Department of Electrical Engineering, UET Lahore
EE432 Computer Networks Lab

Course Instructor: Dr. Naveed Nawaz	**Dated: 23/09/2024**
Session: Fall 2024	**Semester: 7th**

LAB 5 Transport Layer Protocol: Transmission Control Protocol (TCP)

Name	Roll. No.	Report Marks (10)	Viva Marks (5)	Total Marks (15)
Ayesha Ahmad	2021-EE-052			

Signature: _____

Contents

List of Figures

Transfer Layer Protocol: Transmission Control Protocol (TCP)

5.1. Objectives

In this lab, we'll investigate the behavior of TCP in detail. We'll do so by analyzing a trace of the TCP segments sent and received in transferring a file from your computer to a remote server. We'll study TCP's use of sequence and acknowledgement numbers for providing reliable data transfer.

5.2. Instructions

1. Read carefully before starting the lab.
2. These exercises are to be done individually.
3. You are supposed to provide the answers to the questions listed at the end of this document and upload the completed report to your course's LMS site.
4. Avoid plagiarism by copying from the Internet or from your peers. You may refer to source/ text but you must paraphrase the original work. Your submitted work should be written by yourself.
5. Complete the lab half an hour before the lab ends.
6. At the end of the lab, a viva will be conducted to evaluate your understanding.

5.3. Background

5.3.1. The Transmission Control Protocol (TCP) Introduction

TCP provides connections between clients and servers. A TCP client establishes a connection with a given server, exchanges data with that server across the connection, and then terminates the connection.

TCP also provides reliability. When TCP sends data to the other end, it requires an acknowledgment in return. If an acknowledgment is not received, TCP automatically retransmits the data and waits a longer amount of time. After some number of retransmissions, TCP will give up, with the total amount of time spent trying to send data typically between 4 and 10 minutes (depending on the implementation).

TCP contains algorithms to estimate the round-trip time (RTT) between a client and server dynamically so that it knows how long to wait for an acknowledgment. For example, the RTT on a LAN can be milliseconds while across a WAN, it can be seconds. Furthermore, TCP continuously estimates the RTT of a given connection, because the RTT is affected by variations in the network traffic.

TCP also sequences the data by associating a sequence number with every byte that it sends. For example, assume an application writes 2,048 bytes to a TCP socket, causing TCP to send two segments, the first containing the data with sequence numbers 1–1,024 and the second containing the data with sequence numbers 1,025–2,048. (A segment is the unit of data that TCP passes to IP.) If the segments arrive out of order, the receiving TCP will reorder the two segments based on their sequence numbers before passing the data to the receiving application. If TCP receives duplicate data from its peer (say the peer thought a segment was lost and retransmitted it, when it wasn't really lost, the network was just overloaded), it can detect that the data has been duplicated (from the sequence numbers), and discard the duplicate data.

There is no reliability provided by UDP. UDP itself does not provide anything like acknowledgments, sequence numbers, RTT estimation, timeouts, or retransmissions. If a UDP datagram is duplicated in the network, two copies can be delivered to the receiving host. Also, if a UDP client sends two datagrams to the same destination, they can be reordered by the network and arrive out of order.

TCP provides flow control. TCP always tells its peer exactly how many bytes of data it is willing to accept from the peer at any one time. This is called the advertised window. At any time, the window is the amount of room currently available in the receive buffer, guaranteeing that the sender cannot overflow the receive buffer. The window changes dynamically over time: As data is received from the sender, the window size decreases, but as the receiving application reads data from the buffer, the window size increases. It is possible for the window to reach 0: when TCP's receive buffer for a socket is full and it must wait for the application to read data from the buffer before it can take any more data from the peer.

Finally, a TCP connection is full-duplex. This means that an application can send and receive data in both directions on a given connection at any time. This means that TCP must keep track of state information such as sequence numbers and window sizes for each direction of data flow: sending and receiving. After a full-duplex connection is established, it can be turned into a simplex connection if desired.

5.3.2. TCP Connection Establishment and Termination

5.3.2.1. Three-Way Handshake

Following scenario occurs when a TCP connection is established:

1. The server must be prepared to accept an incoming connection. This is normally done by calling socket, bind, and listen and is called a passive open.
2. The client issues an active open by calling connect. This causes the client TCP to send a "synchronize" (SYN) segment, which tells the server the client's initial sequence number for the data that the client will send on the connection. Normally, there is no data sent with the SYN; it just contains an IP header, a TCP header, and possible TCP options (which we will talk about shortly).
3. The server must acknowledge (ACK) the client's SYN and the server must also send its own SYN containing the initial sequence number for the data that the server will send on the connection. The server sends its SYN and the ACK of the client's SYN in a single segment.
4. The client must acknowledge the server's SYN.

The minimum number of packets required for this exchange is three; hence, this is called TCP's three-way handshake. We show the three segments in Figure 5.1.

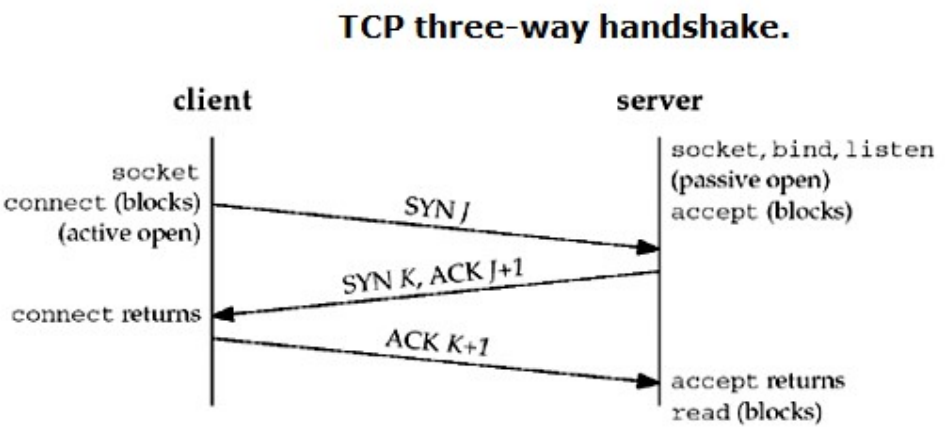


Figure 5.1: TCP three-way handshake.

We show the client's initial sequence number as J and the server's initial sequence number as K. The acknowledgment number in an ACK is the next expected sequence number for the end sending the ACK. Since a SYN occupies one byte of the sequence number space, the acknowledgment number in the ACK of each SYN is the initial sequence number plus one. Similarly, the ACK of each FIN is the sequence number of the FIN plus one.

An everyday analogy for establishing a TCP connection is the telephone system. The socket function is the equivalent of having a telephone to use. bind is telling other people your telephone number so that they can call you. listen is turning on the ringer so that you will hear when an incoming call arrives. connect requires that we know the other person's phone number and dial it. accept is when the person being called answers the phone. Having the client's identity returned by accept (where the identify is the client's IP address and port number) is similar to having the caller ID feature show the caller's phone number. One difference, however, is that accept returns the client's identity only after the connection has been established, whereas the caller ID feature shows the caller's phone number before we choose whether to answer the phone or not.

5.3.2.2. TCP Connection Termination

While it takes three segments to establish a connection, it takes four to terminate a connection:

1. One application calls close first, and we say that this end performs the active close. This end's TCP sends a FIN segment, which means it is finished sending data.
2. The other end that receives the FIN performs the passive close. The received FIN is acknowledged by TCP. The receipt of the FIN is also passed to the application as an end-of-file (after any data that may have already been queued for the application to receive), since the receipt of the FIN means the application will not receive any additional data on the connection.
3. Sometime later, the application that received the end-of-file will close its socket. This causes its TCP to send a FIN.
4. The TCP on the system that receives this final FIN (the end that did the active close) acknowledges the FIN.

Since a FIN and an ACK are required in each direction, four segments are normally required. We use the qualifier "normally" because in some scenarios, the FIN in Step 1 is sent with data. Also, the segments in Steps 2 and 3 are both from the end performing the passive close and could be combined into one segment. We show these packets in Figure 5.2.

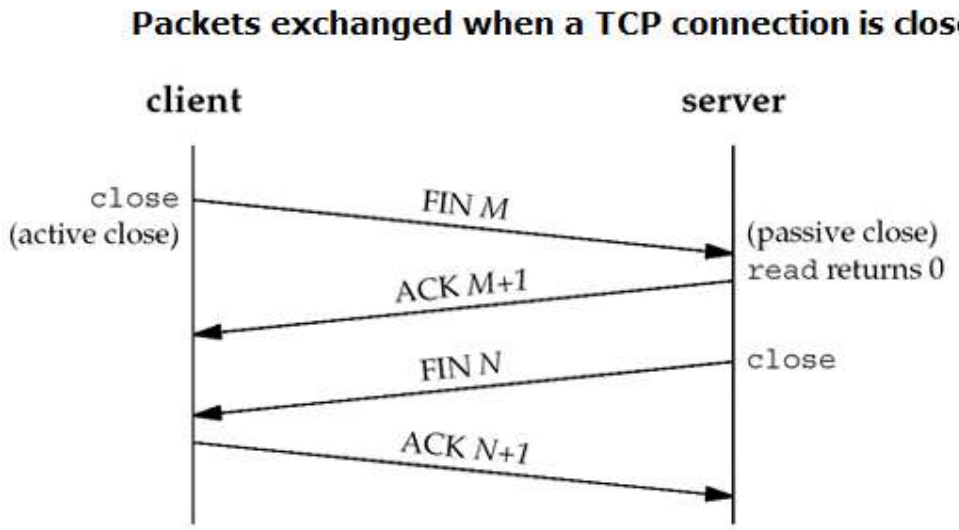


Figure 5.2: Packets exchanged when a TCP connection is closed.

A FIN occupies one byte of sequence number space just like a SYN. Therefore, the ACK of each FIN is the sequence number of the FIN plus one.

Between Steps 2 and 3 it is possible for data to flow from the end doing the passive close to the end doing the active close. This is called a half.

The sending of each FIN occurs when a socket is closed. We indicated that the application calls close for this to happen, but realize that when a Unix process terminates, either voluntarily (calling exit or having the main function return) or involuntarily (receiving a signal that terminates the process), all open descriptors are closed, which will also cause a FIN to be sent on any TCP connection that is still open.

Although we show the client in Figure 5.2 performing the active close, either end – the client or the server – can perform the active close. Often the client performs the active close, but with some protocols (notably HTTP), the server performs the active close.

5.4. Procedure

1. Capturing a bulk TCP transfer from your computer to a remote server: Before beginning our exploration of TCP, we'll need to use Wireshark to obtain a packet trace of the TCP transfer of a file from your computer to a remote server. You'll do so by accessing a Web page that will allow you to enter the name of a file stored on your computer and then transfer the file to a Web server using the HTTP POST method. We're using the POST method rather than the GET method as we'd like to transfer a large amount of data from your computer to another computer. Of course, we'll be running Wireshark during this time to obtain the trace of the TCP segments sent and received from your computer.
2. Start up your web browser. Go the http://gaia.cs.umass.edu/wireshark-labs/alice.txt and retrieve an ASCII copy of Alice in Wonderland. Store this file somewhere on your computer.
3. Next go to http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html. You should see a screen that looks like:

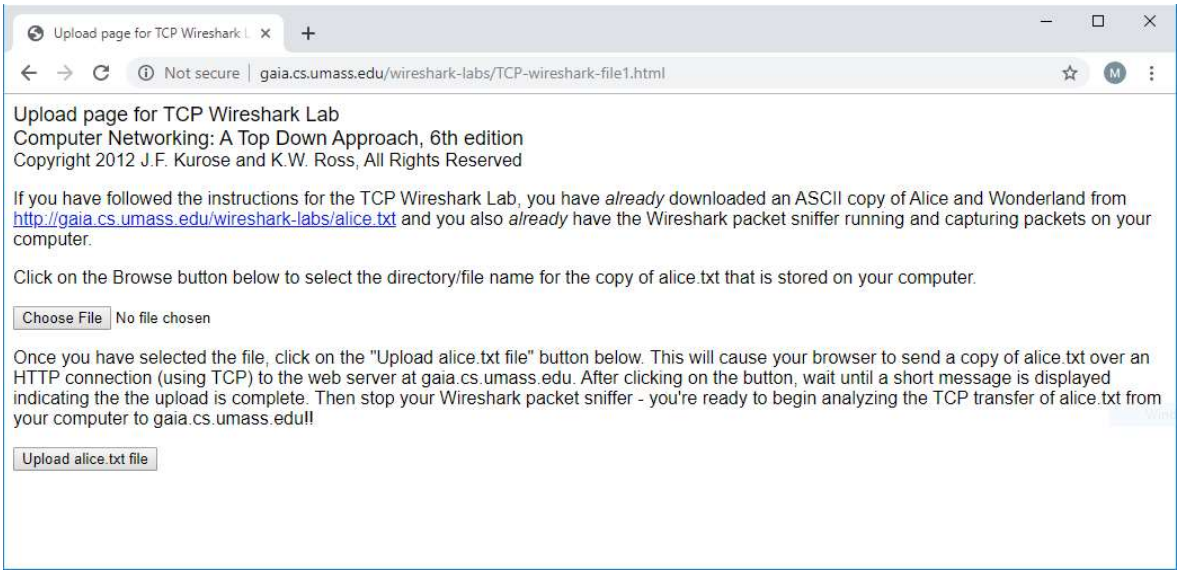


Figure 5.3: Screenshot of "http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html"

4. Use the Choose File button in this form to enter the name of the file (full path name) on your computer containing Alice in Wonderland (or do so manually). Don't yet press the "Upload alice.txt file" button.
5. Now start up Wireshark and begin packet capture (Capture->Start) and then press OK on the Wireshark Packet Capture Options screen.
6. Returning to your browser, press the "Upload alice.txt file" button to upload the file to the gaia.cs.umass.edu server. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.
7. Stop Wireshark packet capture. Your Wireshark window should look similar to the window shown below.

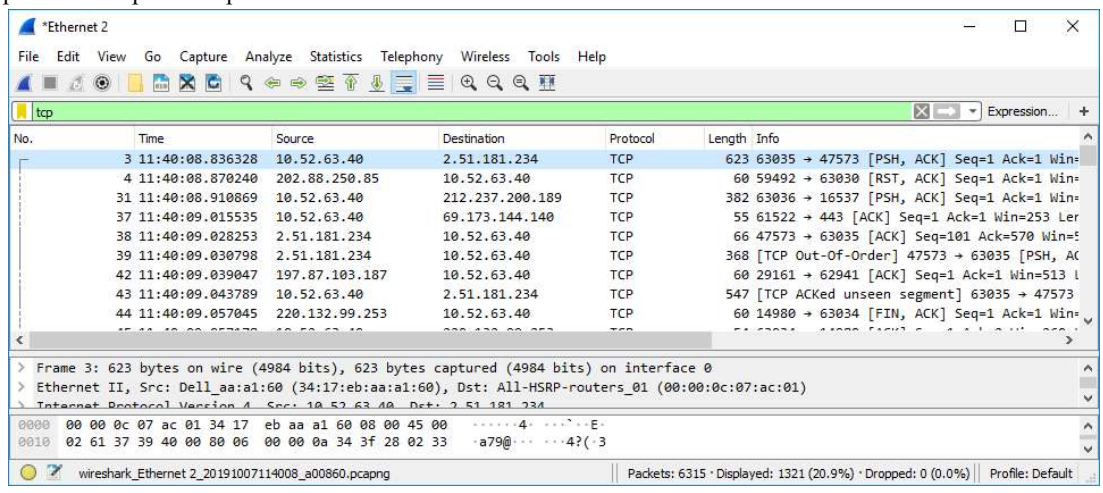


Figure 5.4: Wireshark window after packet capture

8. First, filter the packets displayed in the Wireshark window by entering "tcp" into the display filter specification window towards the top of the Wireshark window. What you should see is series of TCP and HTTP messages between your computer and gaia.cs.umass.edu. You should see the initial three-way handshake containing a SYN message. You should see an HTTP POST message and a series of "HTTP Continuation" messages being sent from your computer to gaia.cs.umass.edu. Recall from our discussion in the earlier HTTP Wireshark lab, that is no such thing as an HTTP Continuation message – this is Wireshark's way of indicating that there are multiple TCP segments being used to carry a single HTTP message. You should also see TCP ACK segments being returned from gaia.cs.umass.edu to your computer.
9. **Obtaining credit for this lab:** Now, please proceed to the questions section to answer the questions. You must note down your answers in this file itself. Please note that every student must upload this file (after duly filling in the answers) on Google Classroom to obtain credit. Please clarify with your instructor/lab engineer if you have any queries.

5.4.1. Questions

1. What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? To answer this question, it's probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "details of the selected packet header window".

Client IP Address: 10.5.88.203 Client TCP port number: 57383

No.	Time	Source	Destination	port	Protocol	Length	Info
…	08:27:06.471800	10.5.88.203	128.119.245.12	57383	TCP	66	57383 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256
…	08:27:07.011346	128.119.245.12	10.5.88.203	80	TCP	66	80 → 57383 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1
…	08:27:07.011629	10.5.88.203	128.119.245.12	57383	TCP	54	57383 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
…	08:27:07.014560	10.5.88.203	128.119.245.12	57383	HTTP	775	POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1
…	08:27:07.014990	10.5.88.203	128.119.245.12	57383	HTTP	13194	Continuation
…	08:27:08.034681	10.5.88.203	128.119.245.12	57383	HTTP	1514	Continuation
…	08:27:09.718301	10.5.88.203	128.119.245.12	57383	TCP	1514	[TCP Retransmission] 57383 → 80 [PSH, ACK] Seq=1 Ack=1
…	08:27:10.214526	128.119.245.12	10.5.88.203	80	TCP	60	80 → 57383 [ACK] Seq=1 Ack=1461 Win=32128 Len=0

2. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

Server IP Address: 128.119.245.12 Server TCP port number: 80

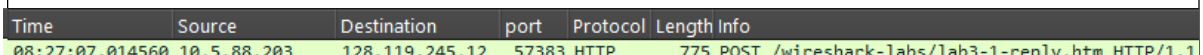
No.	Time	Source	Destination	port	Protocol	Length	Info
…	08:27:06.471800	10.5.88.203	128.119.245.12	57383	TCP	66	57383 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256
…	08:27:07.011346	128.119.245.12	10.5.88.203	80	TCP	66	80 → 57383 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1
…	08:27:07.011629	10.5.88.203	128.119.245.12	57383	TCP	54	57383 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
…	08:27:07.014560	10.5.88.203	128.119.245.12	57383	HTTP	775	POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1
…	08:27:07.014990	10.5.88.203	128.119.245.12	57383	HTTP	13194	Continuation
…	08:27:08.034681	10.5.88.203	128.119.245.12	57383	HTTP	1514	Continuation
…	08:27:09.718301	10.5.88.203	128.119.245.12	57383	TCP	1514	[TCP Retransmission] 57383 → 80 [PSH, ACK] Seq=1 Ack=1
…	08:27:10.214526	128.119.245.12	10.5.88.203	80	TCP	60	80 → 57383 [ACK] Seq=1 Ack=1461 Win=32128 Len=0

3. Recall the TCP lecture studied in class and explain the content of the TCP header like sequence number, acknowledgement number and checksum etc.

Source Port: The port number of the application in the sender host and is 16 bits in size.
Destination Port: The port number of the application in the receiver host and is 16 bits in size.
Sequence Number: It is a counter used to keep track of every byte sent outward by a host, and signifies the number of bytes transmitted till then.
Acknowledgement Number: It is a cumulative value in a TCP segment that indicates the byte number of the last received data byte, helping the sender know which bytes have been successfully received by the recipient.
Checksum: It is an error detection mechanism to determine the integrity of the data transmitted over a network. It is calculated by taking the binary value of all the fields in the TCP header and the data, treating them as a large integer, and then performing a bit-wise ones complement on that integer.
Header Length: This field specifies the number of 32-bit words present in the TCP header. This field helps the receiver to know from where the actual data begins. The HLEN field is of 4 bits, and it ranges from 20 bytes to 60 bytes in TCP header size.
Flags: The control flag bit is 6 bits, 1 bi each for URG, ACK, PSH, RST, SYN and FIN.
Receive Window Size: It tells how many the sender can receive without acknowledgement.
Urgent Data Pointer: It refers to data that should be processed as soon as possible.
Options: The options field contains 40 bytes that are used for several purposes. .

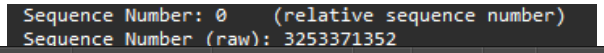
4. What type of http packets used in transferring file to gaia.cs.umass.edu?

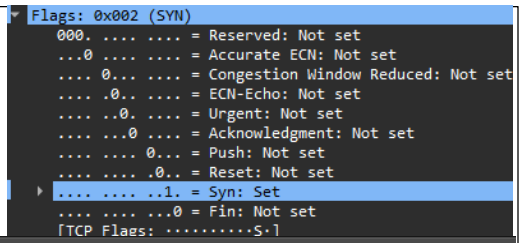
The POST (creating resources) type of HTTP packet was used to transfer the file onto the server.

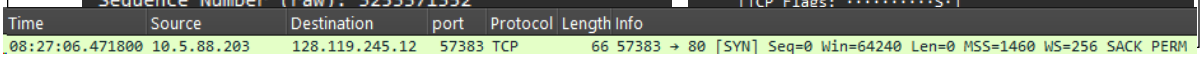


5. By looking into which field in TCP segment you can identify that a given segment is a SYN segment? What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu?

The Flags field in the Header of the TCP segment has SYN and ACK bits. When the SYN=1 and ACK=0, it implies that the given segment is a SYN segment.
The Sequence Number (Seq) of the first TCP connection is:
Seq (Relative) = 0
Seq (Raw) = 3253371352

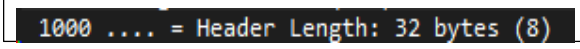


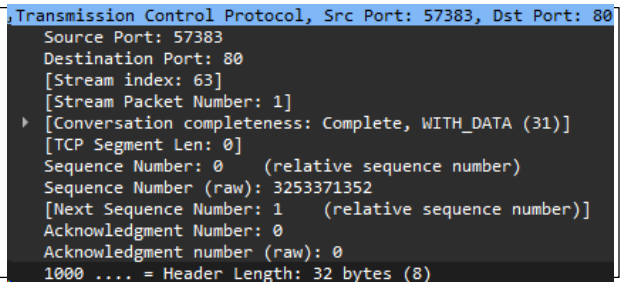




6. What is the header length of TCP verify it with Wire-shark?

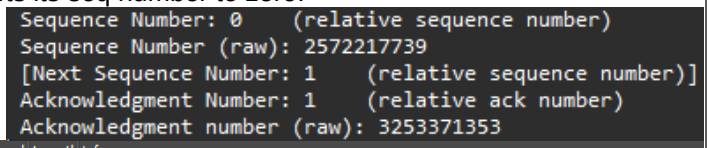
The TCP header length is 32 bytes.

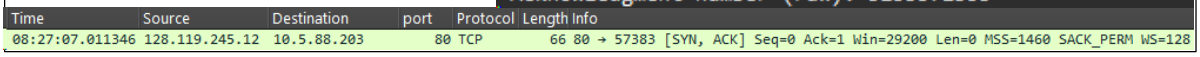




7. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the ACKnowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value?

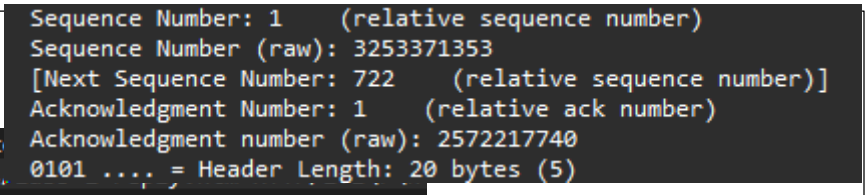
The ack number is calculated as the number of bytes successfully sent. Even though, this packet is not carrying any data but connection settings, the ack number is increased by 1 which tells the client it has received the SYN packet while it sets its seq number to zero.
Seq (Relative) = 0
Ack (Relative) = 1

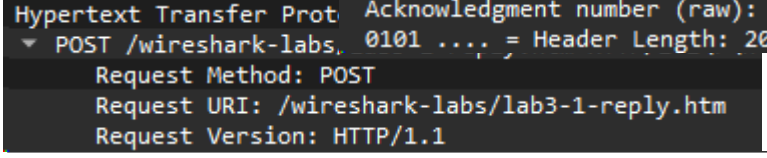




8. What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to look into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.

Seq (Relative) = 1
Seq (Raw) = 3253371352





9. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first 4 segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received?

1st segment:	Seq (Relative) = 1 Seq (Raw) = 3253371353 ACK time = 08:27:10.214526	Time = 08:27:07.014560
2nd segment:	Seq (Relative) = 722 Seq (Raw) = 3253372074 ACK time = 08:27:10.685399	Time = 08:27:07.014990
3rd segment:	Seq (Relative) = 13862 Seq (Raw) = 3253385214 ACK time = 08:27:11.247706	Time = 08:27:08.034681
4th segment:	Seq (Relative) = 15322 Seq (Raw) = 3253386674 ACK time = 08:27:11.768575	Time = 08:27:10.214637

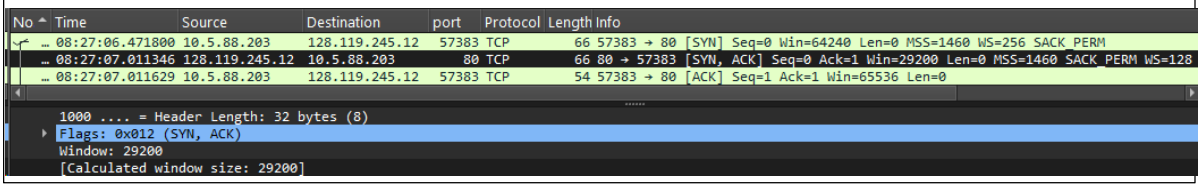
10. What is the length of each of the first four TCP segments?

TCP Segment Lengths:

1st segment	=	721	bytes data + 54 bytes header = 775	bytes total
2nd segment	=	13140	bytes data + 54 bytes header = 13194	bytes total
3rd segment	=	1460	bytes data + 54 bytes header = 1514	bytes total
4th segment	=	1460	bytes data + 54 bytes header = 1514	bytes total

11. What is the minimum amount of available buffer space advertised at the receiver?

The minimum amount of available buffer space advertised at the received for the entire trace is indicated first ACK from the server, its value is **29200 bytes.**



12. Calculate the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated it.

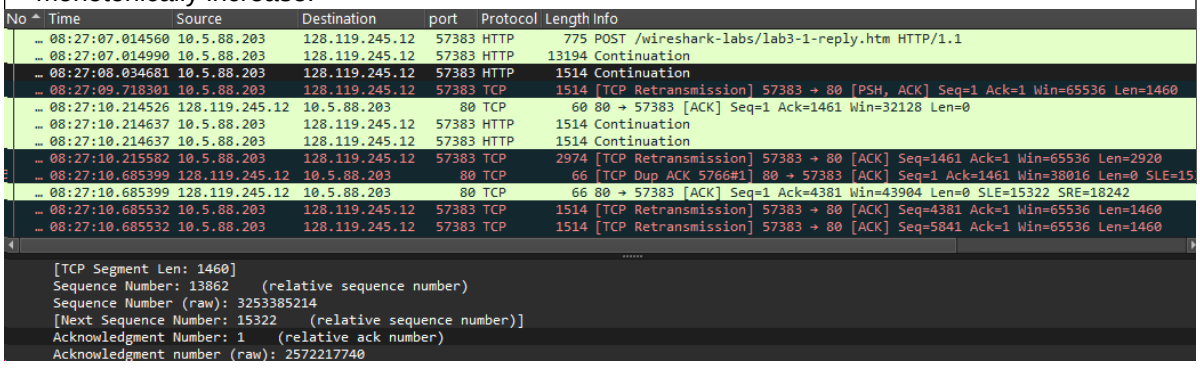
The alice.txt occupied the hard drive is 152,120 bytes.

1st ACK Time	=	08:27:10.214526	->	1st ACK Number = 1461
Last ACK Time	=	08:27:15.160761	->	Last ACK Number = 153025
Upload Time	=	00:00:04.946235	->	Bytes Transferred = 151564

As 151,564 bytes were transferred in 4.946235 seconds, the throughput for the TCP connection is computed as 151,564 / 4.946235 = **30,642.2966155 bytes/second**.

13. How can you find that a packet is retransmitted?

A retransmitted packet is flagged as "TCP Retransmission" in the info column in Wireshark. A packet can also be identified as retransmitted when the segment number does NOT monotonically increase.



Let's now examine the amount of data sent per unit time from the client to the server. Rather than calculating this from the raw data in the Wireshark window, we'll use one of Wireshark's TCP graphing utilities - Time-Sequence-Graph(Stevens) - to plot out data. Select a TCP segment in the Wireshark's "listing of captured-packets" window. Then select the menu : Statistics-> TCP Stream Graph-> Time-Sequence-Graph(Stevens).

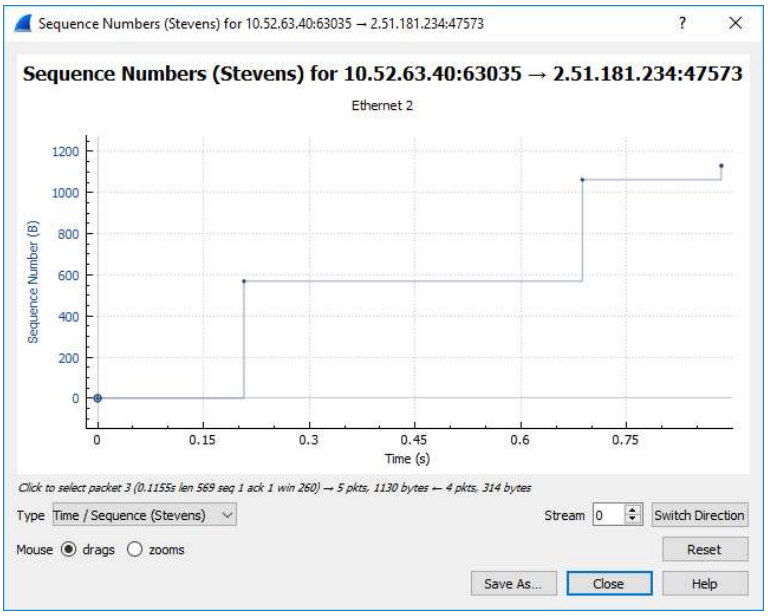
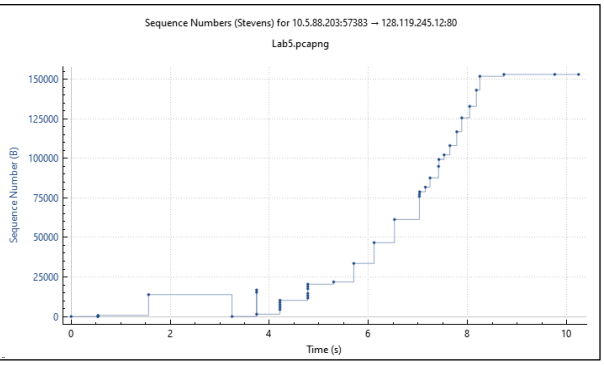


Figure 5.5: Time sequence graph (Stevens)

Here, each dot represents a TCP segment sent, plotting the sequence number of the segment versus the time at which it was sent. Note that a set of dots stacked above each other represents a series of packets that were sent back-to-back by the sender.

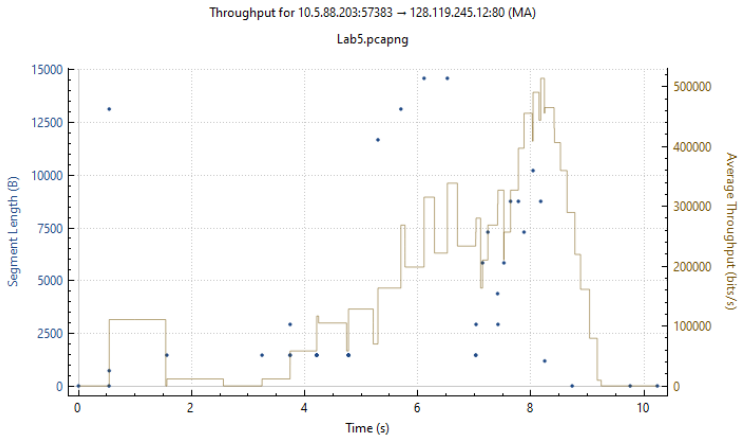
14. Use the Time-Sequence-Graph(Stevens) plotting tool and show the plot you obtained for the TCP segment. Also explain that graph in few lines.

The Sequence number **increases and decrease** between seconds 1 and 4 which implies that the packets were **retransmitted**.
The sequence number **starts from 0** and then it keeps increasing **till the last byte 151564** is transmitted successfully.
The x-axis represents time in seconds, and the y-axis represents TCP sequence number.
Each TCP segment is represented by a vertical line, and the length of the line represents the number of bytes in the segment.



15. Use the TCP stream graph and plot the throughput graph also explain it in few lines.

The x-axis represents time in seconds and the y-axis represents throughput in bytes per second. The throughput varies significantly during the file transmission. It **goes to near 0 between seconds 1 and 4**, which can be verified with the sequence number graph to see that the packets required retransmission. Then **as time goes on the throughput increases** until the transmission is near complete.



Assessment Rubrics for Computer Networks Lab 5

Student Name: _____ Roll Number: _____

Method:

Lab report evaluation and instructor observation during lab sessions.

Outcomes Assessed:

a. Ability to condut experiments as well as to analyze and interpret data
b. Ability to adhere to safety and disciplinary rules
c. Ability to use the techniques, skills and modern engineering tools necessary for engineering practice

Performance	Exceeds expectation (5-4)	Meets expectation (3-2)	Does not meet expectation (1)	Marks
Realization of experiment (a)	Downloads and installs required software and sets up the system according to the experiment requirements	Needs guidance to set up the system according to the experiment requirements	Incapable of selecting relevant software to the experiment and unable to setup the system with required software tools	
Conducting experiment (a, c)	Carries out each procedural step in a satisfactory manner and studies outputs of the software application rigorously	Needs assistance or guidance to proceed through experiment steps, studies outputs with minor errors in interpretation	Unable to carry out procedural steps and make any useful observations of outputs	
Laboratory safety and disciplinary rules (b)	Observes lab safety rules; adheres to the lab disciplinary guidelines aptly	Observes safety rules and disciplinary guidelines with minor deviations	Disregards lab safety and disciplinary rules	
Data collection (c)	Completes data collection from the experiment setup by following procedural steps, ensures that the data is entered in the lab manual according to the specified instructions	Completes data collection with minor error and enters data in lab manual with slight deviation from guidelines	Fails at collecting data by giving proper inputs and observing output states of experiment setup, unable to fill the lab manual properly	
Data analysis (a, c)	Analyzes the data obtained from experiment thoroughly and accurately verifies it with theoretical understanding, accounts for any discrepancy in data from theory with sound explanation	Analyzes data with minor error and correlates it with theoretical values reasonably. Attempts to account for any discrepancy in data from theory	Unable to establish the relationship between practical and theoretical values and lacks the theoretical understanding to explain any discrepancy in data	