienced significant packet drop and data loss.

Segment Lost %	Bandwidth (Mbits/s)
0	95
1	85
2	56
4	22
10	5
20	1.5
30	0.05

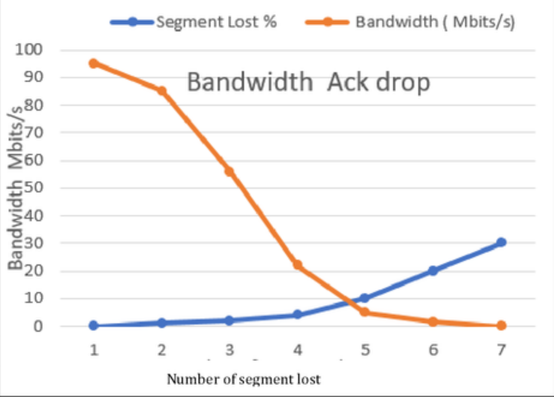
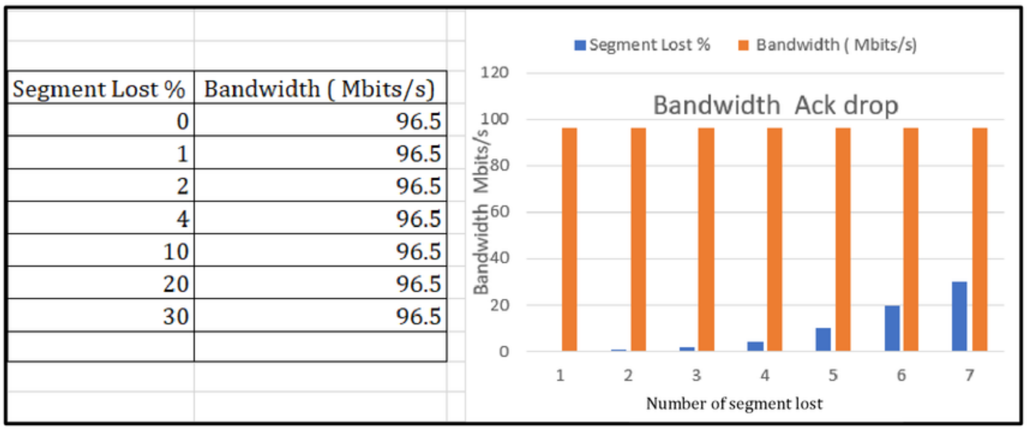


Figure 3.13 Volume of data reduction

Figure 3.13 reveals that performance reduced markedly with the increase in data loss during the transmission. At 0% of packet loss, the bandwidth was almost entirely used during transmission. Unexpectedly, when the loss increased by 1%, the performance reduced to 85 Mbits/s and continued to deteriorate to almost 0 Mbits/s once losses reached 30%. This could have been due to the fact that during the capture, there was some interference in the transmission.

1.6.8.5 Experiment when data link experienced reduction in ACK

The same values as in the previous test were used to detect performance when the system experienced a reduction in acknowledgments. Surprisingly, no change was observed.



Segment Lost %	Bandwidth (Mbits/s)
0	96.5
1	96.5
2	96.5
4	96.5
10	96.5
20	96.5
30	96.5

Figure 3.14 Volume of ACK reduction

The drastic reduction observed in the UDP/IP transmission when the system was experiencing increased packet loss, warranted further investigation. To authenticate this drop in performance, measurements were taken and changes were made to these experiments. The MTU was set to 1400 for Hosts A and C and the router used the usual FTP protocol. No change was observed.

1.6.8.6 Tests with Cubic when experiencing data drop

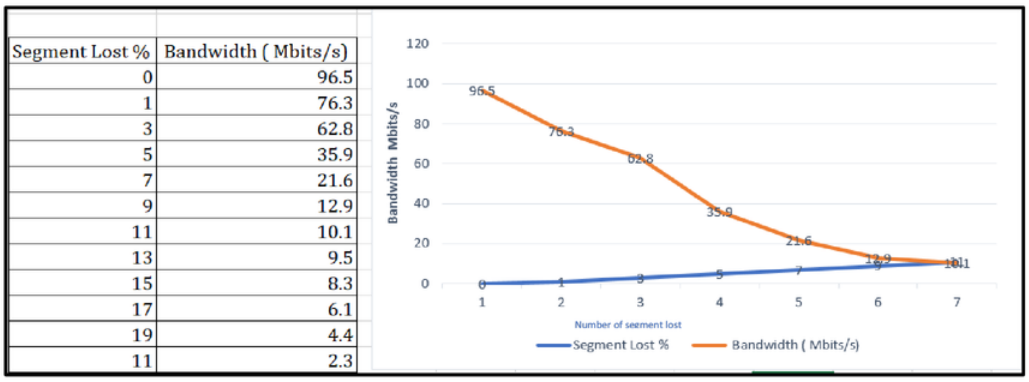
The algorithm Cubic was selected because of its ability to prevent congestion in the systems running on the two hosts and in the router. For the experiment, the following values were set:

Initialized Value: 0

Captured Data Size =25 MB

MTU = 1400 for all hosts

While capturing traffic for 30 minutes, *Wireshark* was used for packet inspection throughout the network. As a result, *Wireshark* took more time than expected to complete the packet captures for each test. It was assumed that the large volume of data was generating an overflow in the data path resulting in the dramatic reduction in bandwidth.



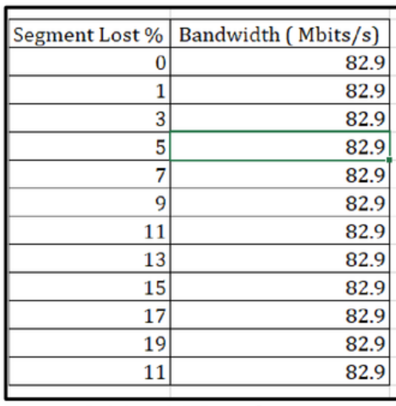
Segment Lost %	Bandwidth (Mbits/s)
0	96.5
1	76.3
3	62.8
5	35.9
7	21.6
9	12.9
11	10.1
13	9.5
15	8.3
17	6.1
19	4.4
11	2.3

Figure 3.15 Volume of data reduction in Cubic

The network's performance when the connection was experiencing data loss is shown in figure 3.15. Once again, performance decreased drastically with increasing data loss.

1.6.8.7 Tests with Cubic when experiencing ACK drop

A similar experiment to that adopted with the Cubic programme was conducted with the decrease in ACK messages. The result obtained is presented in the next figures.



Segment Lost %	Bandwidth (Mbits/s)
0	82.9
1	82.9
3	82.9
5	82.9
7	82.9
9	82.9
11	82.9
13	82.9
15	82.9
17	82.9
19	82.9
11	82.9

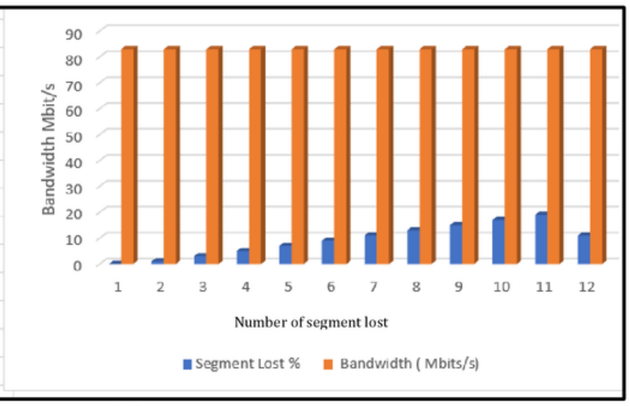


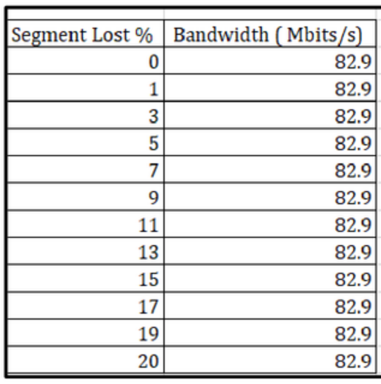
Figure 3.16 ACK performance reduction in Cubic

Figure 3.16 shows that performance remained unchanged at 82.9 Mbits/s even when the system experienced ACK loss. The trials with ACK loss as seen in the graphs above suggest that changing the values of TCP variables has little impact on performance when experiencing ACK loss.

1.6.8.8 Experiments with different congestion avoidance algorithms for ACK loss

1.6.8.8.1 Cubic and Reno

The two congestion avoidance algorithms produced the same experimental results when the volume of Acknowledgements reduced. All TCP variables were filtered out for this experiment.



Segment Lost %	Bandwidth (Mbits/s)
0	82.9
1	82.9
3	82.9
5	82.9
7	82.9
9	82.9
11	82.9
13	82.9
15	82.9
17	82.9
19	82.9
20	82.9

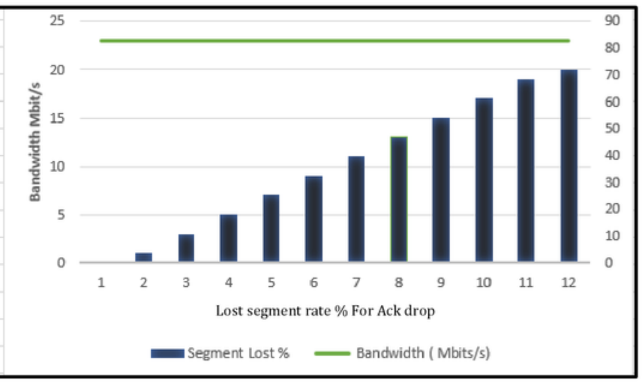
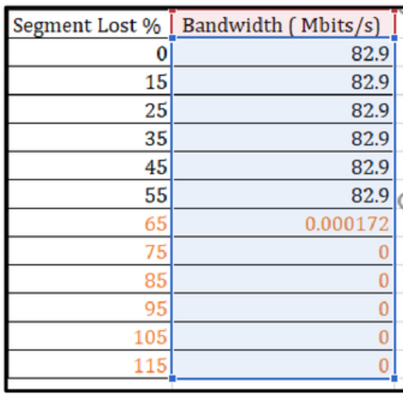


Figure 3.17 ACK loss with Cubic and Reno

Figure 3.17 shows the system performance while the connection was experiencing ACK loss. The data reveal that there was no change in performance regardless of whether the loss was 1% or 10% for all congestion avoidance algorithms. This experiment shows that even a relatively high amount of acknowledgement loss has no impact on performance. In order to further test this claim, additional experiments with ACK loss were conducted.

Figure 3.17 reveals that a small amount of ACK loss had no impact on network performance. Accordingly, the volume of ACK loss was increased ten times from the previous test, introducing 10%, 20%, 30%...100% of ACK loss to the network instead of 1%, 2%, 3%...10%. Surprisingly, there was no change in bandwidth. The data obtained from this experiment are presented in figure 3.19 below.



Segment Lost %	Bandwidth (Mbits/s)
0	82.9
15	82.9
25	82.9
35	82.9
45	82.9
55	82.9
65	0.000172
75	0
85	0
95	0
105	0
115	0

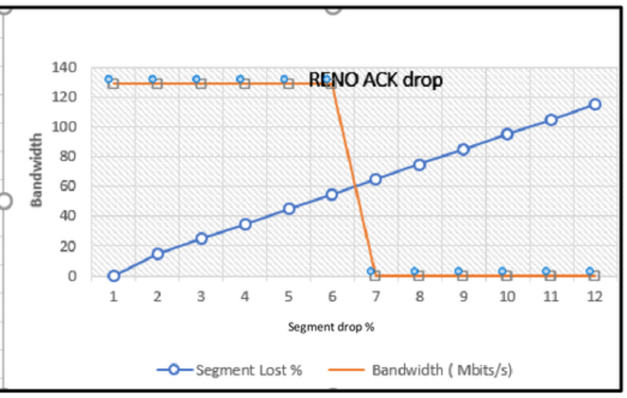


Figure 3.18 ACK loss with Reno with TCP variables turned off

The data in figure 3.18 reveal that, after increasing the amount of ACK loss, performance remained unchanged until 50% of ACK had been lost. At 60% of ACK loss, performance dropped to almost 0 (i.e. 0.0000004).

Similar tests were conducted with the four TCP variables by turning them on separately in each test. To compare the performance of the four TCP variables (when turned on) with the results shown in figure 3.19, tests (3.7.5.6, 3.7.5.7, 3.7.5.8, 3.7.5.9) were done.

1.6.8.8.2 Tests with different TCP variables: tcp_window_scaling

The first test was conducted with tcp_window_scaling = 1 for ACK loss

Segment Lost %	Bandwidth (Mbits/s)
0	8.89
15	8.89
25	8.89
35	8.89
45	8.89
55	8.89
65	0.00017
75	0
85	0
95	0
105	0
115	0

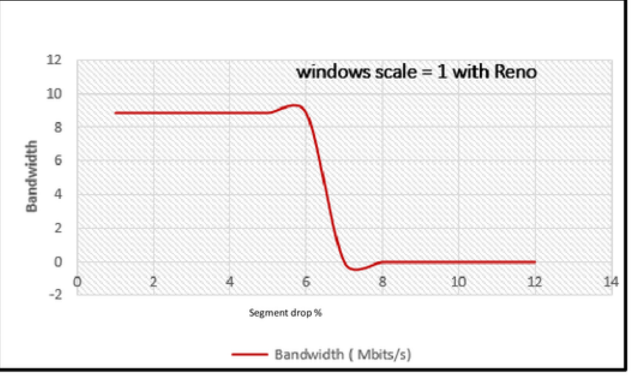
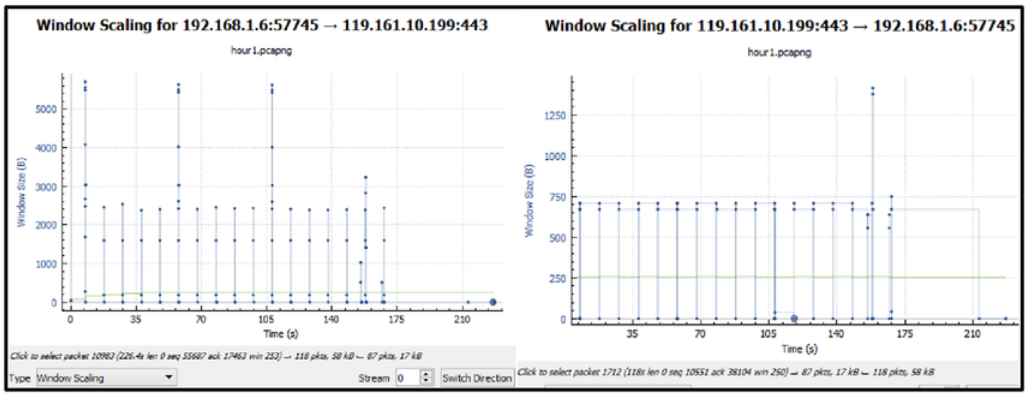


Figure 3.19a Performance of tcp_window_scaling with variables turned off



Window Scaling for 192.168.1.6:57745 → 119.161.10.199:443
hour1.pcapng
Window Size (B)
Time (s)
Window Scaling for 119.161.10.199:443 → 192.168.1.6:57745
hour1.pcapng
Window Size (B)
Time (s)
Type Window Scaling
Stream 0
Switch Direction

Click to select packet 10983 (226.4s len 0 seq 55687 ack 17463 win 253) → 118 pkts, 58 kB ← 87 pkts, 17 kB

Click to select packet 1712 (118s len 0 seq 10551 ack 38104 win 250) → 87 pkts, 17 kB ← 118 pkts, 58 kB

Figure 3.19b Performance of tcp_window_scaling with variables turned on

After increasing the loss, performance remained unchanged till 30% of ACK were lost, although performance decreased slightly when 40% of acknowledgments were lost. As figure 3.19a demonstrates, increasing the ACK loss to 60% caused performance to drop to almost 0 (i.e. 0.00014).

The second figure (3.19b) shows the variation in the window size controls at the receiver end. For the first transmission, the size was set to one. This first TCP segment incorporates ACK from the receiver with all the transmission information (such as window size) required by the receive end. After 5 to 50 seconds the size stays constant at 748 Bytes and can transmit up to two segments followed by one ACK at a time. At 53 seconds, we observed a sudden change to 1550 Bytes for a maximum of 4 segments transmitted at a time. This variation could be due to packet loss and the retransmission of lost packets. The connections closed at 225 seconds with the last ACK segment received at the receiver end. Figure 3.20b shows that the receiver window opens and closes, and the transmission happens segment by segment with the variable controlling the window size expressed in Bytes.

1.6.8.8.3 Tests with different TCP variables: tcp_adv_win_scale

In this test, tcp_adv_win_scale was turned on. The meaning of this value was discussed in section 3.7.5. during the experiments in data loss. The same experiments were conducted for ACK loss to observe its impact on performance.

Segment Lost %	Bandwidth (Mbits/s)
0	8.89
15	8.89
25	8.89
35	8.89
45	7.2
55	8.89
65	0.00017
75	0
85	0
95	0
105	0
115	0

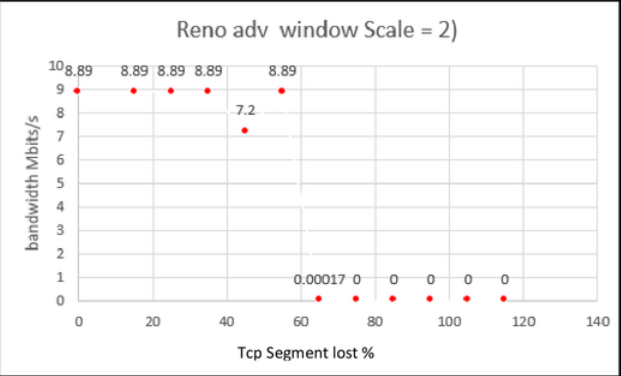
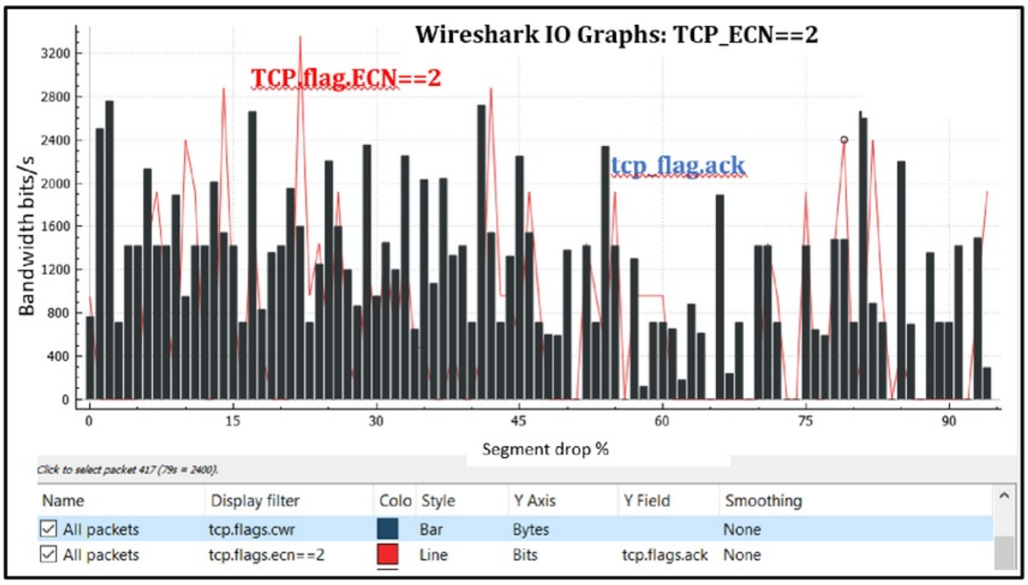


Figure 3.19c: Performance of tcp_adv_win_scale with variables turned on

Figure 3.19c shows a reduction in performance at 40% ACK loss. Otherwise, there is no significant difference from earlier experiments. As before, when reaching 60% loss, performance reduced to 0 (0.001535).

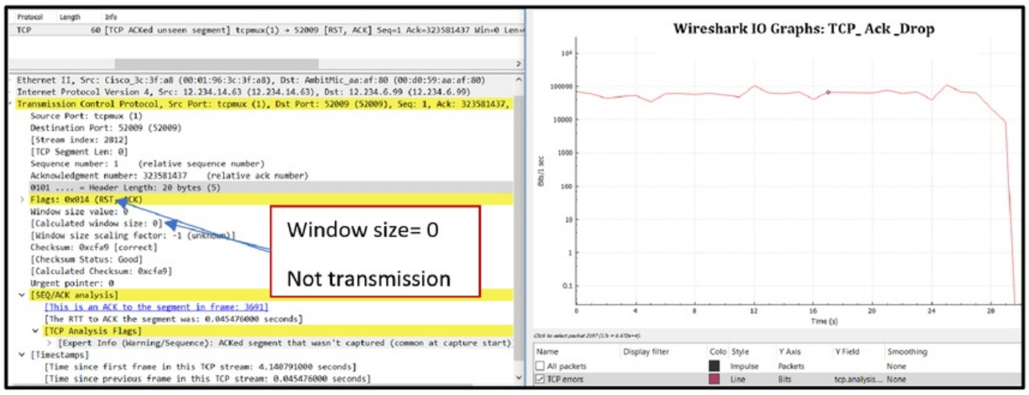
1.6.8.8.4 Tests with different TCP variables: tcp_ecn

In this test, tcp_ecn (Explicit Congestion Notification) was turned on, so that explicit notifications were delivered while congestion occurred.



Name	Display filter	Colo	Style	Y Axis	Y Field	Smoothing
☑ All packets	tcp.flags.cwr	■	Bar	Bytes		None
☑ All packets	tcp.flags.ecn==2	🟥	Line	Bits	tcp.flags.ack	None

Figure 3.20a tcp_ecn = 2 for ACK loss from *Wireshark* capture



```
Protocol  Length  Info
TCP           60 [TCP ACKed unseen segment] tcpmux(1) → 52009 [RST, ACK] Seq=1 Ack=323581437 Win=0 Len=

Ethernet II, Src: Cisco_3c:3f:a8 (00:01:96:3c:3f:a8), Dst: AmbitMic_aa:af:80 (00:d0:59:aa:af:80)
Internet Protocol Version 4, Src: 12.234.14.63 (12.234.14.63), Dst: 12.234.6.99 (12.234.6.99)
Transmission Control Protocol, Src Port: tcpmux (1), Dst Port: 52009 (52009), Seq: 1, Ack: 323581437,
    Source Port: tcpmux (1)
    Destination Port: 52009 (52009)
    [Stream index: 2812]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 323581437    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x014 (RST, ACK)
    Window size value: 0
    [Calculated window size: 0]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xcfa9 [correct]
    [Checksum Status: Good]
    [Calculated Checksum: 0xcfa9]
    Urgent pointer: 0
  ⌄ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 3691]
      [The RTT to ACK the segment was: 0.045476000 seconds]
    ⌄ [TCP Analysis Flags]
      > [Expert Info (Warning/Sequence): ACKed segment that wasn't captured (common at capture start)]
  ⌄ [Timestamps]
      [Time since first frame in this TCP stream: 4.140791000 seconds]
      [Time since previous frame in this TCP stream: 0.045476000 seconds]
```

Name	Display filter	Colo	Style	Y Axis	Y Field	Smoothing
☐ All packets		■	Impulse	Packets		None
☑ TCP errors		🟥	Line	Bits	tcp.analysis....	None

Figure 3.20b Performance of tcp_ecn with variables turned on

Performance was unchanged up to 50% of ACK loss i.e. 9.37 Mbits/s. When 60% of ACK loss was reached, performance dropped to 7.08 Mbits/s. However, compared with figure 3.21, tcp_ecn could improve the performance at 60% quite significantly. However, it was not able to handle 70% of ACK loss.

To conclude the experiments with ACK loss as seen in the above graphs (Figure 3.20a and Figure 3.20b), changing the values of TCP variables has little impact on performance when experiencing ACK loss. Some additional experiments were conducted with TCP variables both for data loss and ACK loss with Cubic to verify the results of Reno. However, no major differences in results were found.

1.6.9 Time –Sequence graph analysis

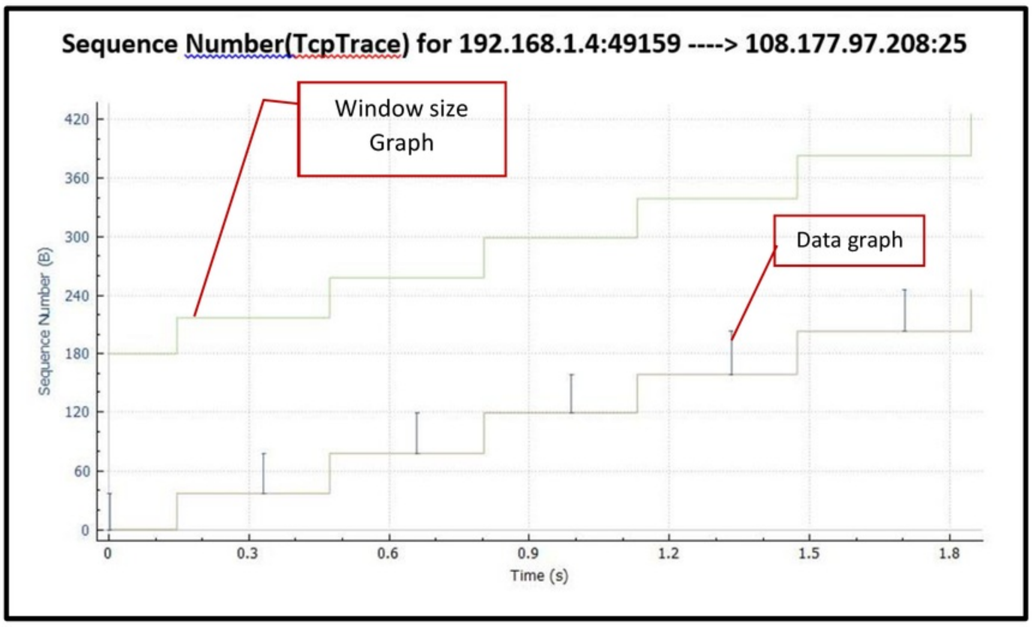


Figure 3.21 Time-sequence plots - examples of typical TCP connections

Figure 3.21 shows packets gathered from the email application (*Thunderbird*) on the active Ethernet link. The packet displayed here explains the performance of TCP connections established between client and server applications. Figure 3.21 illustrates the behavior of a TCP link transferring email data using the SMTP (Single mail transport protocol) port 25. This protocol is used by the email client application to retrieve messages from the mail server. SMTP primarily exchanges very short messages to define who is involved in the conversation. When the email server has completed all verifications, the data is transferred to the server using the application port 25 server side and on port (49159) client side. Once the transfer is successfully completed and data stored on the server, the server replies with an acknowledgement to the client computer that the file has been successfully downloaded. The session is closed when the client computer replies to the FIN ACK.

Figure 3.21 also represents the time-sequence plot of the host sender conversation to the SMTP server. The large quantity of packets sent after 0.3(s) represent the actual contents of the message. The smaller packets at the beginning of the transmission are only packets exchanged to establish the connection between the two ends. Very small packets are those used for acknowledgment. This figure indicates that the size of the file transferred was bigger than Maximum Transfer Unit(MTU) size in byte of the Ethernet network, I TCP communication system the recommend size most be less than 1500 byte. Therefore, the protocol required that the packet be fragmented before it could be transported. Figure 3.22 also shows how close the window size plot is to the data plot. The closer the distance between the two graphs, the smaller the remaining TCP buffering space to allow more byte transfer. When the two graphs are close together, the window-full phenomenon is reached so that no byte transfer can be allowed. Therefore, when the two graphs are far apart, a Sufficient TCP buffer remains, as illustrated in figure 3.22.

1.6.10 Throughput graph analysis

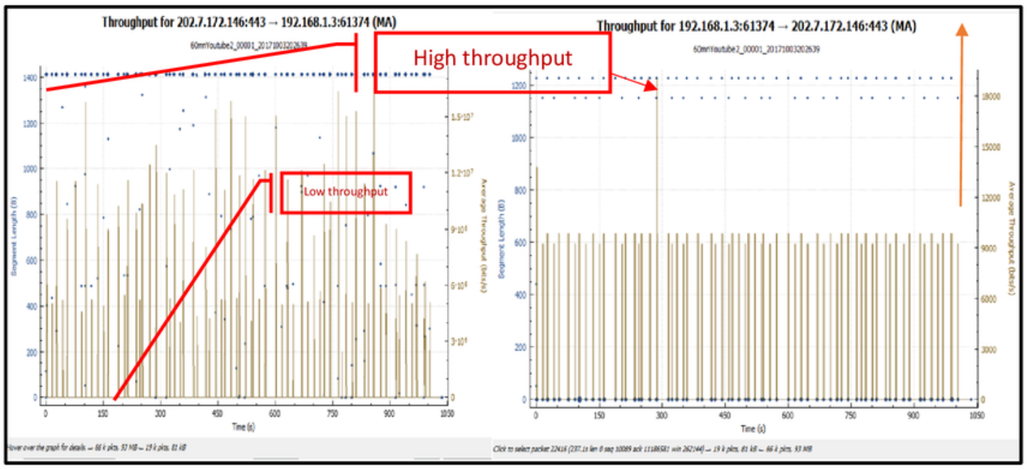
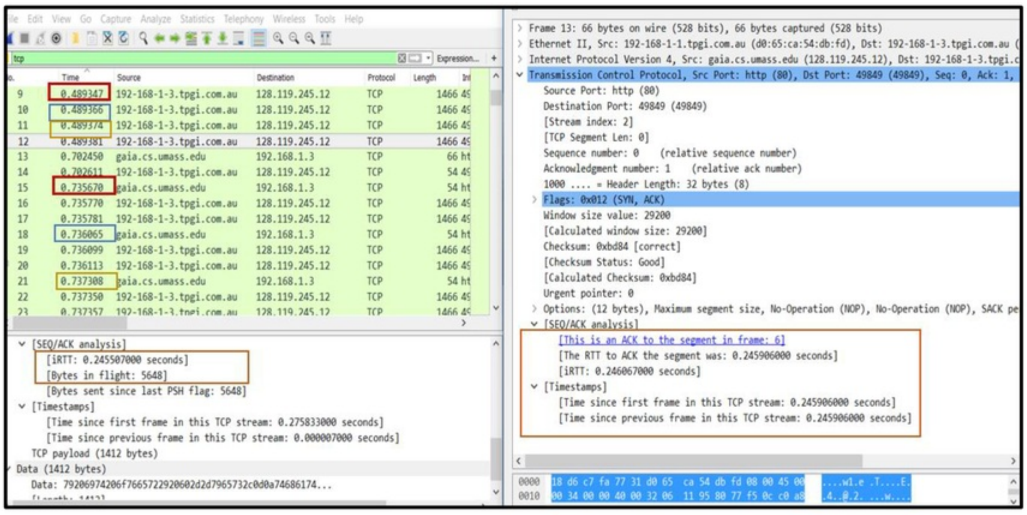


Figure 3.22 Throughput plot of video trace

The throughput graph in figure 3.22 presents the TCP sequence number count over time. Since these sequences are actual application data, this result of this application throughput is expressed in bits/sec. Figure 3.23 also indicating that for this conversation, this throughput is not stable but varies around 1.7x10^7 bits/second or 17Mbytes/sec to 2.5Mb/sec from the server port 433 to the client on port 61374, and goes around 9000 to 10000bit/sec in the opposite direction from client to server. The low throughput can explain that the network is transmitting ACK packets only. The variation noticed on the left-hand side could be caused by unstable file downloads or the way the video application is supposed to work.

1.6.11 Round trip analysis



```
tcp
No.  Time      Source                    Destination      Protocol  Length  In
9    0.489347  192-168-1-3.tpgi.com.au   128.119.245.12   TCP       1466 49
10   0.489366  192-168-1-3.tpgi.com.au   128.119.245.12   TCP       1466 49
11   0.489374  192-168-1-3.tpgi.com.au   128.119.245.12   TCP       1466 49
12   0.489381  192-168-1-3.tpgi.com.au   128.119.245.12   TCP       1466 49
13   0.702450  gaia.cs.umass.edu         192.168.1.3      TCP         66 ht
14   0.702611  192-168-1-3.tpgi.com.au   128.119.245.12   TCP         54 49
15   0.735670  gaia.cs.umass.edu         192.168.1.3      TCP         54 ht
16   0.735770  192-168-1-3.tpgi.com.au   128.119.245.12   TCP       1466 49
17   0.735781  192-168-1-3.tpgi.com.au   128.119.245.12   TCP       1466 49
18   0.736065  gaia.cs.umass.edu         192.168.1.3      TCP         54 ht
19   0.736099  192-168-1-3.tpgi.com.au   128.119.245.12   TCP       1466 49
20   0.736113  192-168-1-3.tpgi.com.au   128.119.245.12   TCP       1466 49
21   0.737308  gaia.cs.umass.edu         192.168.1.3      TCP         54 ht
22   0.737350  192-168-1-3.tpgi.com.au   128.119.245.12   TCP       1466 49
23   0.737357  192-168-1-3.tpgi.com.au   128.119.245.12   TCP       1466 49

  [SEQ/ACK analysis]
    [iRTT: 0.245507000 seconds]
    [Bytes in flight: 5648]
    [Bytes sent since last PSH flag: 5648]
  [Timestamps]
    [Time since first frame in this TCP stream: 0.275833000 seconds]
    [Time since previous frame in this TCP stream: 0.000007000 seconds]
  TCP payload (1412 bytes)
Data (1412 bytes)
  Data: 79206974206f7665722920602d2d7965732c0d0a74686174...
```

```
> Frame 13: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
> Ethernet II, Src: 192-168-1-1.tpgi.com.au (d0:65:ca:54:db:fd), Dst: 192-168-1-3.tpgi.com.au (
> Internet Protocol Version 4, Src: gaia.cs.umass.edu (128.119.245.12), Dst: 192-168-1-3.tpgi.c
v Transmission Control Protocol, Src Port: http (80), Dst Port: 49849 (49849), Seq: 0, Ack: 1,
    Source Port: http (80)
    Destination Port: 49849 (49849)
    [Stream index: 2]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x012 (SYN, ACK)
    Window size value: 29200
    [Calculated window size: 29200]
    Checksum: 0xbd84 [correct]
    [Checksum Status: Good]
    [Calculated Checksum: 0xbd84]
    Urgent pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK pe
  v [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 6]
      [The RTT to ACK the segment was: 0.245906000 seconds]
      [iRTT: 0.246067000 seconds]
  v [Timestamps]
      [Time since first frame in this TCP stream: 0.245906000 seconds]
      [Time since previous frame in this TCP stream: 0.245906000 seconds]

0000  18 d6 c7 fa 77 31 d0 65  ca 54 db fd 08 00 45 00   ....w1.e .T....E.
0010  00 34 00 00 40 00 32 06  11 95 80 77 f5 0c c0 a8   .4..@.2. ...w....
```

Figure 3.23 (SYN ACK) segment sequence number and acknowledgment

The following analysis concerns the TCP segment illustrated in figure 3.23 which contains the HTTP POST as the first segment of the TCP connection. The figure also represents the sequence numbers of the first six segments in the TCP. Table 3.0 below shows the time each ACK segment was sent and each ACK segment received by the client computer with the IP address 192.168.1.3. Data on the time delay between each TCP segment being sent and received, allows us to calculate the round trip time (RTT) value for each segment and evaluate the estimated RTT value after the receipt of each ACK. Figure 3.24 represents graphically the RTT in the TCP connection.

Segment number	ACK sending time	ACK receiving time	Round Trip Time (seconds)
9	0.489347	0.735670	0.2463
10	0.489366	0.736065	0.2466
11	0.489374	0.737308	0.2479
12	0489381	0.739816	0.2489
16	0735770	0.981867	0.2460
17	0.735781	0.983827	0.2480

Table 3.0 Delay between sending and receiving each ACK segment

The difference between the received time and the sending time as shown in Table 3.0 indicates the RTT value for each segment. The estimated RTT value was calculated using the following formula, where alpha is 0.125 is a constant:

EstimatedRTT = (1 - Alpa) * EstimatedRTT (previous)+ Alpha* SampleRTT

As Table 3.0 demonstrates, the calculation of the estimated RTT after ACK segment 9 remains unchanged at 0.2463 seconds.

Segment number	Sample RTT	Estimated RTT (previous)	Estimated RTT
9	0.2463	0.2463	0.2463
10	0.2466	0.2463	0.2463375
11	0.2479	0.2463375	0.246532813
12	0.2489	0.24653281	0.246828711
16	0.246	0.24682871	0.246725122
17	0.248	0.24672512	0.246884482

Table 3.1: Estimated RTT after receiving ACK

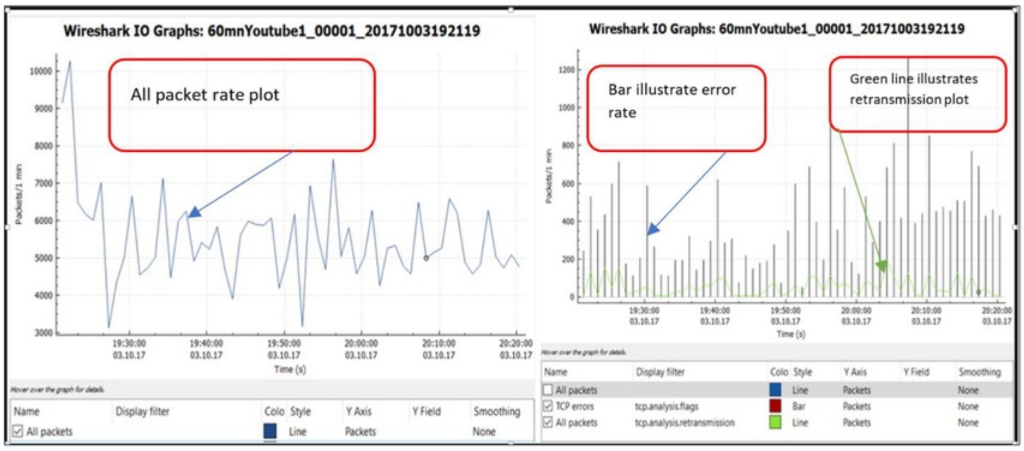
The sample RTT is the RTT for every transmitted segment. Table 3.0 shows how the value of the sample fluctuates from one segment to another possibly due to network congestion or load variation on the end user. It is therefore normal to take the average value of this sample RTT which is the estimated value calculated and illustrated in Table 3.1. Along with the TCP stream graph, the RTT provides a good understanding of the network's performance.

1.6.12 TCP Retransmission

When the TCP connection is established or in the process of being established, the packet or group TCP segment sent from the sender side expects an ACK from the receiver to confirm reception of the packet. If the ACK is not received after a certain period, the sender decides to retransmit the packet. This section sets out to identify the reason for the retransmission.

$$\boxed{\text{Throughput (Byte/s)} = \text{Window Size} / \text{RTT}}$$

The formula above shows that as the time delay increases, the throughput decreases. As the window size increases, more transmission is enabled, thereby increasing throughput. Some networks, such as old cellular networks and long-distance communication lines, experience very high delay rates during operation. The previous sections (3.7.5) explained several methods for improving the application's throughput. These techniques include using the TCP window sliding size, reno, cubic, Vegas for congestion control and congestion avoidance and some applications that use multiple connections per application.

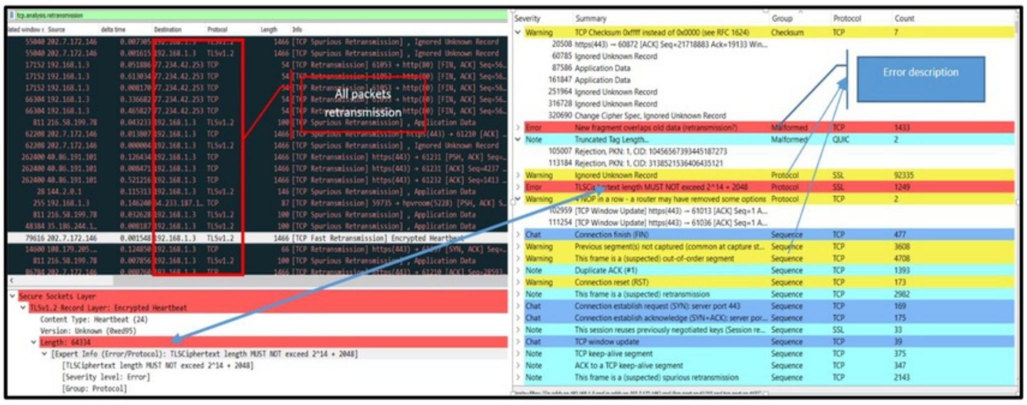

Wireshark IO Graphs: 60mnYoutube1_00001_20171003192119
All packet rate plot
Packets/1 min
Time (s)
Hover over the graph for details.
Wireshark IO Graphs: 60mnYoutube1_00001_20171003192119
Bar illustrate error rate
Green line illustrates retransmission plot
Packets/1 min
Time (s)
Hover over the graph for details.

Name	Display filter	Colo	Style	Y Axis	Y Field	Smoothing
☑ All packets		🟦	Line	Packets		None

Name	Display filter	Colo	Style	Y Axis	Y Field	Smoothing
☐ All packets		🟦	Line	Packets		None
☑ TCP errors	tcp.analysis.flags	🟥	Bar	Packets		None
☑ All packets	tcp.analysis.retransmission	🟩	Line	Packets		None

Figure 3.24 Packet error and retransmission plot

The more the network slows down, the higher the rate of retransmission. Figure 3.24 shows on the left the *YouTube* video streaming packet transmission plot expressed in packets per minute. The graph on the right-hand side graph shows the error rate and the retransmission plot (green). Of the 330527 packets, almost 2982 (1%) were retransmitted. Figure 3.25 shows that during an interval of 30 minutes, an average of 8000 packets per minute were exchanged between the video server (destination port 433), and the client (destination port 61231). On average, 400 erroneous (= with errors) packets per minute were re-transmitted, followed by an average of 150 packets per minute. Figure 3.25 presents a screenshot of the retransmitted communication between the server (destination port 433(HTTPS)) and the causes of the retransmission.



```
tcp.analysis.retransmission
lated window s Source          delta time Destination    Protocol  Length Info
  55040 202.7.172.146    0.007305 192.168.1.3    TLSv1.2   1466 [TCP Spurious Retransmission] , Ignored Unknown Record
  55040 202.7.172.146    0.001615 192.168.1.3    TLSv1.2   1466 [TCP Spurious Retransmission] , Ignored Unknown Record
  17152 192.168.1.3      0.051886 77.234.42.253  TCP         54 [TCP Retransmission] 61053 → http(80) [FIN, ACK] Seq=56…
  17152 192.168.1.3      0.613034 77.234.42.253  TCP         54 [TCP Retransmission] 61053 → http(80) [FIN, ACK] Seq=56…
  17152 192.168.1.3      0.008170 77.234.42.253  TCP         54 [TCP Retransmission] 61053 → http(80) [FIN, ACK] Seq=56…
  66304 192.168.1.3      0.336682 77.234.42.253  TCP         54 [TCP Retransmission] 61053 → http(80) [FIN, ACK] Seq=56…
  66304 192.168.1.3      0.465827 77.234.42.253  TCP         54 [TCP Retransmission] 61053 → http(80) [FIN, ACK] Seq=56…
    811 216.58.199.78    0.043233 192.168.1.3    TLSv1.2    100 [TCP Spurious Retransmission] , Application Data
  62208 202.7.172.146    0.013807 192.168.1.3    TCP       1466 [TCP Spurious Retransmission] https(443) → 61210 [ACK] …
  62208 202.7.172.146    0.000004 192.168.1.3    TLSv1.2   1466 [TCP Spurious Retransmission] , Ignored Unknown Record
 262400 40.86.191.101    0.126434 192.168.1.3    TCP       1466 [TCP Retransmission] https(443) → 61231 [PSH, ACK] Seq=…
 262400 40.86.191.101    0.008471 192.168.1.3    TCP       1466 [TCP Retransmission] https(443) → 61231 [ACK] Seq=4237 …
 262400 40.86.191.101    0.521216 192.168.1.3    TCP       1466 [TCP Retransmission] https(443) → 61232 [ACK] Seq=1413 …
     28 144.2.0.1        0.115313 192.168.1.3    TLSv1.2    146 [TCP Spurious Retransmission] , Application Data
    255 192.168.1.3      0.146240 54.233.187.1…  TCP         87 [TCP Retransmission] 59735 → hpvroom(5228) [PSH, ACK] S…
    811 216.58.199.78    0.032628 192.168.1.3    TLSv1.2    100 [TCP Spurious Retransmission] , Application Data
  48384 35.186.244.1…    0.008187 192.168.1.3    TLSv1.2    100 [TCP Spurious Retransmission] , Application Data
  79616 202.7.172.146    0.001548 192.168.1.3    TLSv1.2   1466 [TCP Fast Retransmission] Encrypted Heartbeat
  14600 108.179.205…     0.124850 192.168.1.3    TCP         66 [TCP Retransmission] https(443) → [illegible] [SYN, ACK] Seq=…
    811 216.58.199.78    0.007056 192.168.1.3    TLSv1.2    100 [TCP Spurious Retransmission] , Application Data
  86784 202.7.172.146    0.000768 192.168.1.3    TCP       1466 [TCP Retransmission] https(443) → 61210 [ACK] Seq=28593…

v Secure Sockets Layer
  v TLSv1.2 Record Layer: Encrypted Heartbeat
      Content Type: Heartbeat (24)
      Version: Unknown (0xed95)
    v Length: 64334
      v [Expert Info (Error/Protocol): TLSCiphertext length MUST NOT exceed 2^14 + 2048]
          [TLSCiphertext length MUST NOT exceed 2^14 + 2048]
          [Severity level: Error]
          [Group: Protocol]
```

```
Severity   Summary                                                  Group      Protocol  Count
v Warning  TCP Checksum 0xffff instead of 0x0000 (see RFC 1624)      Checksum   TCP       7
     20508 https(443) → 60872 [ACK] Seq=21718883 Ack=19133 Win…
     60785 Ignored Unknown Record
     87586 Application Data
    161847 Application Data
    251964 Ignored Unknown Record
    316728 Ignored Unknown Record
    320690 Change Cipher Spec, Ignored Unknown Record
> Error    New fragment overlaps old data (retransmission?)          Malformed  TCP       1433
v Note     Truncated Tag Length…                                     Malformed  QUIC      2
    105007 Rejection, PKN: 1, CID: 10456567393445187273
    113184 Rejection, PKN: 1, CID: 3138521536406435121
> Warning  Ignored Unknown Record                                    Protocol   SSL       92335
> Error    TLSCiphertext length MUST NOT exceed 2^14 + 2048          Protocol   SSL       1249
v Warning  4 NOP in a row - a router may have removed some options   Protocol   TCP       2
    102959 [TCP Window Update] https(443) → 61013 [ACK] Seq=1 A…
    111254 [TCP Window Update] https(443) → 61036 [ACK] Seq=1 A…
> Chat     Connection finish (FIN)                                   Sequence   TCP       477
> Warning  Previous segment(s) not captured (common at capture st…   Sequence   TCP       3608
> Warning  This frame is a (suspected) out-of-order segment          Sequence   TCP       4708
> Note     Duplicate ACK (#1)                                        Sequence   TCP       1393
> Warning  Connection reset (RST)                                    Sequence   TCP       173
> Note     This frame is a (suspected) retransmission                Sequence   TCP       2982
> Chat     Connection establish request (SYN): server port 443       Sequence   TCP       169
> Chat     Connection establish acknowledge (SYN+ACK): server por…   Sequence   TCP       175
> Note     This session reuses previously negotiated keys (Session re…  Sequence  SSL     33
> Chat     TCP window update                                         Sequence   TCP       39
> Note     TCP keep-alive segment                                    Sequence   TCP       375
> Note     ACK to a TCP keep-alive segment                           Sequence   TCP       347
> Note     This frame is a (suspected) spurious retransmission       Sequence   TCP       2143
```

Figure 3.25   Retransmission screenshot and expert information summary

The left-hand side of Figure 3.25 displays many retransmission packets from the receiver with the IP address 192.128.1.3, and the sender with IP address 77.254.42.253 and vice versa. It also provides an indication of the type of error and its location. The right-hand side of figure 3.25 presents a summary of the possible reasons that could have caused the retransmission. Possible explanations for the retransmission include checksum error, corrupted packet (malformed), protocol error such as Encryption or Authentication problem (TLSCiphertext length exceeds the recommended set value), and sequence problems such as an out of order segment, suspected frame or packet time- out when the time to live expires before the packet arrives at its destination.

1.6.13 TCP Observations

The outcome of these tests revealed some interesting findings related to the TCP. First, the TCP provides a reliable, connection-oriented transport service. Data involved in the TCP connection is sequenced and confirms that data arrived safely at its destination by sending an acknowledgement. Second, in TCP-based communication, automatic retransmission is provided for corrupted, duplicated, or lost TCP segments. Flow control techniques and congestion avoidance mechanisms are provided to reduce network or TCP host saturation.

The third observation that can be made is that communications in TCP connections begin with a three-way process [SYN, (SYN-ACK) and ACK] called a Handshake. The value of the sequence number field in the TCP header increases by the value of bytes enclosed in each individual segment during the data exchange. TCP uses a system where each end of the connection tracks its own sequence number as well as other end host numbers.

Retransmissions are activated by Duplicate Acknowledgments called Fast Recovery Mechanisms or a Retransmit Timeout (RTO) when lost segments occur. The figure indicates that three identical ACKs prompt an auto–retransmission of the packet. In TCP, a method known as Selective Acknowledgments automatically controls the transmission of packet loss. Window scaling is another technique used in TCP to manage the advertised receive buffer space above the 65,535byte limit. When a receiver announces the window size of zero to the sender, the buffer lacks the space to receive any packet from another host. Therefore, the transmission is interrupted or completely stopped. In *Wireshark*, an Expert feature also enables the detection of packet loss, windows variable condition, out-of-order packets and retransmission.

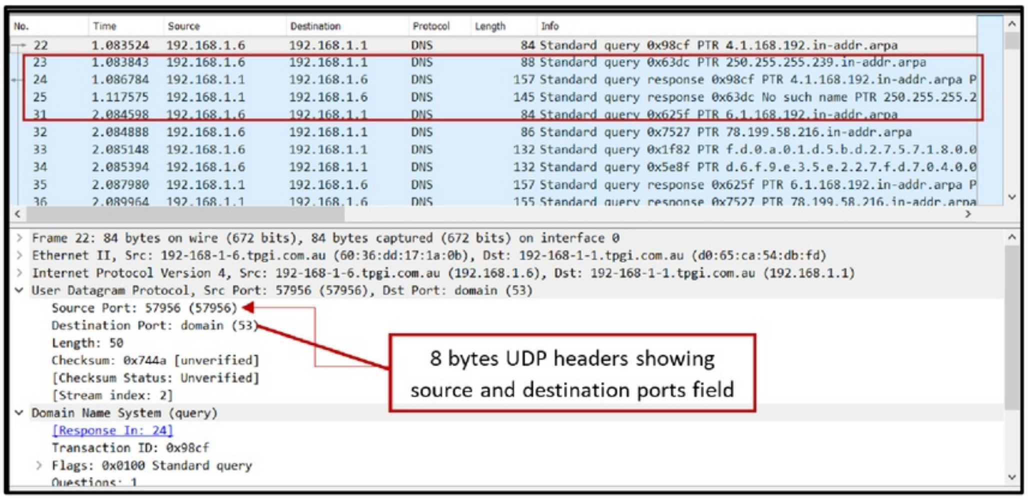
1.7 User Datagram Protocol (UDP) Traffic Analysis

1.7.1 Purpose of the User Datagram Protocol

UDP is a very simple transport layer protocol that provides connectionless service, multiplexing and protection functionality of the carried data. Since the UDP protocol does not guarantee end-to-end reliable service, this section explores the protocol's behaviour to observe its structure.

The UDP is composed of a simple header including four fields - source and destination ports, packet length and the checksum field. The receiver uses the checksum to decide whether the packet should be dropped or accepted. When the packet is dropped, the UDP has no mechanism to recover or retransmit the lost packet. As discussed earlier, the UDP provides a connectionless transport service usually to the following upper layer protocol: DNS, TFTP, RTP, DHCP, and video streaming application.

1.7.2 UDP- base DNS analysis



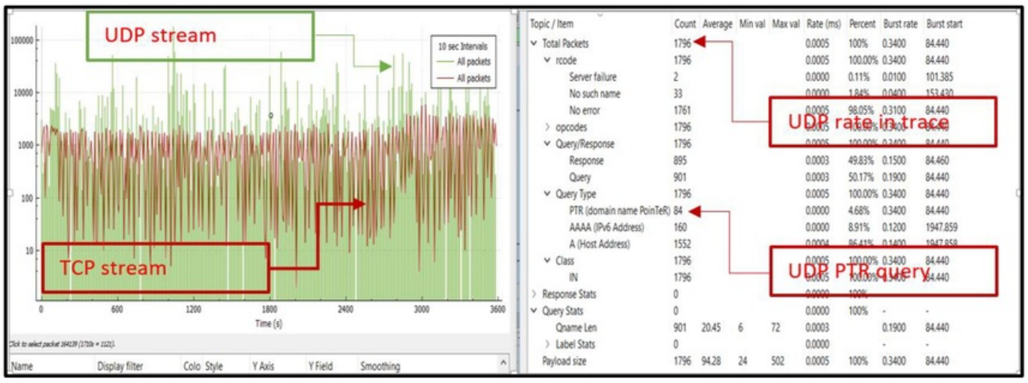
```
No.   Time      Source        Destination   Protocol Length Info
22    1.083524  192.168.1.6   192.168.1.1   DNS       84 Standard query 0x98cf PTR 4.1.168.192.in-addr.arpa
23    1.083843  192.168.1.6   192.168.1.1   DNS       88 Standard query 0x63dc PTR 250.255.255.239.in-addr.arpa
24    1.086784  192.168.1.1   192.168.1.6   DNS      157 Standard query response 0x98cf PTR 4.1.168.192.in-addr.arpa P
25    1.117575  192.168.1.1   192.168.1.6   DNS      145 Standard query response 0x63dc No such name PTR 250.255.255.2
31    2.084598  192.168.1.6   192.168.1.1   DNS       84 Standard query 0x625f PTR 6.1.168.192.in-addr.arpa
32    2.084888  192.168.1.6   192.168.1.1   DNS       86 Standard query 0x7527 PTR 78.199.58.216.in-addr.arpa
33    2.085148  192.168.1.6   192.168.1.1   DNS      132 Standard query 0x1f82 PTR f.d.0.a.0.1.d.5.b.d.2.7.5.7.1.8.0.0
34    2.085394  192.168.1.6   192.168.1.1   DNS      132 Standard query 0x5e8f PTR d.6.f.9.e.3.5.e.2.2.7.f.d.7.0.4.0.0
35    2.087980  192.168.1.1   192.168.1.6   DNS      157 Standard query response 0x625f PTR 6.1.168.192.in-addr.arpa P
36    2.089964  192.168.1.1   192.168.1.6   DNS      155 Standard query response 0x7527 PTR 78.199.58.216.in-addr.arpa

> Frame 22: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
> Ethernet II, Src: 192-168-1-6.tpgi.com.au (60:36:dd:17:1a:0b), Dst: 192-168-1-1.tpgi.com.au (d0:65:ca:54:db:fd)
> Internet Protocol Version 4, Src: 192-168-1-6.tpgi.com.au (192.168.1.6), Dst: 192-168-1-1.tpgi.com.au (192.168.1.1)
v User Datagram Protocol, Src Port: 57956 (57956), Dst Port: domain (53)
    Source Port: 57956 (57956)
    Destination Port: domain (53)
    Length: 50
    Checksum: 0x744a [unverified]
    [Checksum Status: Unverified]
    [Stream index: 2]
v Domain Name System (query)
    [Response In: 24]
    Transaction ID: 0x98cf
  > Flags: 0x0100 Standard query
    Questions: 1
```

8 bytes UDP headers showing source and destination ports field

Figure 3.26 UDP- based Domain Name System (DNS)

Figure 3.26 illustrates normal UDP- based communication such as DNS standard PTR record and respond. The query is sent from Packet number 22 with source port 57956 to the destination port Domain (53). Packet 24 with source address 192.168.1.1 (router) and the destination address 192.168.1.6 (client) illustrate the response to the query. As with TCP, the UDP source field is used to open a listening port for the packet response, and determines the protocol or application sending the packet. On the other hand, the destination field in the UDP header defines the destination process receiving the packet.

1.7.3 UDP Statistical Analysis

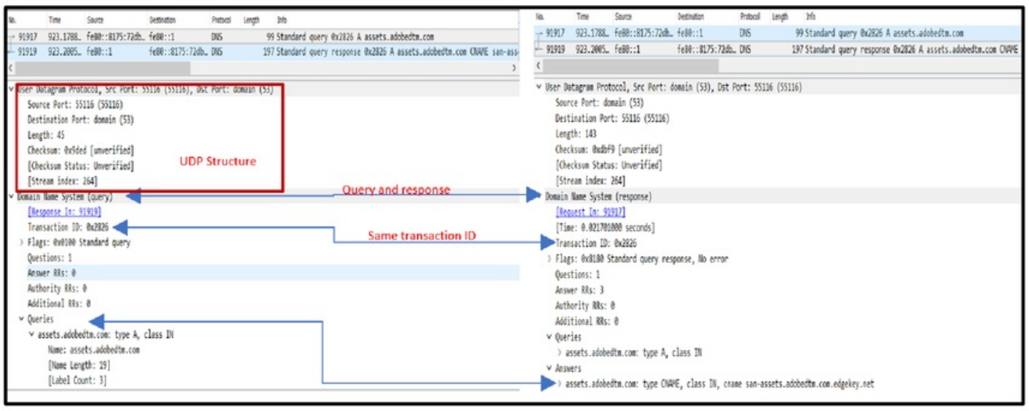


```
Topic / Item                   Count  Average  Min val  Max val  Rate (ms)  Percent  Burst rate  Burst start
v Total Packets                1796                              0.0005     100%     0.3400      84.440
  v rcode                      1796                              0.0005     100.00%  0.3400      84.440
      Server failure           2                                 0.0000     0.11%    0.0100      101.385
      No such name             33                                0.0000     1.84%    0.0400      153.430
      No error                 1761                              0.0005     98.05%   0.3100      84.440
  > opcodes                    1796                              0.0005     100.00%  0.3400      84.440
  v Query/Response             1796                              0.0005     100.00%  0.3400      84.440
      Response                 895                               0.0003     49.83%   0.1500      84.460
      Query                    901                               0.0003     50.17%   0.1900      84.440
  v Query Type                 1796                              0.0005     100.00%  0.3400      84.440
      PTR (domain name PoinTeR) 84                               0.0000     4.68%    0.3400      84.440
      AAAA (IPv6 Address)      160                               0.0000     8.91%    0.1200      1947.859
      A (Host Address)         1552                              0.0004     86.41%   0.1400      1947.858
  v Class                      1796                              0.0005     100.00%  0.3400      84.440
      IN                       1796                              0.0005     100.00%  0.3400      84.440
> Response Stats               0                                 0.0000     100%
v Query Stats                  0                                 0.0000     100%     -           -
    Qname Len                  901    20.45    6        72       0.0003              0.1900      84.440
  > Label Stats                0                                 0.0000              -           -
  Payload size                 1796   94.28    24       502      0.0005     100%     0.3400      84.440
```

Figure 3.27 UDP statistical analysis

Figure 3.27 illustrates the volume of UDP present in this communication. From zero to 600 seconds an average of 10000 user datagram packets were transmitted. Only 2000 TCP segments were exchanged during the communication. The right-hand side of the figure defines more precisely the exact number of UPD datagram (1796) packets, the number of PTR queries and responses and more detail about this network communication. A total of 901(50.17%) PTR queries were made followed by 45.83% responses.

1.7.4 UDP structure



```
No.     Time       Source              Destination         Protocol  Length  Info
91917   923.1788…  fe80::8175:72db…    fe80::1             DNS       99      Standard query 0x2826 A assets.adobedtm.com
91919   923.2005…  fe80::1             fe80::8175:72db…    DNS       197     Standard query response 0x2826 A assets.adobedtm.com CNAME san-ass

v User Datagram Protocol, Src Port: 55116 (55116), Dst Port: domain (53)
    Source Port: 55116 (55116)
    Destination Port: domain (53)
    Length: 45
    Checksum: 0x9ded [unverified]
    [Checksum Status: Unverified]
    [Stream index: 264]
v Domain Name System (query)
    [Response In: 91919]
    Transaction ID: 0x2826
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v assets.adobedtm.com: type A, class IN
        Name: assets.adobedtm.com
        [Name Length: 19]
        [Label Count: 3]
```

```
No.     Time       Source              Destination         Protocol  Length  Info
91917   923.1788…  fe80::8175:72db…    fe80::1             DNS       99      Standard query 0x2826 A assets.adobedtm.com
91919   923.2005…  fe80::1             fe80::8175:72db…    DNS       197     Standard query response 0x2826 A assets.adobedtm.com CNAME

v User Datagram Protocol, Src Port: domain (53), Dst Port: 55116 (55116)
    Source Port: domain (53)
    Destination Port: 55116 (55116)
    Length: 143
    Checksum: 0xdbf9 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 264]
  Domain Name System (response)
    [Request In: 91917]
    [Time: 0.021701000 seconds]
    Transaction ID: 0x2826
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 3
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > assets.adobedtm.com: type A, class IN
  v Answers
    > assets.adobedtm.com: type CNAME, class IN, cname san-assets.adobedtm.com.edgekey.net
```

Figure 3.28 UDP structure DNS-base

Figure 3.28 illustrates the detailed structures of the use of user datagram protocol header. The UDP header is formed of four fields as shown in the figure from top to bottom: source and destination port, address field and destination and field length (which is the length of the packet without data link padding). This field is not really needed in the whole communication process. The UDP length is based on the IP header payload length field. It can also be calculated by subtracting 20 bytes (IP header) to obtain the UDP length value.

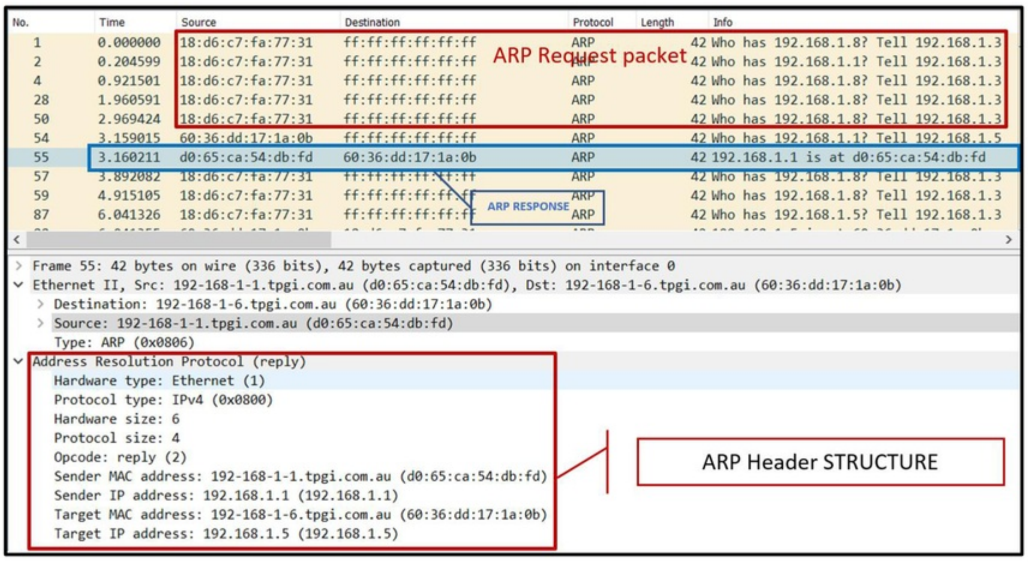
The checksum field provides an algorithm to perform sequence verification on the content of the UDP. In UDP-based communication, checksum is not always required. The example in figure 3.32 shows that the checksum field is unverified. The value 0X0000 informs the receiver of the non-validation of the checksum.

1.7.5 UDP Protocol Observations

Since the checksum field in the UDP header is optional, it cannot be used in some communications. The UDP is not able to recover lost packets because it is connectionless. Therefore, applications that use UDP at the transport layer must provide their own retransmission process. The port address field in the UDP header provides information about the application using the transport.

1.8 Network layer Protocol

1.8.1 Address resolution Protocol

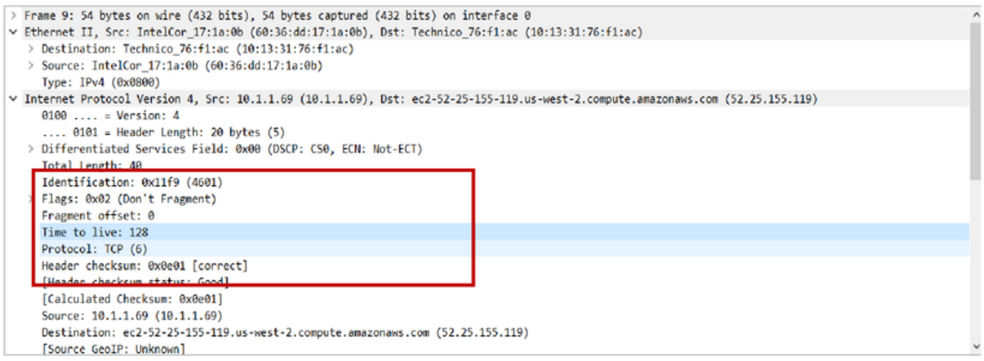


```
No.   Time       Source              Destination         Protocol  Length  Info
1     0.000000   18:d6:c7:fa:77:31   ff:ff:ff:ff:ff:ff   ARP       42 Who has 192.168.1.8? Tell 192.168.1.3
2     0.204599   18:d6:c7:fa:77:31   ff:ff:ff:ff:ff:ff   ARP       42 Who has 192.168.1.1? Tell 192.168.1.3
4     0.921501   18:d6:c7:fa:77:31   ff:ff:ff:ff:ff:ff   ARP       42 Who has 192.168.1.8? Tell 192.168.1.3
28    1.960591   18:d6:c7:fa:77:31   ff:ff:ff:ff:ff:ff   ARP       42 Who has 192.168.1.8? Tell 192.168.1.3
50    2.969424   18:d6:c7:fa:77:31   ff:ff:ff:ff:ff:ff   ARP       42 Who has 192.168.1.8? Tell 192.168.1.3
54    3.159015   60:36:dd:17:1a:0b   ff:ff:ff:ff:ff:ff   ARP       42 Who has 192.168.1.1? Tell 192.168.1.5
55    3.160211   d0:65:ca:54:db:fd   60:36:dd:17:1a:0b   ARP       42 192.168.1.1 is at d0:65:ca:54:db:fd
57    3.892082   18:d6:c7:fa:77:31   ff:ff:ff:ff:ff:ff   ARP       42 Who has 192.168.1.8? Tell 192.168.1.3
59    4.915105   18:d6:c7:fa:77:31   ff:ff:ff:ff:ff:ff   ARP       42 Who has 192.168.1.8? Tell 192.168.1.3
87    6.041326   18:d6:c7:fa:77:31   ff:ff:ff:ff:ff:ff   ARP       42 Who has 192.168.1.5? Tell 192.168.1.3

> Frame 55: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
v Ethernet II, Src: 192-168-1-1.tpgi.com.au (d0:65:ca:54:db:fd), Dst: 192-168-1-6.tpgi.com.au (60:36:dd:17:1a:0b)
  > Destination: 192-168-1-6.tpgi.com.au (60:36:dd:17:1a:0b)
  > Source: 192-168-1-1.tpgi.com.au (d0:65:ca:54:db:fd)
    Type: ARP (0x0806)
v Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: 192-168-1-1.tpgi.com.au (d0:65:ca:54:db:fd)
    Sender IP address: 192.168.1.1 (192.168.1.1)
    Target MAC address: 192-168-1-6.tpgi.com.au (60:36:dd:17:1a:0b)
    Target IP address: 192.168.1.5 (192.168.1.5)
```

Figure 3.29 ARP structure

In general, the Address Resolution Protocol (ARP) provides the LAN hosts' address resolution. These local hosts can be considered as the final end-point of the communication or as a local switching device such as a router. Before creating an ARP request, the local ARP cache must be inspected. Then using the same format, both the ARP request and the response are sent through the network. Figure 3.29 shows a typical ARP request, where the broadcast address (ff:ff:ff:ff:ff:ff) address are broadcast including the question (Who has this address?) within the local network. In Packet number 55, the ARP reply is sent directly to the requested host address (60:36:dd:17:1a:0b). Figure 3.29 also presents the ARP header component. This indicates that the ARP does not contain an IP header in its structure; therefore, the ARP packets are not routable. In network troubleshooting, the ARP helps detect IP address duplication in the local network.

1.8.2 Internet Protocol



```
> Frame 9: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
v Ethernet II, Src: IntelCor_17:1a:0b (60:36:dd:17:1a:0b), Dst: Technico_76:f1:ac (10:13:31:76:f1:ac)
  > Destination: Technico_76:f1:ac (10:13:31:76:f1:ac)
  > Source: IntelCor_17:1a:0b (60:36:dd:17:1a:0b)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 10.1.1.69 (10.1.1.69), Dst: ec2-52-25-155-119.us-west-2.compute.amazonaws.com (52.25.155.119)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x11f9 (4601)
  > Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x0e01 [correct]
    [Header checksum status: Good]
    [Calculated Checksum: 0x0e01]
    Source: 10.1.1.69 (10.1.1.69)
    Destination: ec2-52-25-155-119.us-west-2.compute.amazonaws.com (52.25.155.119)
    [Source GeoIP: Unknown]
```

Figure 3.30 Time to live variation

Internet Protocol version 4 provides connectionless routing service at the network layer and IP protocol switch packet moving across the network. It also uses techniques to fragment packets to the required MTU size before switching them through the network. IP also guarantees the application of the Quality of service (QoS) information in the packet. Examination of the IP header structure in figure 3.30 indicates the reduction of the Time to Live (TTL) field value of each packet forwarded by the router. This explains the number of hops the packet travels before reaching its destination. The figure also shows how IP packets behave while travelling through an internet network. The IP ID value field remains the same for each packet from the same fragment. Some packets do not allow any IP fragmentation at any point in the network as shown in figure 3.30 (packet number). If the network layer receives a small packet, it can be transmitted. But large packets cannot be sent through because the MTU requirement is not met; consequently, the packet is simply dropped. In this case, the Internet Control Message Protocol (ICMP) provides an error message service for any problem at the network layer. The response is then sent back to indicate the reason for the packet drop.

4 Results

As demonstrated in the previous section, *Wireshark* incorporates numerous functions that allow analysts to investigate network problems. Network problems are typically caused by device failures, poor network configuration and internal and external attacks are all possible causes of network problems. *Wireshark* is effective at investigating all such problems.

The primary action in mitigating network problems consists of a good understanding of packet traces in those areas that experience performance issues. Network administrators must be aware of the importance of Wireshark and new algorithms because of their effectiveness at finding causes and solving problems that are time-consuming to discover.

This thesis has described several methods of analysing packet traces using *Wireshark* and presented examples of common local network attacks. It has also reported mitigation measures and methods of reducing the impact that these create on network performance. Filters and extra functions such as *Follow TCP Stream* and *Export Info* are also provided by Wireshark through which more rigorous analysis of traffic can be performed. The use of graphs provides an efficient way of interpreting packet traces and detecting attacks.

With the help of Wireshark's Time-Sequence Graph (figure 3.21), TCP Round Trip Time Graph (figure 3.7), and the Throughput Graph from TCP data flow (fig. 3.22), it was possible isolate the reasons for the network performance problems, and derive the key performance indicators of the network. The Advanced IO Graphs solved the mathematic part of the problem by providing computing network values of such variables as SUM (variable), COUNT (variable), MIN (variable), and MAX (variable) on the packets. Display filters also played a major role in analysing the traffic in the Advanced IO Graphs. The Round-Trip Time Graph, as illustrated in figure 3.7, defined the time between data being transmitted and the other end of the communication. TCP ACK Throughput Graphs, as illustrated in figure 3.22, are one way of plotting the entire bytes seen in the trace at a precise time. If the throughput value diminishes, the time it takes to transfer data increases. The TCP Time-Sequence Graphs plot each TCP packet based on the variation of sequence numbers over time. In addition, this graph type depicts the ACKs seen and the window size.

5 Discussion

Overview

The network traffic was captured with the use of *Wireshark* – a network monitoring tool. It first captured the packets from the home network. These packets were then analysed with the help of the display filters and statistics included in *Wireshark* (Shimonski, 2013). The analysis of TCP, UDP, IP and HTTP provided details of the network traffic. These details included information about packet loss, packet transmission, round trip time, time to live and retransmission. The HTTP analysis provided the protocol hierarchy for the captured packets by calculating the packet counter and load distribution for the captured packets. The TCP and HTTP analysis were used to identify the best performance of the network compared to the other analyses. The TCP analysis provided overall network traffic information using the IO graph and other features of *Wireshark* to resolve packet loss, packet retransmission, congestion control and congestion avoidance.

TCP protocol data flow and congestion avoidance

During data drop, the analysis of the two different congestion avoidance algorithms -Cubic and Reno - demonstrated considerable similarity in performance drop. A significant drop was observed for both congestion avoidance algorithms.

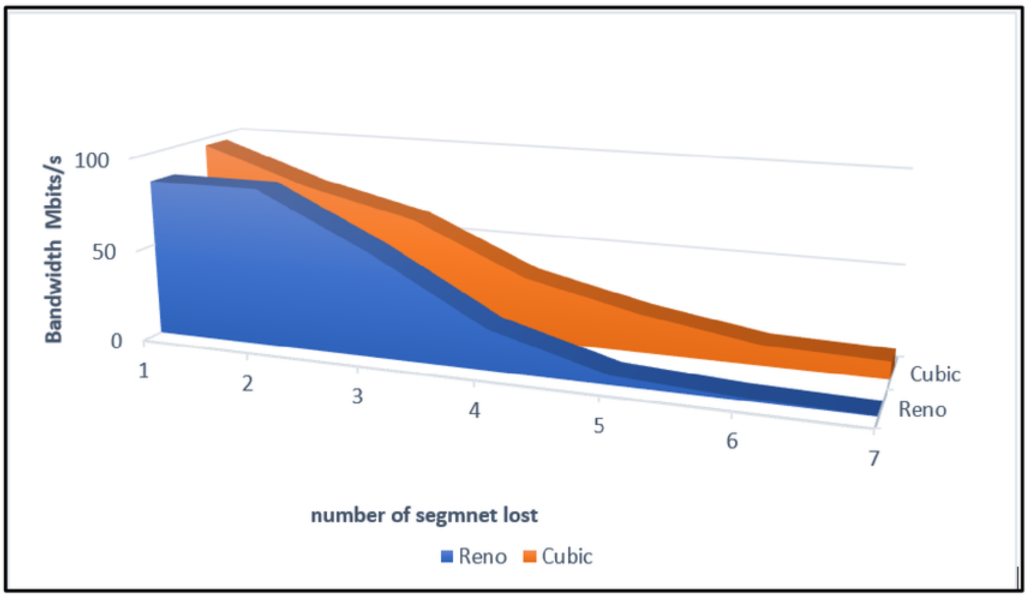


Figure 5.1 Performance of two different congestion avoidance algorithms while experiencing data loss

As figure 5.1 shows, for 1% packet drop, the bandwidth rates were 9.18 and 9.19 for Reno and Cubic respectively. For 5% data drop these bandwidths decreased to 5.86 and 7.07. When data loss was increased to 10%, the bandwidth rates declined to 1.93 and 2.34 respectively.

Outcome from the experiments when link experienced ACK loss

During the experiments, when the link was experiencing ACK drop, performance was not affected by a small number of acknowledgement drops. In the case of data drops, even for 1% packet drop, a decrease in performance was noticeable. While the ACK drop occurred, performance was unchanged for the link and remained constant at 9.37 Mbits/s, even when the link experienced 10% of ACK loss.

To observe the performance when the link was experiencing a high volume of ACK drops, the quantity of ACK drops was increased 10 times. Instead of 1%, 2%, 3%... 10% ACK drop rate, the performance for 10%, 20%, 30%....100% ACK drop was observed. In these experiments, a temporary performance drop was noticed after a 40% increase in the rate of acknowledgement drops. Congestion occurred in the network when the ACK drop rate reached 70%. The same experiments were repeated several times to ensure these results. Each time, a reduction in performance was observed after the ACK drop rate reached 40%; communication stalled completely at 60% acknowledgement drop.

The design of the congestion avoidance methods probably explains the remarkable performance drop during heavy packet loss. Since the transmission rate reduced to 50% for each lost packet, the transmission rate is increased linearly. But after the next packet loss, the transmission rate again decreases to 50% of the last value. Consequently, there will be an imbalance between the increase (linear) and decrease (halved) in transmission rate which will result in rapidly reducing performance. This may be one of the reasons behind the reduced performance in the TCP.

When the fast-retransmit mechanism signals congestion, the sender, instead of returning to Slow Start uses the Multiplicative Decrease Congestion Avoidance (MDCA) software, which is part of the fast recovery mechanism. During this drop, if the congestion window is less than 10 packets or the congestion window is within two packets of the receiver's advertised window, the Fast Recovery mechanism will be initiated. When three packets are dropped in a single window of data and the number of packets between the first and second dropped packets is less than 2+3W/4 (W is the congestion window just before the Fast Retransmit), the sender will wait for a retransmit timeout. When four packets are dropped in a single window, the sender will have to wait for a retransmit timeout. With the increased number of dropped packets in the same window, the likelihood of having to wait until retransmission occurs increases, thereby disrupting data transmission and halting transmission completely at a certain point. When the fast retransmission mechanism is disrupted, the fast recovery mechanism will not be able to perform properly [34].

When filtering the number of packets lost, the bandwidth dropped randomly. So, there is a high possibility that more than one packet has been dropped in the same window. This indicates an inefficient use of the fast retransmission and fast recovery mechanism which may explain the significant reduction.

The figure 5.2 illustrate the volume in packet per second of a home network traffic. the graph in figure 5.2 provides a pattern that is similar to the majority of the plot obtained in this project. The plot in figure 5.3 illustrates the predictable two-hour prediction that repeat every half an hour.

During the daily traffic study, the plot indicates that the traffic double in volume in two directions of the connection, from 5 am to 11AM during working day. Daily and weekly pattern are very similar. The peak is observing round midday. With 20% increase in packets, 30% in data flow. figure 5.3 indicated that, the download direction continually carries very high rate than the upload direction.

Figure 5.6 illustrates two statistics flow traffic related. The plot indicated that during peak time the peak state ca go up to 245,000 bytes per second.

The figure 5.5 shows the different element that form the traffic over 1 day in a home network the graph reveal line representing the traffic contain in term of IP base protocol and the breakdown of both UDP and TCP application. The observation show that TCP is the dominant protocol in the communication. Over the period of one day with an average of 94% of packet and 70% flow on the connection, UDP arrives in second position with roughly 8% of packets. Other protocol such as ICMP and ARP

Occupied the third position with an average of 1.5% packets.

Figure 5.7 reveal the composition of the common application measured I the home network for a period of one week. For each application, server and client are combine as one application. the graph shows that web base application comes first on the link with almost 65% of packets. Followed by the files transfer application with the average of 4% packets, SMTP comes third with with roughly 2 % of packet , finally telnet with less than 1% of the packet.

6 Conclusion

Time-sequence appears to be an important tool for trace analysis. Exploring the time at which different processes occur in the network enables the analyst to investigate what happens in the lifetime of the connection. This opens up new questions about the performance of protocols used today such as TCP, IP, UDP and others forming the TCP/IP stack.

To understand the reason behind TCP's drastically reduced performance during increasing packet loss, different congestion avoidance algorithms together with different types of TCP variables were analyzed. A decrease in transmission rate with a single packet loss was observed. This increase in packet loss led to a significant reduction in the performance of the network.

In a small network, the behaviour of TCP Reno was analysed by introducing packet loss. TCP Reno performed well when no packet loss was introduced, but by introducing 1% packet loss, the performance reduced from 9.37 Mbps to 9.18 Mbps which represents a 2% decrease in performance. When reaching 10% packet loss, performance of TCP Reno reduced to 1.93 Mbps.

A similar type of experiment was done while introducing external ACK loss. A small amount of ACK loss did not affect performance at all; it remained constant at 9.37 Mbps. But a large amount of ACK loss, for example 40% ACK loss, caused performance to drop from 9.13 Mbps to 8.18 Mbps and with a 60% reduction in ACK, performance decreased to 0.00014 Mbps (Megabyte per second). When the two congestion avoidance algorithms were compared, it was clear that Cubic performed better than Reno. In Cubic, the performance was constant at 93.7 Mbps until 60% of ACK drop. With Reno, performance decreased to 17.9 Mbps while the link was experiencing a 30% ACK drop and for 60% ACK loss, performance declined to 0%, as with Cubic.

ACK loss is not as serious as loss of data packets since they are cumulative, meaning that a missing ACK will be recovered when the next ACK is delivered. However, when excessive ACK loss occurs, the sender has to retransmit the packets. In this situation, it is likely that a high ACK loss rate will rapidly increase the number of retransmitted packets which will once again be lost and retransmitted. This will create a huge load on the network which is likely to result in a significant performance drop.

In conclusion, the experimental results demonstrate that a small amount of ACK loss does not affect network performance, while a large amount of ACK loss was unable to be handled by the TCP. This report demonstrates the existence of bugs that are concealed by the robustness of the TCP/IP. This problem can be solved in two phases. First the Reset (RST) must be found, and then the problem identified and fixed. Research is currently being carried out which aims to increase the reliability of TCP. Therefore, it is likely that performance problems involving TCP in existing networks will be ameliorated by the next generation of advanced algorithms.

Future research

Wireshark, as a network protocol analyser, performs better than other network monitoring tools. In the future, Wireshark will continue to be used for intrusion detection. It provides efficient detection of malicious packets in the network. The future development of Wireshark is likely to focus on enhancing its characteristics as a robust intrusion detection system. Because it has the most powerful features, it is the most popular network protocol. Accordingly, it is used for analysis, trouble shooting and communications protocol development, principally by capturing, viewing and analyzing the captured packets. As a network monitoring tool, it provides high quality performance and reduces intermittent connectivity.

Two major recommendations can be made for continuing to develop efficient network monitoring tools. First, by using more powerful processors and hardware configurations, the data processing speed could be increased, which may reduce the amount of packet loss. This should also have a positive impact on performance. However, the potential impact of other TCP variables may also influence the bandwidth rate. Second, research should focus on observing the behaviour of other Congestion Avoidance Algorithms during multiple packet loss to increase efficiency in the network. While the TCP may survive a single packet loss, the challenge remains to develop a protocol that can cope with multiple packet losses.

Appendix 1 More figures

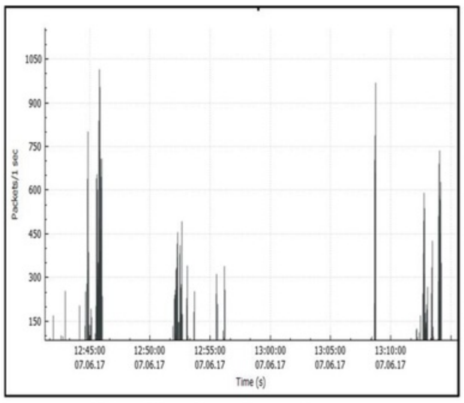


Fig 5.2 Packet volume for 1 day on domestic network

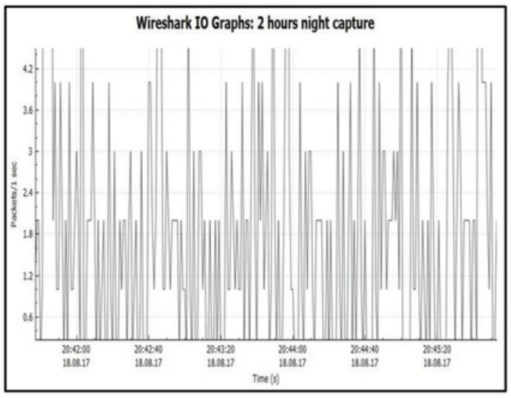


Fig 5.3 Packet volume for 2 hours on domestic network

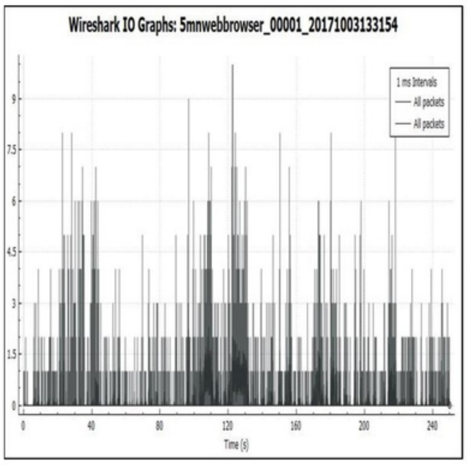


Fig 5.4 Packet volume for 1 days on domestic network combine with UDP rate

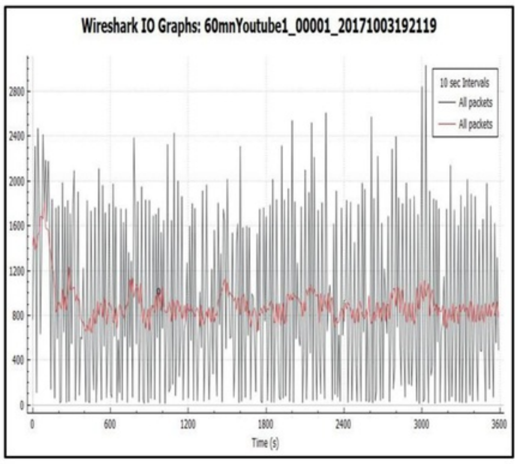


Fig 5.5 Packet volume for 1 days on domestic network combine with TCP rate

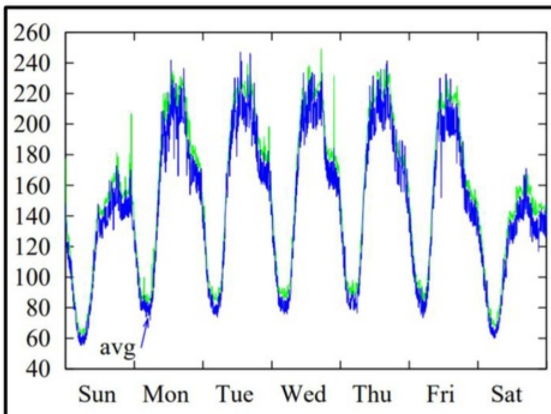
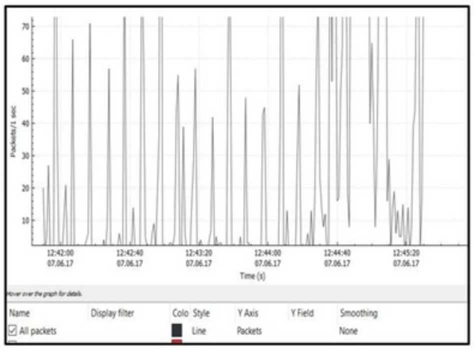
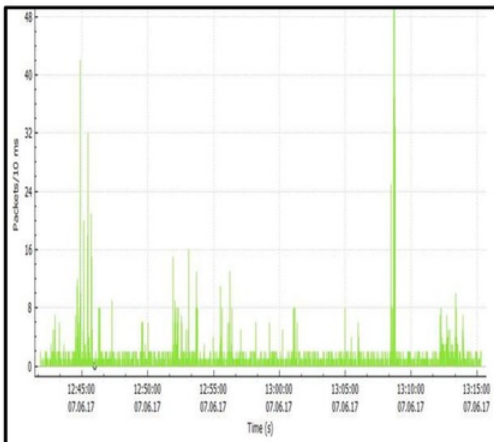


Fig 5.6 flow volume for 1 week on home network



Name	Display filter	Colo	Style	Y Axis	Y Field	Smoothing
☑ All packets		■	Line	Packets		None

Fig 5.7 Average packet size 1 week on home link



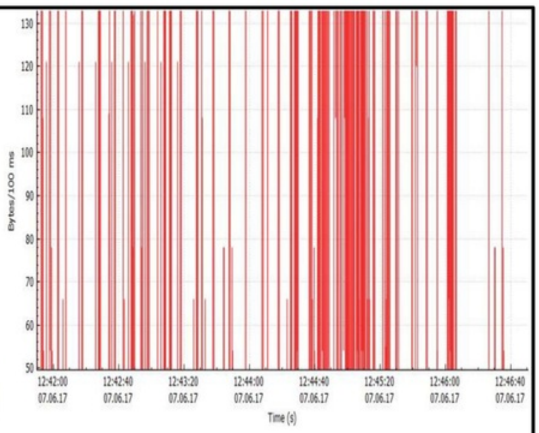


Fig 5.8 average UPD volume in the home link

Fig 5.9 average TCP volume in the home link

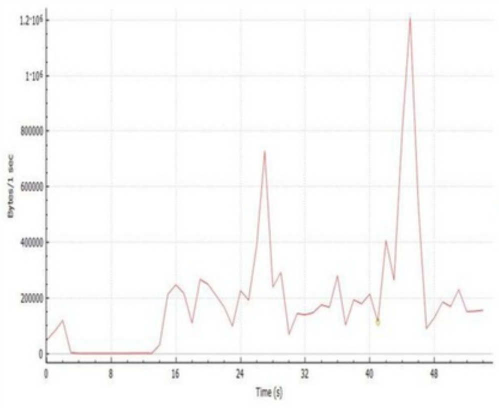


Fig 5.10 Average traffic per TCP 1 hour home network

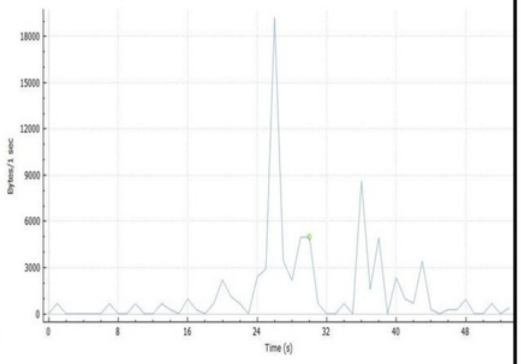
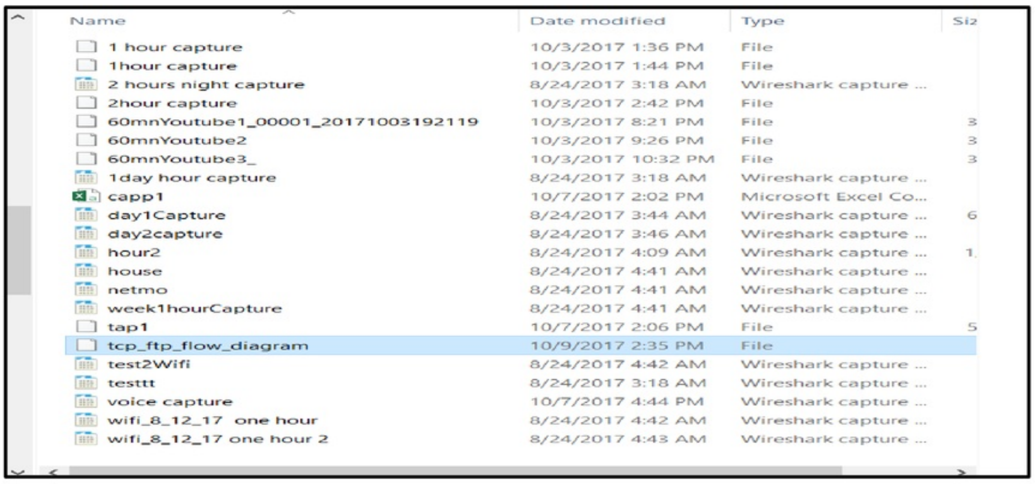


Fig 5.11 Average traffic per TCP size week home network



Name	Date modified	Type	Siz
1 hour capture	10/3/2017 1:36 PM	File	
1hour capture	10/3/2017 1:44 PM	File	
2 hours night capture	8/24/2017 3:18 AM	Wireshark capture ...	
2hour capture	10/3/2017 2:42 PM	File	
60mnYoutube1_00001_20171003192119	10/3/2017 8:21 PM	File	3
60mnYoutube2	10/3/2017 9:26 PM	File	3
60mnYoutube3_	10/3/2017 10:32 PM	File	3
1day hour capture	8/24/2017 3:18 AM	Wireshark capture ...	
capp1	10/7/2017 2:02 PM	Microsoft Excel Co...	
day1Capture	8/24/2017 3:44 AM	Wireshark capture ...	6
day2capture	8/24/2017 3:46 AM	Wireshark capture ...	
hour2	8/24/2017 4:09 AM	Wireshark capture ...	1.
house	8/24/2017 4:41 AM	Wireshark capture ...	
netmo	8/24/2017 4:41 AM	Wireshark capture ...	
week1hourCapture	8/24/2017 4:41 AM	Wireshark capture ...	
tap1	10/7/2017 2:06 PM	File	5
tcp_ftp_flow_diagram	10/9/2017 2:35 PM	File	
test2Wifi	8/24/2017 4:42 AM	Wireshark capture ...	
testtt	8/24/2017 3:18 AM	Wireshark capture ...	
voice capture	10/7/2017 4:44 PM	Wireshark capture ...	
wifi_8_12_17 one hour	8/24/2017 4:42 AM	Wireshark capture ...	
wifi_8_12_17 one hour 2	8/24/2017 4:43 AM	Wireshark capture ...	

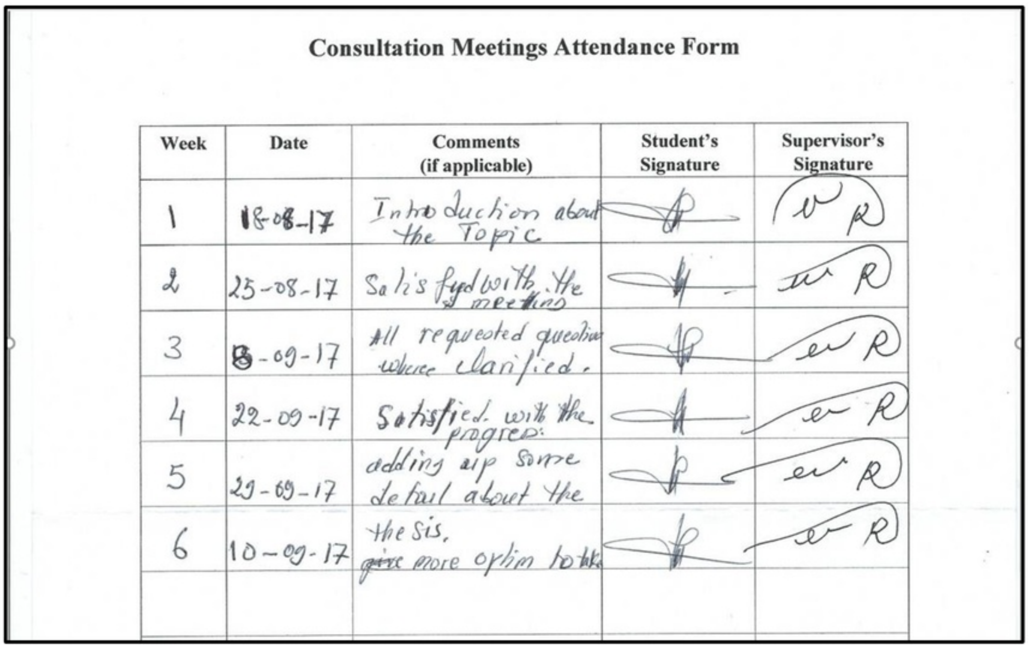
Figure 5.12 working folder

Appendix 2 Abbreviations and Definitions

Abbreviation	Definition
TCP	Transmission control protocol
IP	Internet protocol
IPV4	Internet protocol version 4
IPV6	Internet protocol version 6
UDP	User datagram protocol
IGMP	Internet Group Management Protocol
ACK	Acknowledgement
SYN	Synchronization
SEQ	Sequence
HTTP	Hypertext transport protocol
FTP	File transfer protocol
ARPA	Advance Research project agency
RFC	Request for comment
QoS	Quality of service
CoC	Control and comment
LAN	Local Area network

WAN	Wide area network
MCTCP	Multi-connection TCP
ISO	International Organization for Standardization
IAB	Internet architecture board
KPI	Key performance indicator
ITU-R	International telecommunication union
QPSK	Quadrature phase shift keying
QAM	Quadrature amplitude modulation
AM	Amplitude Modulation
NIC	Network interface card
RAM	Random access memory
POP	Post office protocol
SMTP	Single mail transport protocol
ECM	Error correction mechanism
MTU	Maximum Transfer Unit
ISN	Initial Sequence Number

Appendix 3 Details of Consultation Meetings



Consultation Meetings Attendance Form

Week	Date	Comments (if applicable)	Student's Signature	Supervisor's Signature
1	18-08-17	Introduction about the Topic	[illegible]	[illegible]
2	25-08-17	Satisfyd with the meeting	[illegible]	[illegible]
3	8-09-17	All requested question where clarified.	[illegible]	[illegible]
4	22-09-17	Satisfied with the progress	[illegible]	[illegible]
5	29-09-17	adding aip some detail about the	[illegible]	[illegible]
6	10-09-17	thesis. ~~give~~ more optim to the	[illegible]	[illegible]

References

[1]. H. Balakrishnan Seshan, and Sr, Elan, Amir, and Katz, R.H. (1995). "Improving TCP/IP performance over wireless networks". In MobiCom '95 Proceedings of the 1st annual international conference on Mobile computing and networking (pp. 2-11).

[2].E. Postel, "Internet Protocol, DARPA Internet Program Protocol Specification". Marina Del Ray, (1981)

[3]. J. Postel, "DOD Standard Internet Protocol". Defense Advanced Research Projects Agency, Information Processing Techniques Office, (1980). RFC 760, IEN 128.

[4]. K. Claffy and D. Clark, "Adding Enhanced Services to the Internet": Lessons from History. 43rd Telecommunications Policy Research Conference (TPRC). (2015), La Jolla, CA, US, Center for Applied Internet Data Analysis.

[5]. P. Karrberg. "The Emergence of the Mobile Internet in Japan and the UK": 2011 Platforms, Exchange Models and Innovation 1999-2011

[6]. Z. Lukszo and Gerard P. J. "The Operation and Evolution of Infrastructures": The Role of Agent-Based Modeling and Decision Making. International Journal Critical Infrastructures, 5, no. 4, 299-300.

[7]. V. Jacobson. "Congestion Avoidance and Control. Computer Communication Review" 1988 vol. 18, no. 4, pp. 314-329.

[8]. B. Braden, ed. (1989). "Requirements for Internet Hosts --Communication Layers". RFC 1122.

[9]. C. Barré, S. Pluntke, C. Greenhalgh, A. Wischik, D. and Handley, M. (2011). "Improving datacenter performance and robustness with multipath TCP In Proceedings of SIGCOMM" 2011, Toronto, Canada.

[10]. K. Holmberg (2008) Optimization models for routing in switching networks of clos type with many stages. AMO - Advanced Modeling and Optimization, 10(1).

[11]. M. Al-Fares, A. Louk issas, and A. Vahdat. A scalable, commodity data center network architecture. InProc. SIGCOMM 2010.

[12]. P. Geoffrey and Hoefler, T. (2008) "Adaptive routing strategies for modern high-performance networks". In Proceedings of the 2008 16th IEEE Symposium on High Performance Interconnects, pages 165–172, Washington, DC, USA.IEEE Computer Society.

[13]. B. Peirens, G. Detal, S. Barré, and O. Bonaventure, (2016). "Link bonding with transparent Multipath TCP", Internet-Draft, draft-peirens-mptcp-transparent-00.

[14]. C. Raiciu, C. Paasch, S. Barré, A. Ford and M. Handley, (2012). How Hard Can It Be? "Designing and Implementing a Deployable Multipath TCP". In USENIX Symposium of Networked Systems Design and Implementation (NSDI '12).

[15]. RFC 3439 - Some Internet Architectural Guidelines and Philosophy. ietf.org.

[16]. G. Walter The Illustrated Network: How TCP/IP Works in a Modern Network (PDF). Morgan Kaufmann. p. 26. ISBN 978-0123745415.

[17]. S. Andrew and Tanenbaum, (2003). Computer networks. Upper Saddle River, New Jersey: Prentice Hall. ISBN 0-13-066102-3.

[18]. A. Cecil "A Summary of Network Traffic Monitoring and Analysis Techniques".

[19]. J. Liu, F. Liu, and Ansari, N. (2014). "Monitoring and analyzing big traffic data of a large-scale cellular network with Hadoop". IEEE Network, 28(4), pp.32-39.

[20]. R. Shimonski, "The Wireshark field guide. Amsterdam" : Syngress (2013).

[21]. V. Tarasov and S. Malakhov (2015). "Statistical data handling program of Wireshark analyzer and incoming traffic research". Proceedings of the Institute for System Programming of the RAS, (3), pp.303-314.

[22]. W. Wang, X. Zhang, W. Shi, S. Lian, S. and Feng, D. (2012). "Understanding and analyzing network traffic". IEEE Network, 26(1), pp.4-5.

[23]. K. Xu, F. Wang, L. Gu, J. Gao, J and Y. Jin, Y. (2013). "Characterizing home network traffic: an inside view. Personal and Ubiquitous Computing", 18(4), pp.967-975.

[24]. G. Vinton Cerf, and Robert E Khan "A protocol for packet network intercommunication", IEEE transactions on communications

[25]. W. Goralski. "The Illustrated Network": How TCP/IP Works in a Modern Network (PDF). Morgan Kaufmann. p. 26. ISBN 978-0123745415.

[26]. D.Leith and R. Shorten, "H-TCP: TCP for high-speed and long-distance networks", http://www.hamilton.ie/net/htcp3.pdf[Acc. 11 Dec. 2011]

[27]. D.E. Comer, Internetworking with tcp/ip. 5th ed., New Jersy: Prentice Hall, 2006, pp.212-213.

[28]. K. Fall and Sally Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", ee.lbl.gov/papers/sacks.pdf [Acc. 5 Jun. 2013]

[29]. How to Use Wireshark to Capture, Filter and Inspect Packets. (2017). Howtogeek.com. Retrieved 1 September 2017, from https://www.howtogeek.com/104278/how-to-use-wireshark-to-capture-filter-and-inspect-packets

[30]. Honda, Osamu, Hiroyuki Ohsaki, Makoto Imase, Mika Ishizuka and Junichi Murayama, "Understanding TCP over TCP: Effects of TCP Tunneling on End-to-End

Throughput and Latency", http://www.ispl.jp/~oosaki/papers/Honda05_ITCom.pdf [Acc. Date September 19, 2011]

[31]. all About Wireshark, http://www.wireshark.org/about.html [Acc. 28 Oct. 2011]

[32]. Netcat for Windows, April 10, 2009 19:52, http://joncraton.org/blog/46 [Acc. 28 Oct. 2011

[33]. Xu, K., Wang, F., Gu, L., Gao, J. and Jin, Y. (2013). Characterizing home network traffic: an inside view. Personal and Ubiquitous Computing, 18(4), pp.967-975.

[34]. Tanenbaum, Andrew S. (2003). "Computer networks". Upper Saddle River, New Jersey: Prentice Hall. ISBN 0-13-066102-3.