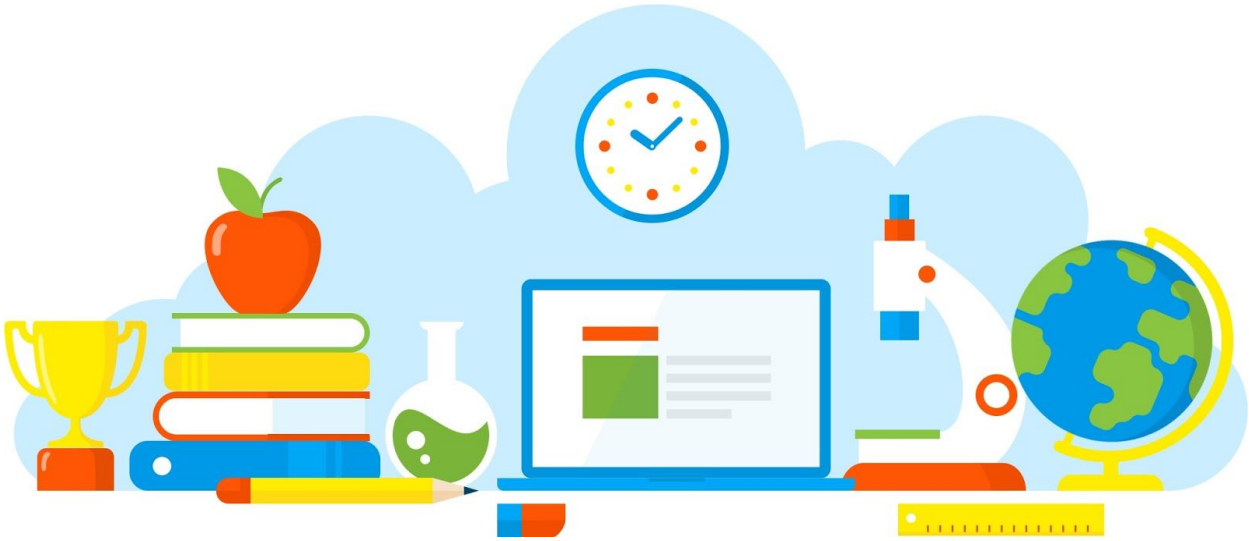


Assignment Report

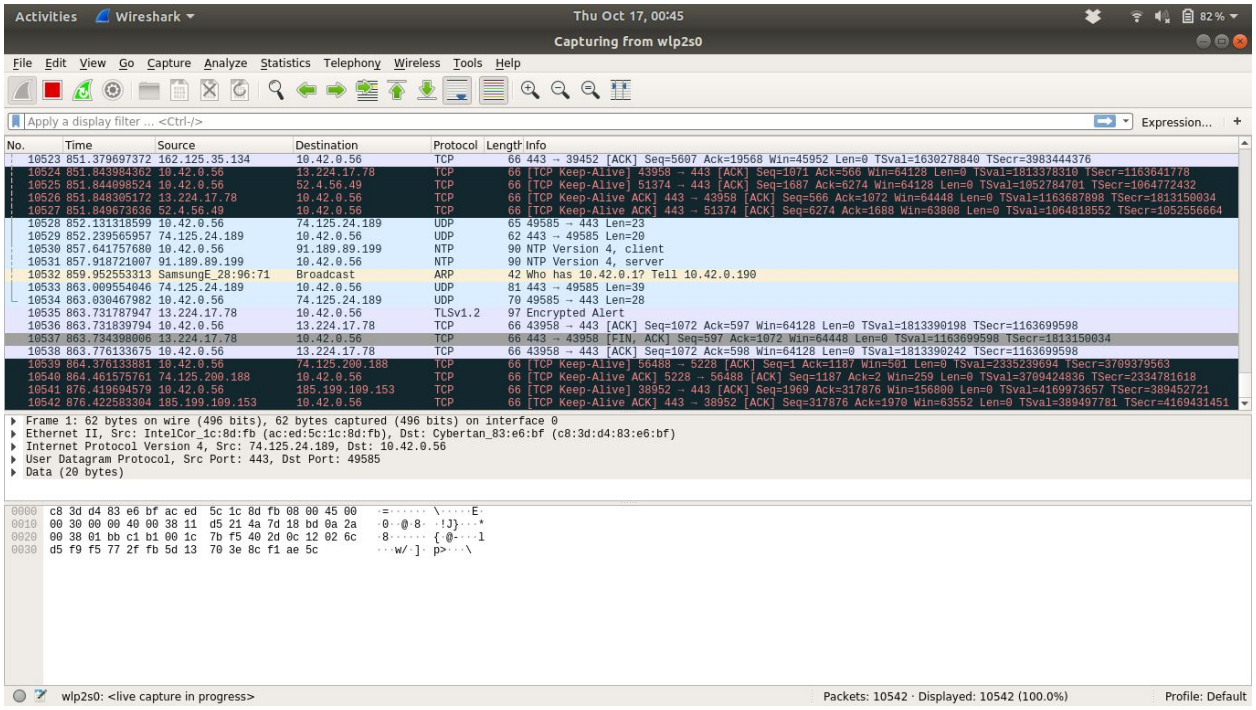
*This report contains work from **Assignment 9** of course CSN 361.*

Name- Kaustubh Trivedi
Enrollment Number - 17114044
Class - CSE B.Tech. 3rd Year

Submission Files - Repository Link

QUESTION 1: Install Wireshark and explore its uses to capture network traffic. You have to capture normal internet traffic for 20-30 minutes from your system using Wireshark. You need to copy this data in CSV / TXT file.

SOLUTION SCREENSHOT:



QUESTION 2: Take the CSV / TXT, which is generated in Problem Statement 1 as an input. Write a code (in any programming language of your choice) to extract the following 11 features given below in the table:

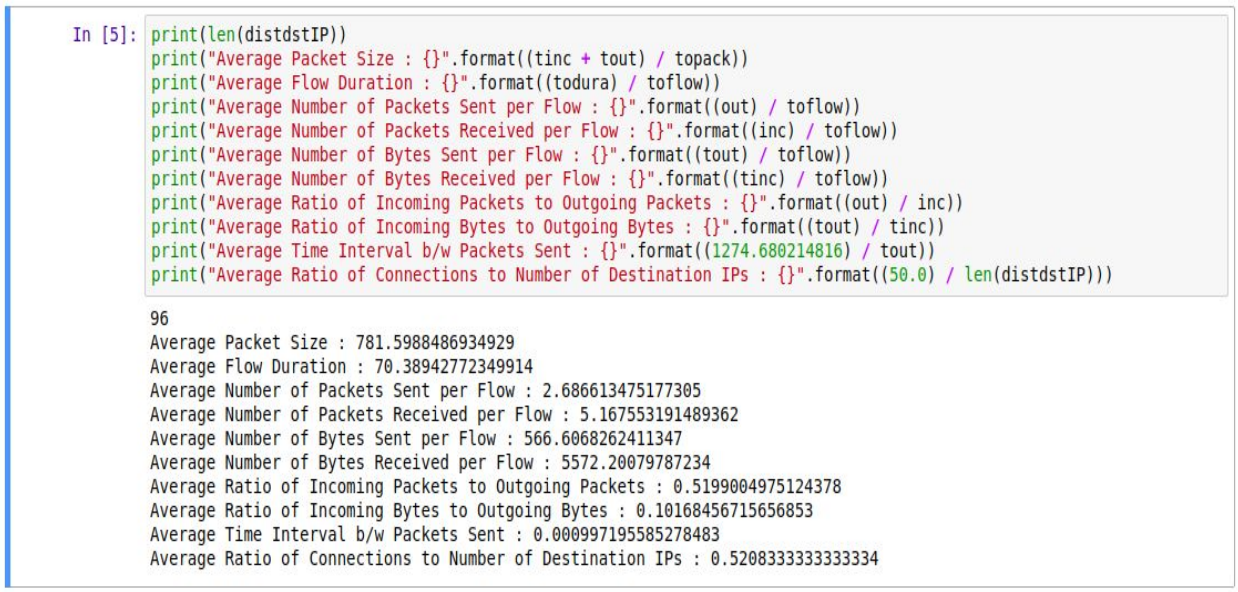
Average Packet Size	Average Flow Duration
Average no of Packets Sent per Flow	Average no of Packets Received per Flow
Average amount of Bytes Sent per Flow	Average amount of Bytes Received per Flow
Average Ratio of Incoming to Outgoing Packets	Average Ratio of Incoming to Outgoing Bytes
Average Time Interval b/w Packets Sent	Average Time Interval b/w Packets Received
Average Ratio of Connections to Number of Destination IPs	

SOLUTION CODE:

```
import csv

reader = csv.DictReader(open("Q1_out.csv"))
tout = 0
tinc = 0
out = 0
inc = 0
inflow = dict()
outflow = dict()
distdstIP = dict()

for raw in reader:
      key = raw["Source"] + raw["Destination"] + raw["Protocol"] +
raw["Source Port"] + raw["Destination Port"]
      distdstIP[raw["Destination"]] = 0
      if key not in inflow:
            inflow[key] = 900
            outflow[key] = 0
      outflow[key] = max(outflow[key], float(raw["Time"]))
      inflow[key] = min(inflow[key], float(raw["Time"]))
      if raw["Source"] == "10.42.0.56":
            out += 1
            tout += int(raw["Length"])
      else:
            inc += 1
            tinc += int(raw["Length"])

toflow = len(outflow) + len(inflow)
todura = 0
topack = out + inc
```

```
for k in inflow:
      todura += (outflow[k] - inflow[k])

print(len(distdstIP))
print("Average Packet Size : {}".format((tinc + tout) / topack))
print("Average Flow Duration : {}".format((todura) / toflow))
print("Average Number of Packets Sent per Flow : {}".format((out) /
toflow))
print("Average Number of Packets Received per Flow : {}".format((inc)
/ toflow))
print("Average Number of Bytes Sent per Flow : {}".format((tout) /
toflow))
print("Average Number of Bytes Received per Flow : {}".format((tinc)
/ toflow))
print("Average Ratio of Incoming Packets to Outgoing Packets :
{}".format((out) / inc))
print("Average Ratio of Incoming Bytes to Outgoing Bytes :
{}".format((tout) / tinc))
print("Average Time Interval b/w Packets Sent :
{}".format((1274.680214816) / tout))
print("Average Ratio of Connections to Number of Destination IPs :
{}".format((50.0) / len(distdstIP)))
```

RESULT SCREENSHOTS:



```
In [5]: print(len(distdstIP))
        print("Average Packet Size : {}".format((tinc + tout) / topack))
        print("Average Flow Duration : {}".format((todura) / toflow))
        print("Average Number of Packets Sent per Flow : {}".format((out) / toflow))
        print("Average Number of Packets Received per Flow : {}".format((inc) / toflow))
        print("Average Number of Bytes Sent per Flow : {}".format((tout) / toflow))
        print("Average Number of Bytes Received per Flow : {}".format((tinc) / toflow))
        print("Average Ratio of Incoming Packets to Outgoing Packets : {}".format((out) / inc))
        print("Average Ratio of Incoming Bytes to Outgoing Bytes : {}".format((tout) / tinc))
        print("Average Time Interval b/w Packets Sent : {}".format((1274.680214816) / tout))
        print("Average Ratio of Connections to Number of Destination IPs : {}".format((50.0) / len(distdstIP)))

        96
        Average Packet Size : 781.5988486934929
        Average Flow Duration : 70.38942772349914
        Average Number of Packets Sent per Flow : 2.686613475177305
        Average Number of Packets Received per Flow : 5.167553191489362
        Average Number of Bytes Sent per Flow : 566.6068262411347
        Average Number of Bytes Received per Flow : 5572.20079787234
        Average Ratio of Incoming Packets to Outgoing Packets : 0.5199004975124378
        Average Ratio of Incoming Bytes to Outgoing Bytes : 0.10168456715656853
        Average Time Interval b/w Packets Sent : 0.000997195585278483
        Average Ratio of Connections to Number of Destination IPs : 0.5208333333333334
```

QUESTION 3: In this problem, the behavior of TCP protocol will be studied using Wireshark. For this assignment download the Wireshark captured trace file named as **tcpethe-trace** from Piazza, which is a packet trace of TCP transfer of a file from a client system to a remote server (named as ser1), obtained by running Wireshark on the client machine. Open **tcpethe-trace** file in Wireshark and answer the following question:

a. What is the IP address and TCP port number used by the client computer (source) that is transferring the file to server (*ser1)*?

b. What is the IP address of server (*ser1*)? On what port number it is sending and receiving the TCP segments for this connection?

c. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and *ser1*? What is it in the segment that identifies the segment as a SYN segment?

d. What is the sequence number of the SYNACK segment sent by ser1 to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did ser1 determine that value? What is it in the segment that identifies the segment as a SYNACK segment?

e. 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 dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.

f. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six 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? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the Round Trip Time (RTT) value for each of the six segments? What is the Estimated RTT value after the receipt of each ACK? Assume that the value of the Estimated RTT is equal to the measured RTT for the first segment, and then is computed using the following Estimated RTT equation for all subsequent segments.

$$\textbf{Estimated RTT} = (1 - \alpha) * \textbf{Estimated RTT} + \alpha * \textbf{SampleRTT}$$

where, the new value of Estimated RTT is a weighted combination of the previous value of Estimated RTT and the new value for SampleRTT. The recommended value of $\alpha = 0.125$.

Note: Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the "listing of captured packets" window that is being sent from the client to the ser1 server. Then select: Statistics→TCP Stream Graph→Round Trip Time Graph.

g. What is the length of each of the first six TCP segments?

h. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

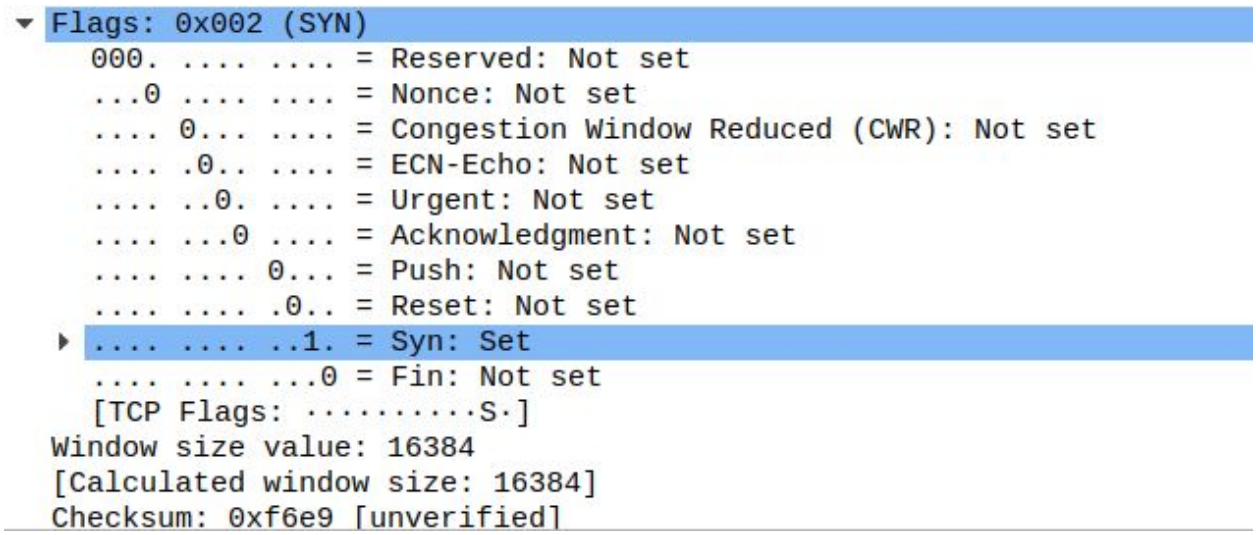
i. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

SOLUTION:

A. 192.168.1.102:1161

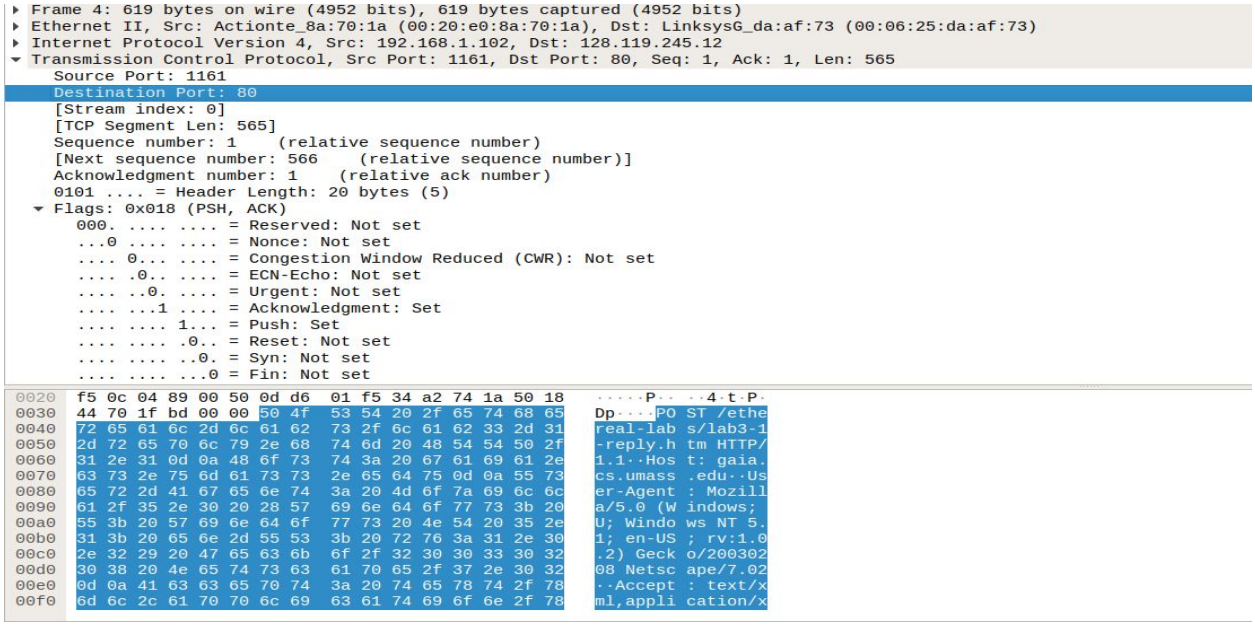
B. 128.119.245.12:80

C. seq = 0, Flag being 0x002 signifies SYN segment bit is 1 and rest are 0. Refer below for screenshot:

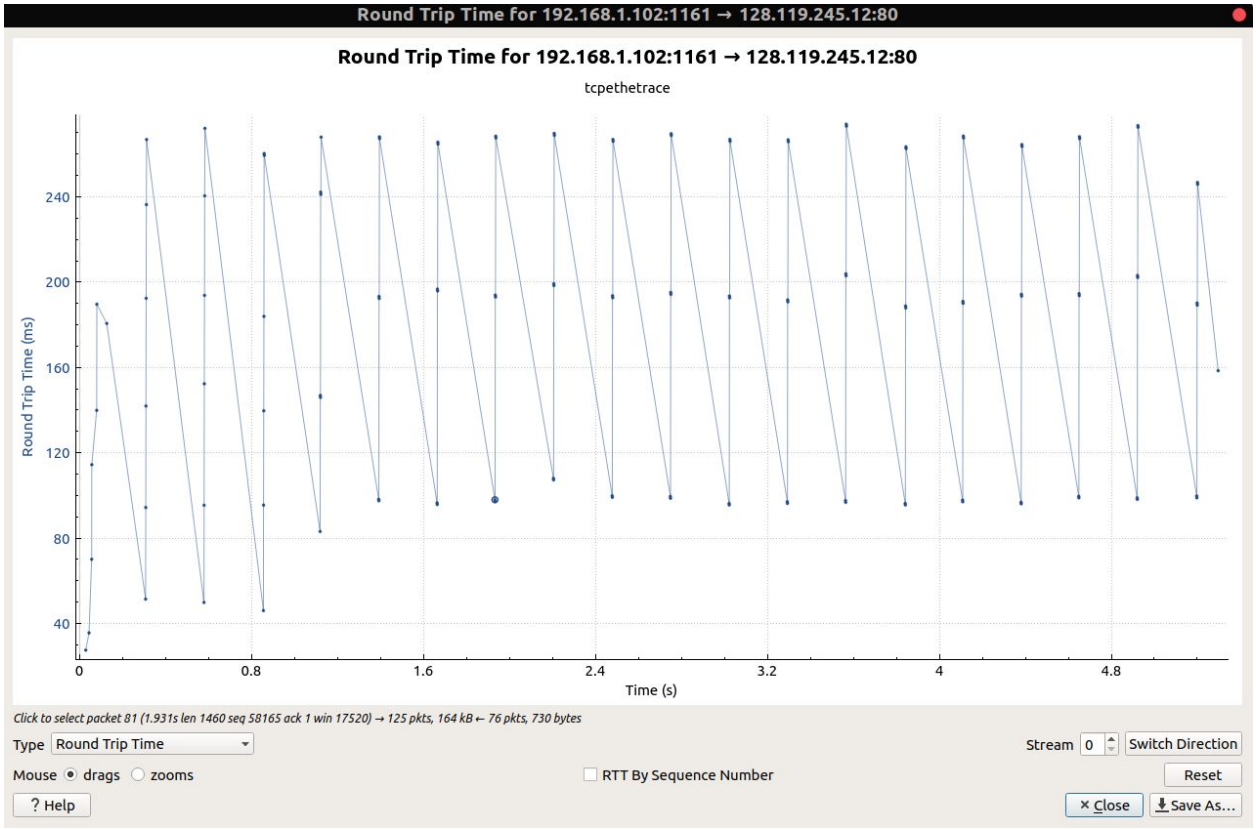


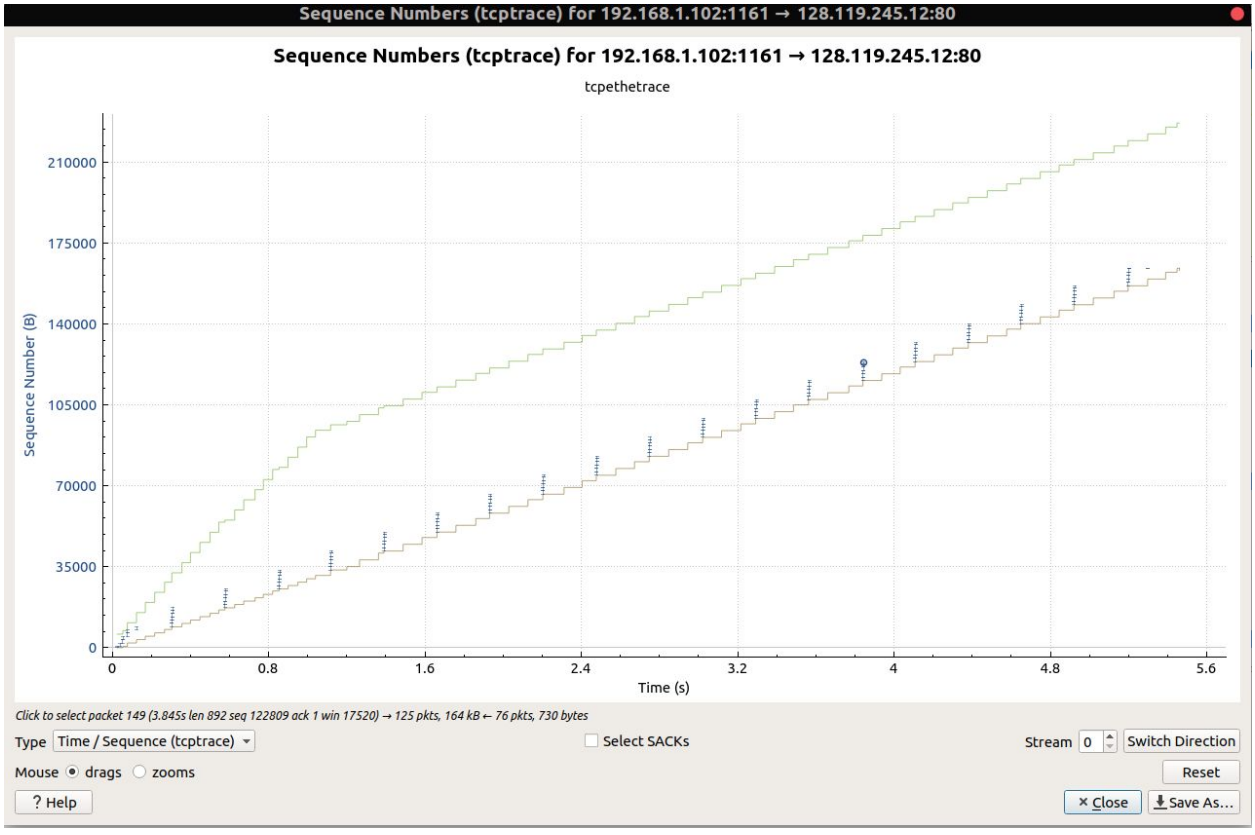
D. 0, Ack field is set to 1, ser1 determined that value by inverting the ack bit received in the SYN segment sent previously to initiate TCP communication, SYN and ACK segments are set to 1 and the rest are 0. e. What is the sequence number of the TCP segm

E. seq = 1



F. Seq numbers of first 6 segments in TCP connection = 1,566, 2026, 3486, 4946, 6406
Time of sending for each of the first 6 segments = 0.026477, 0.041737, 0.054026, 0.05469, 0.077405, 0.078157
Length of each segment is 1460 bytes
Times can be seen from the graph for sending and ACK received.
Est. RTT1 = 0.02746
Est. RTT2 = 0.02746 * 0.75 + 0.25 * 0.035557 = 0.009621546415
Est. RTT3 = 0.009621546415 * 0.75 + 0.25 * 0.070059 = 0.0180203069402164
Est. RTT4 = 0.0180203069402164 * 0.75 + 0.25 * 0.114428 = 0.0301535207619163
Est. RTT5 = 0.0301535207619163 * 0.75 + 0.25 * 0.139894 = 0.0381372224751006
Est. RTT6 = 0.0381372224751006 * 0.75 + 0.25 * 0.189645 = 0.0528356501672178





G. 1460 bytes

H. The minimum amount of buffer space (receiver window) advertised at ser1 for the entire trace is 5840 bytes from the first acknowledgement from the server.

This receiver window grows steadily until a maximum receiver buffer size of 62780 bytes.
The sender is never throttled due to lacking of receiver buffer space by inspecting this trace.

I. The computation of TCP throughput largely depends on the selection of averaging time period.
We select the average time period as the whole connection time.
The average throughput for this TCP connection = ratio between the total amount data and the total transmission time.
The total amount data transmitted can be computed by the difference between the sequence number of the first TCP segment (i.e. 1 byte for No. 4 segment) and the acknowledged sequence number of the last ACK (164091 bytes for No. 202 segment).

Total data = 164091 - 1 = 164090 bytes.

The whole transmission time is the difference of the time instant of the first TCP segment (i.e., 0.026477 second for No.4 segment) and the time instant of the last ACK (i.e., 5.455830 second for No. 202 segment).
Total transmission time = 5.455830 - 0.026477 = 5.4294 seconds.
Hence, the throughput for the TCP connection is computed as 164090/5.4294 = 30.222 KByte/sec.