USC EE450 Fall 2020

Lab #2 Report: TCP

Zeyu Wang

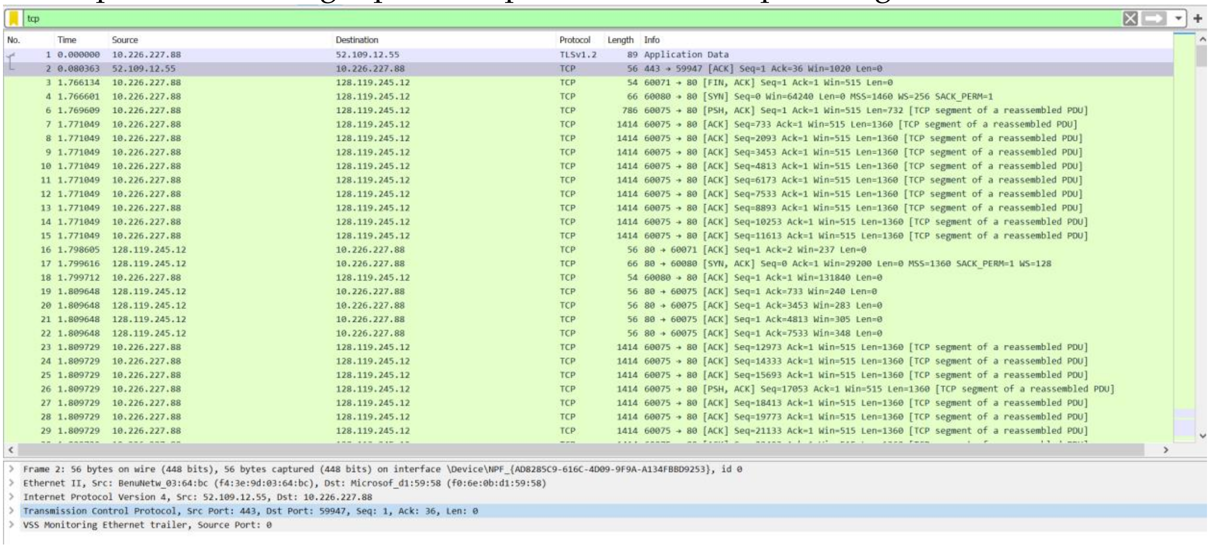
Session 2

1. Abstract

In this lab, I examined and analyzed the use of TCP on a public network by using the Wireshark packet analyzer that runs on Windows, this examination process including the trace of TCP segments, TCP sequence and acknowledgement numbers, and TCP congestion & flow control mechanisms.

Retrieve alic.txt and upload it on [http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html.](http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html) Record the trace of TCP & HTTP exchange information.

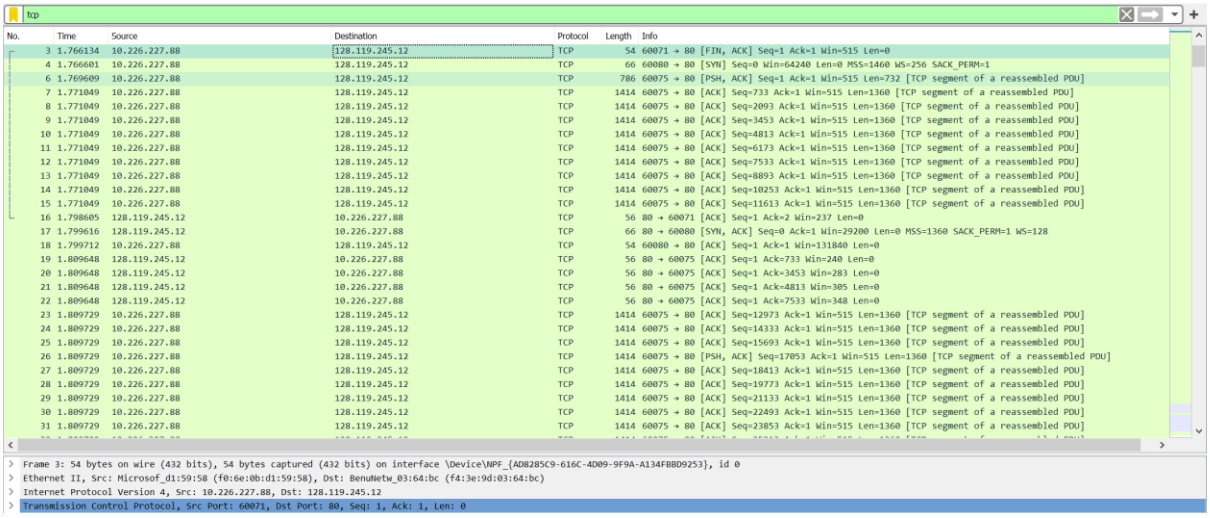
Boot up Wireshark, begin packet capture, and filter tcp messages.


tcp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.226.227.88	52.109.12.55	TLSv1.2	89	Application Data
2	0.080363	52.109.12.55	10.226.227.88	TCP	56	443 → 59947 [ACK] Seq=1 Ack=36 Win=1020 Len=0
3	1.766134	10.226.227.88	128.119.245.12	TCP	54	60071 → 80 [FIN, ACK] Seq=1 Ack=1 Win=515 Len=0
4	1.766601	10.226.227.88	128.119.245.12	TCP	66	60080 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
6	1.769609	10.226.227.88	128.119.245.12	TCP	786	60075 → 80 [PSH, ACK] Seq=1 Ack=1 Win=515 Len=732 [TCP segment of a reassembled PDU]
7	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=733 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
8	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=2093 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
9	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=3453 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
10	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=4813 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
11	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=6173 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
12	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=7533 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
13	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=8893 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
14	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=10253 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
15	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=11613 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
16	1.798605	128.119.245.12	10.226.227.88	TCP	56	80 → 60071 [ACK] Seq=1 Ack=2 Win=237 Len=0
17	1.799616	128.119.245.12	10.226.227.88	TCP	66	80 → 60080 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1360 SACK_PERM=1 WS=128
18	1.799712	10.226.227.88	128.119.245.12	TCP	54	60080 → 80 [ACK] Seq=1 Ack=1 Win=131840 Len=0
19	1.809648	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=733 Win=240 Len=0
20	1.809648	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=3453 Win=283 Len=0
21	1.809648	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=4813 Win=305 Len=0
22	1.809648	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=7533 Win=348 Len=0
23	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=12973 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
24	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=14333 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
25	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=15693 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
26	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [PSH, ACK] Seq=17053 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
27	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=18413 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
28	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=19773 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
29	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=21133 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]

> Frame 2: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{AD8285C9-616C-4D09-9F9A-A134FBBD9253}, id 0
> Ethernet II, Src: BenuNetw_03:64:bc (f4:3e:9d:03:64:bc), Dst: Microsof_d1:59:58 (f0:6e:0b:d1:59:58)
> Internet Protocol Version 4, Src: 52.109.12.55, Dst: 10.226.227.88
> Transmission Control Protocol, Src Port: 443, Dst Port: 59947, Seq: 1, Ack: 36, Len: 0
> VSS Monitoring Ethernet trailer, Source Port: 0

Cumulative Acknowledgments:

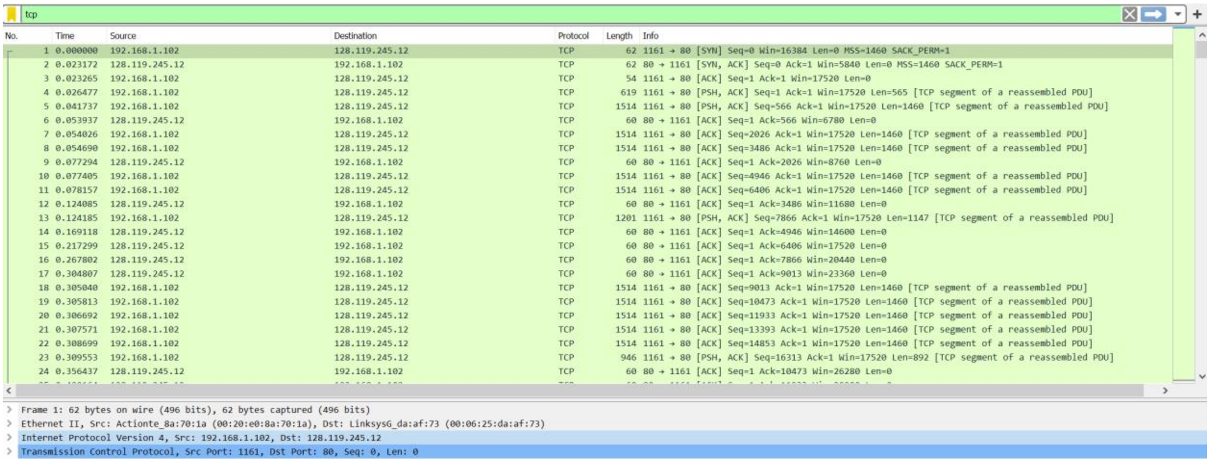

tcp

No.	Time	Source	Destination	Protocol	Length	Info
3	1.766134	10.226.227.88	128.119.245.12	TCP	54	60071 → 80 [FIN, ACK] Seq=1 Ack=1 Win=515 Len=0
4	1.766601	10.226.227.88	128.119.245.12	TCP	66	60080 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
6	1.769609	10.226.227.88	128.119.245.12	TCP	786	60075 → 80 [PSH, ACK] Seq=1 Ack=1 Win=515 Len=732 [TCP segment of a reassembled PDU]
7	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=733 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
8	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=2093 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
9	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=3453 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
10	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=4813 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
11	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=6173 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
12	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=7533 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
13	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=8893 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
14	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=10253 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
15	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=11613 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
16	1.798605	128.119.245.12	10.226.227.88	TCP	56	80 → 60071 [ACK] Seq=1 Ack=2 Win=237 Len=0
17	1.799616	128.119.245.12	10.226.227.88	TCP	66	80 → 60080 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1360 SACK_PERM=1 WS=128
18	1.799712	10.226.227.88	128.119.245.12	TCP	54	60080 → 80 [ACK] Seq=1 Ack=1 Win=131840 Len=0
19	1.809648	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=733 Win=240 Len=0
20	1.809648	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=3453 Win=283 Len=0
21	1.809648	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=4813 Win=305 Len=0
22	1.809648	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=7533 Win=348 Len=0
23	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=12973 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
24	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=14333 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
25	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=15693 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
26	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [PSH, ACK] Seq=17053 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
27	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=18413 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
28	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=19773 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
29	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=21133 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
30	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=22493 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
31	1.809729	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=23853 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]

> Frame 3: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{AD8285C9-616C-4D09-9F9A-A134FBBD9253}, id 0
> Ethernet II, Src: Microsof_d1:59:58 (f0:6e:0b:d1:59:58), Dst: BenuNetw_03:64:bc (f4:3e:9d:03:64:bc)
> Internet Protocol Version 4, Src: 10.226.227.88, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 60071, Dst Port: 80, Seq: 1, Ack: 1, Len: 0

2. Answers to questions in lab

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?
   Client IP address: 192.168.1.102 | TCP port number: 1161



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?

IP address of gaia.cs.umass.edu is 128.119.245.12

Port number is 80 that is sending and receiving TCP segments.

3) What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?

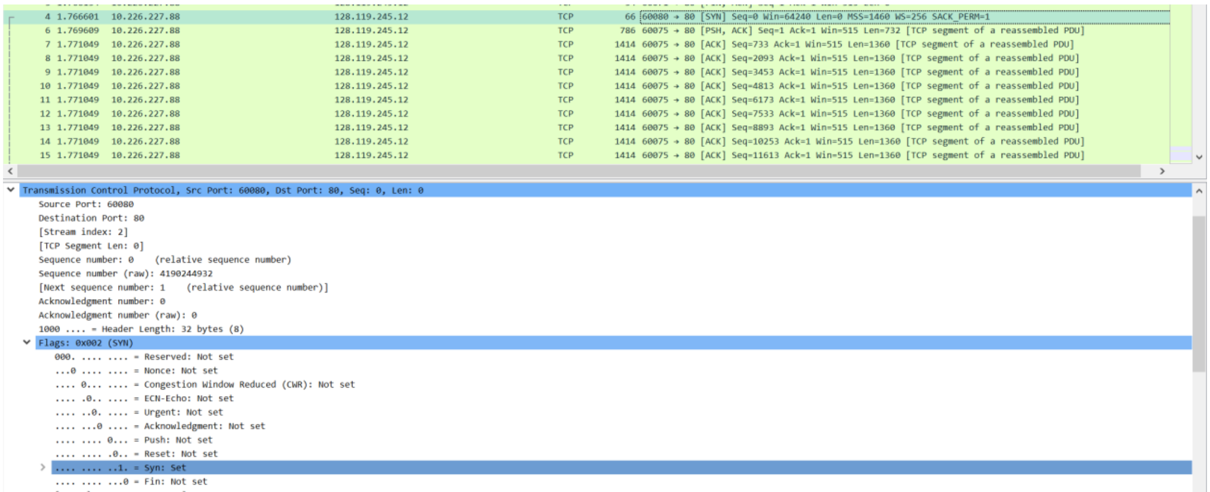
Client IP address is 10.226.227.88

TCP port number is 80 to transfer file to umass.

4) 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? What is it in the segment that identifies the segment as a SYN segment?

Sequence number of the TCP SYN segment is 0.

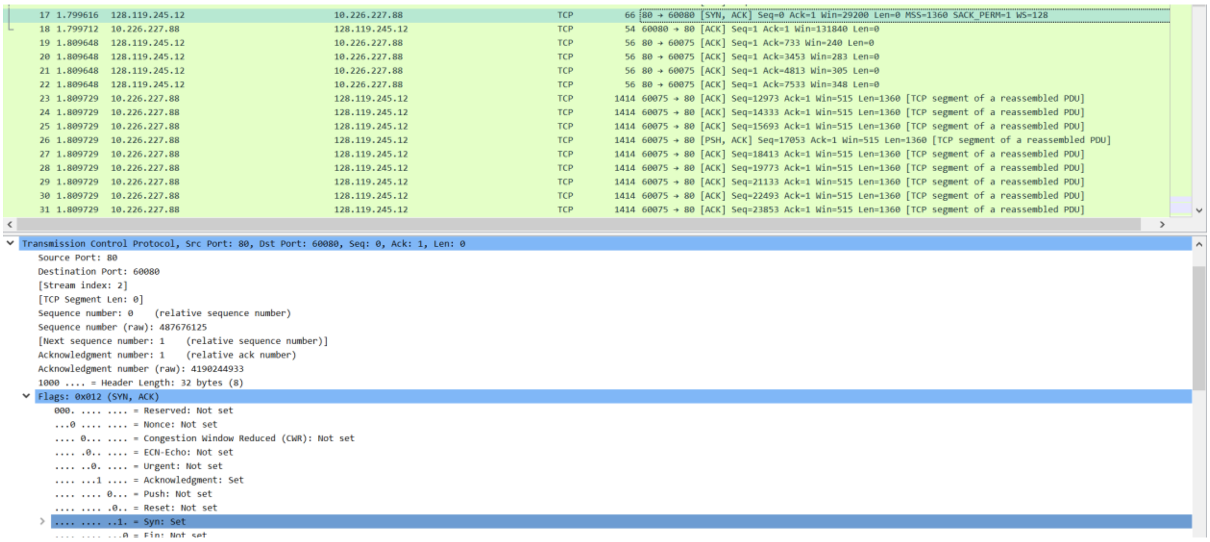
SYN flag is set to 1 to identifies the SYN segment.



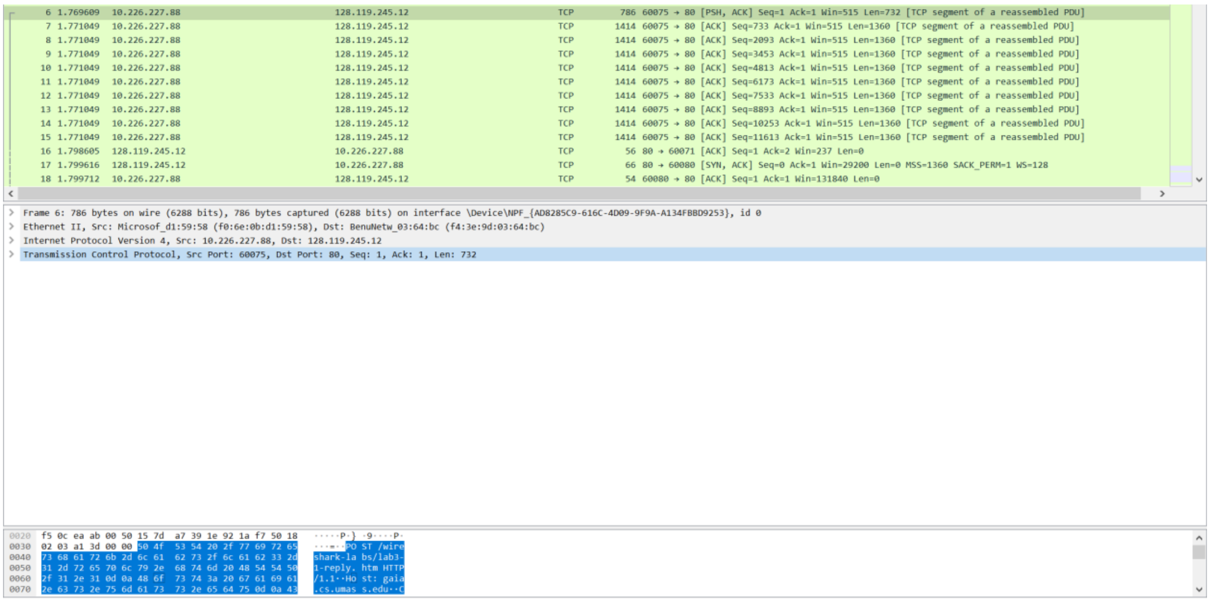
5) What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?

Sequence number of SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN is 0. The value of the Acknowledgement field in the SYNACK segment is 1, which is determined by the web server. The web server adds 1 to the initial sequence number of the SYN segment that sent from the client. Flag is in the segment

that identifies the segment as a SYNACK segment. If both flag of SYN and ACK are set to 1, then it's a SYNACK segment.

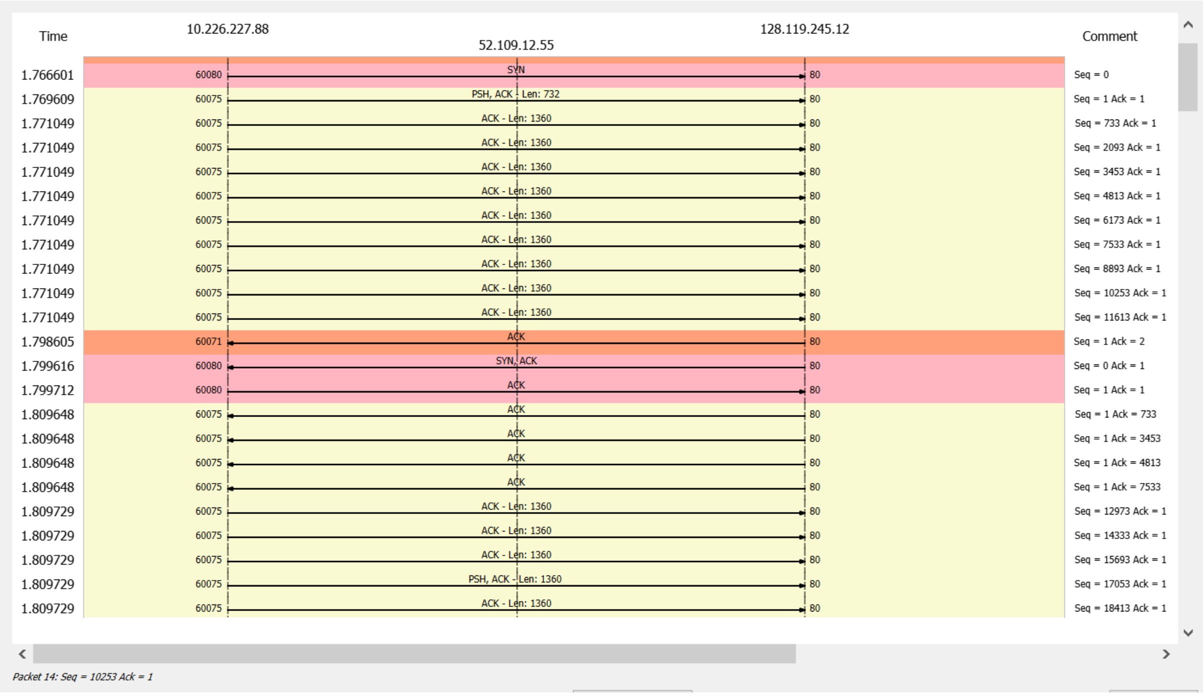


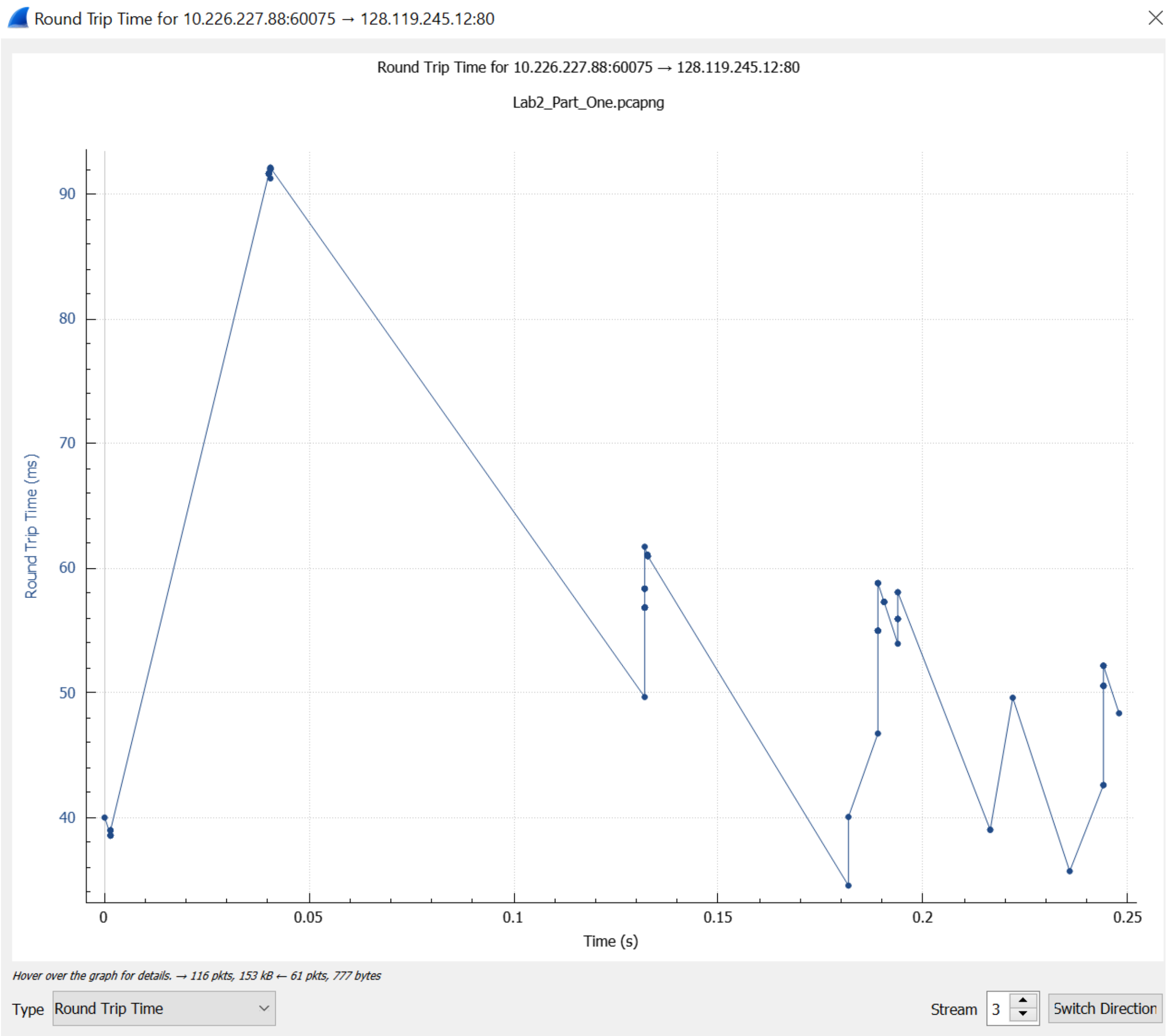
6) What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.
The sequence number of the TCP segment containing the HTTP POST command is 1.



7) 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 RTT value for each of the six segments? What is the `EstimatedRTT` value (see Section 3.5.3, page 242 in text) after the receipt of each ACK? Assume that the value of the `EstimatedRTT` is equal to the measured RTT for the first segment, and then is computed using the `EstimatedRTT` equation on page 242 for all subsequent segments.
First Six Segments in the TCP connection, ACK received time, RTT:

a) Seq = 1 sent at 1.769609; ACK received at 1.809648; RTT = 0.40039 second
b) Seq = 733 sent at 1.771049; ACK received at 0.809648; RTT = 0.38599 second
c) Seq = 2093 sent at 1.771049; ACK received at 1.809648; RTT = 0.038599 second
d) Seq = 3453 sent at 1.771049; ACK received at 1.809648; RTT = 0.038599 second
e) Seq = 4813 sent at 1.771049; ACK received at 1.809648; RTT = 0.038599 second
f) Seq = 6173 sent at 1.771049; ACK received at 1.809648; RTT = 0.038599 second





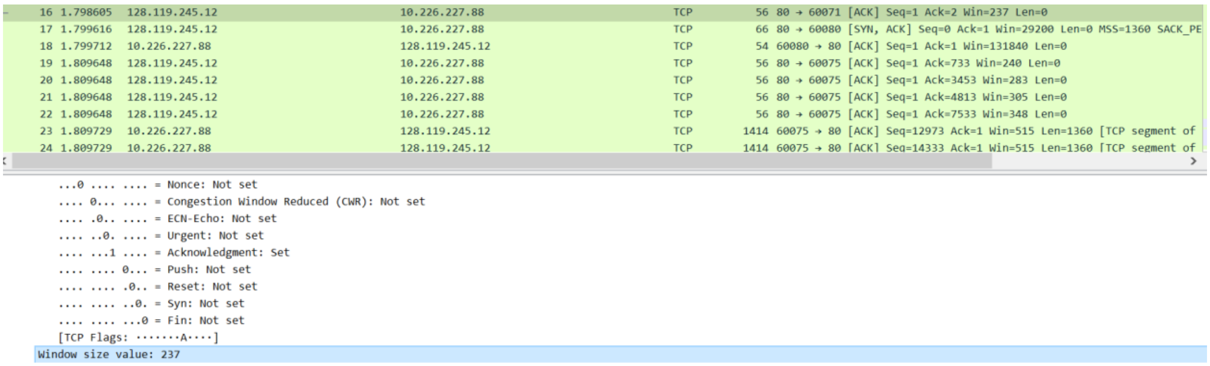
EstimatedRTT = 0.875 * EstimatedRTT + 0.125 * SampleRTT

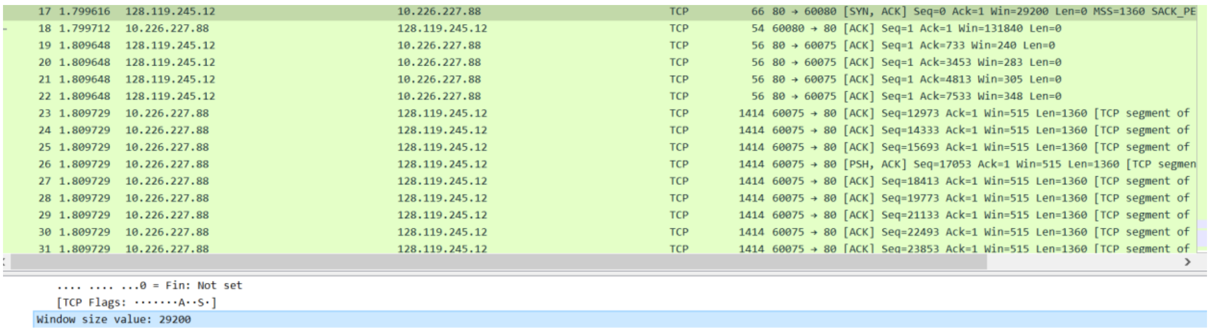
a) EstimatedRTT (Segment 1) = RTT for Segment 1 = 0.40039 second
b) EstimatedRTT (Segment 2) = 0.875 * 0.40039 + 0.125 *0.38599 = 0.35909862 second
c) EstimatedRTT (Segment 3) = 0.875 * 0.35909862+ 0.125 *0.38599 = 0.36246005 second
d) EstimatedRTT (Segment 4) = 0.875 *0.36246005 + 0.125 *0.38599 = 0.36797488 second
e) EstimatedRTT (Segment 5) = 0.875 * 0.36797488 + 0.125 * 0.38599 = 0.37022677 second
f) EstimatedRTT (Segment 6) = 0.875 * 0.37022677 + 0.125 * 0.38599 = 0.37219717 second

8) What is the length of each of the first six TCP segments?

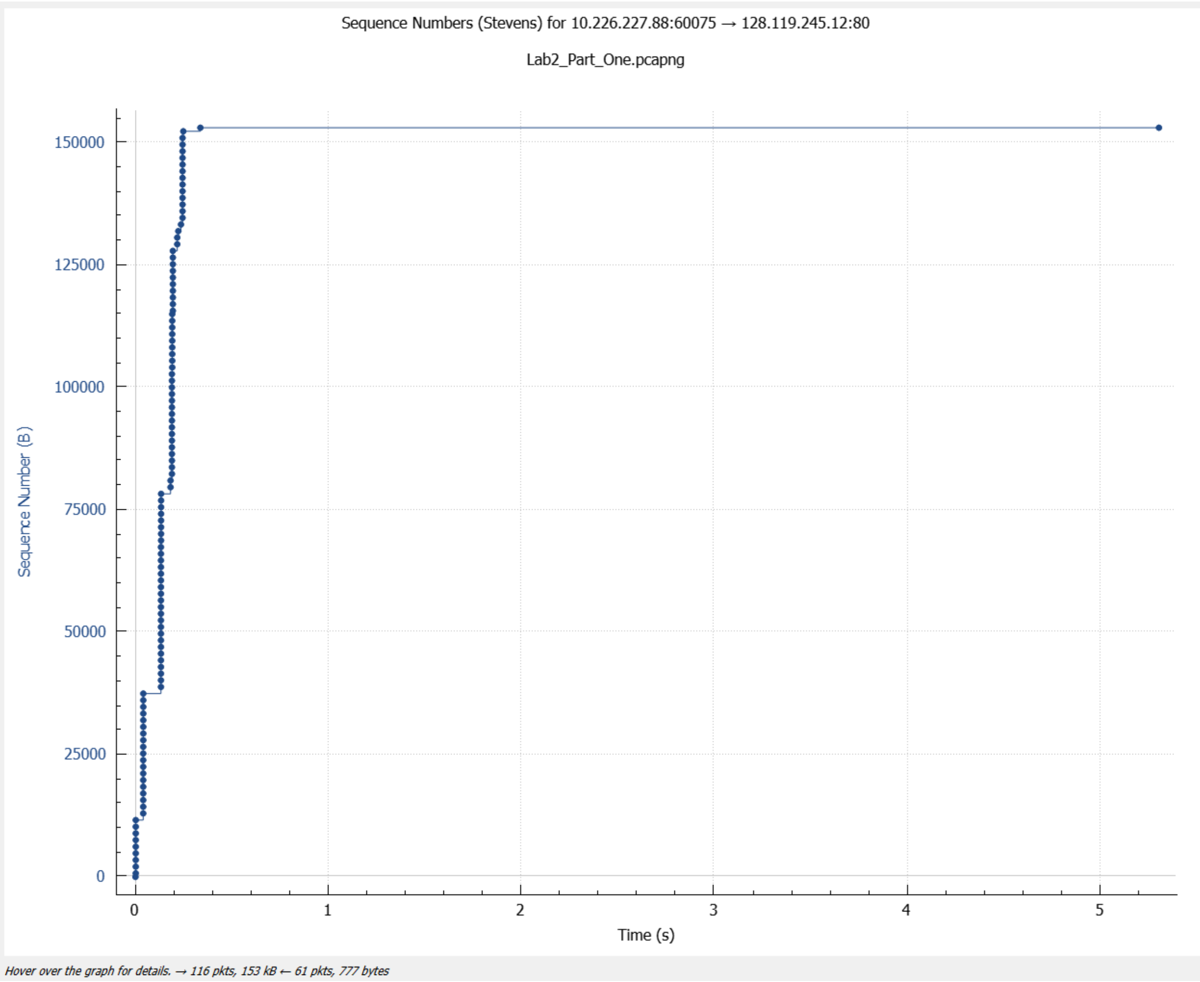
Segment	Length
1	732
2	1360
3	1360
4	1360
5	1360
6	1360

9) 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?
The minimum amount of buffer space advertised at the received for the entire trace is 237 bytes. The send will not throttle since the lack of receiver buffer space. The maximum receiver buffer size is 29200 bytes.





10) Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?
There are no retransmitted segments in the trace file. I checked the sequence number of the TCP segments in the trace file. The Time-Sequence graph showed that the time increased steadily and didn't change suddenly. If retransmitted segment occurred, then the sequence number of this retransmitted segment should be smaller than its neighboring segments. Thus, there are no retransmitted sequence.



11) How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACK'ing every other received segment (see Table 3.2 on page 250 in the text)?

The receiver typically acknowledges 1360 bytes in an ACK. (In the graph below, the difference between each consecutive ACK is 1360 bytes.)

115	1.959900	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=59213 Win=1177 Len=0
116	1.959900	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=61933 Win=1220 Len=0
117	1.959900	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=63293 Win=1243 Len=0
118	1.959900	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=64653 Win=1265 Len=0
119	1.959900	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=66013 Win=1288 Len=0
120	1.959900	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=67373 Win=1311 Len=0

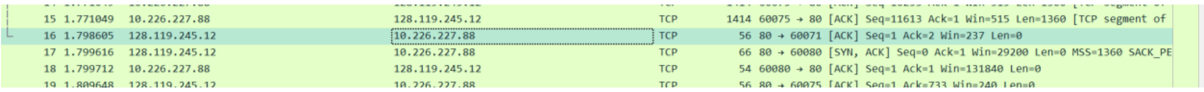
The receiver is ACKing every other segment. For instance, segment #20 is acking #8 and #9 two segments.

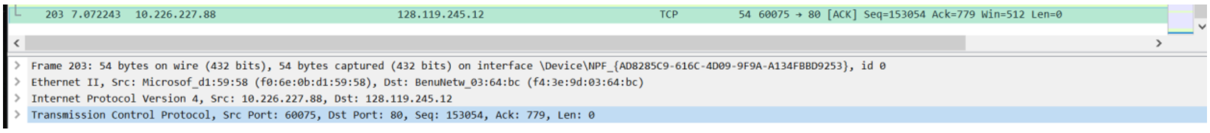
6	1.769609	10.226.227.88	128.119.245.12	TCP	786	60075 → 80 [PSH, ACK] Seq=1 Ack=1 Win=515 Len=732 [TCP segment of a reassembled PDU]
7	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=733 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
8	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=2093 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
9	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=3453 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
10	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=4813 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
11	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=6173 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
12	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=7533 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
13	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=8893 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
14	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=10253 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
15	1.771049	10.226.227.88	128.119.245.12	TCP	1414	60075 → 80 [ACK] Seq=11613 Ack=1 Win=515 Len=1360 [TCP segment of a reassembled PDU]
16	1.798605	128.119.245.12	10.226.227.88	TCP	56	80 → 60071 [ACK] Seq=1 Ack=2 Win=237 Len=0
17	1.799616	128.119.245.12	10.226.227.88	TCP	66	80 → 60080 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1360 SACK_PERM=1 WS=128
18	1.799712	10.226.227.88	128.119.245.12	TCP	54	60080 → 80 [ACK] Seq=1 Ack=1 Win=131840 Len=0
19	1.809648	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=733 Win=240 Len=0
20	1.809648	128.119.245.12	10.226.227.88	TCP	56	80 → 60075 [ACK] Seq=1 Ack=3453 Win=283 Len=0

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

$$Average\ throughput\ for\ the\ TCP\ connection = \frac{Total\ amount\ data\ (bits)}{Total\ transmission\ time\ (seconds)}$$

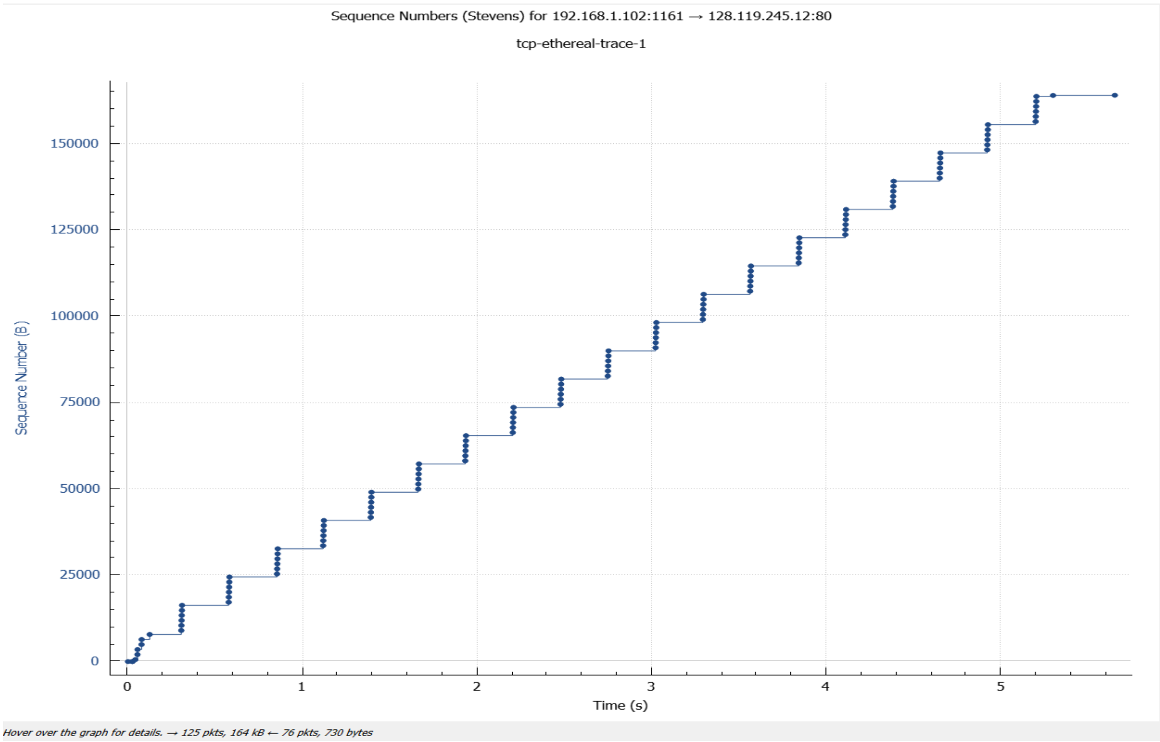
Thus, Total amount of transmitted data = (Sequence # of the first TCP segment) – (ACKed Sequence # of the last ACK) = 153054 – 1 = 153053 bytes.
The whole transmission time = |Time (1st TCP segment) – Time (Last TCP segment) | = |1.798605 – 7.072243| = 5.273638 seconds. Hence, the Throughput = 153053/5.273638 = 29.022KBps

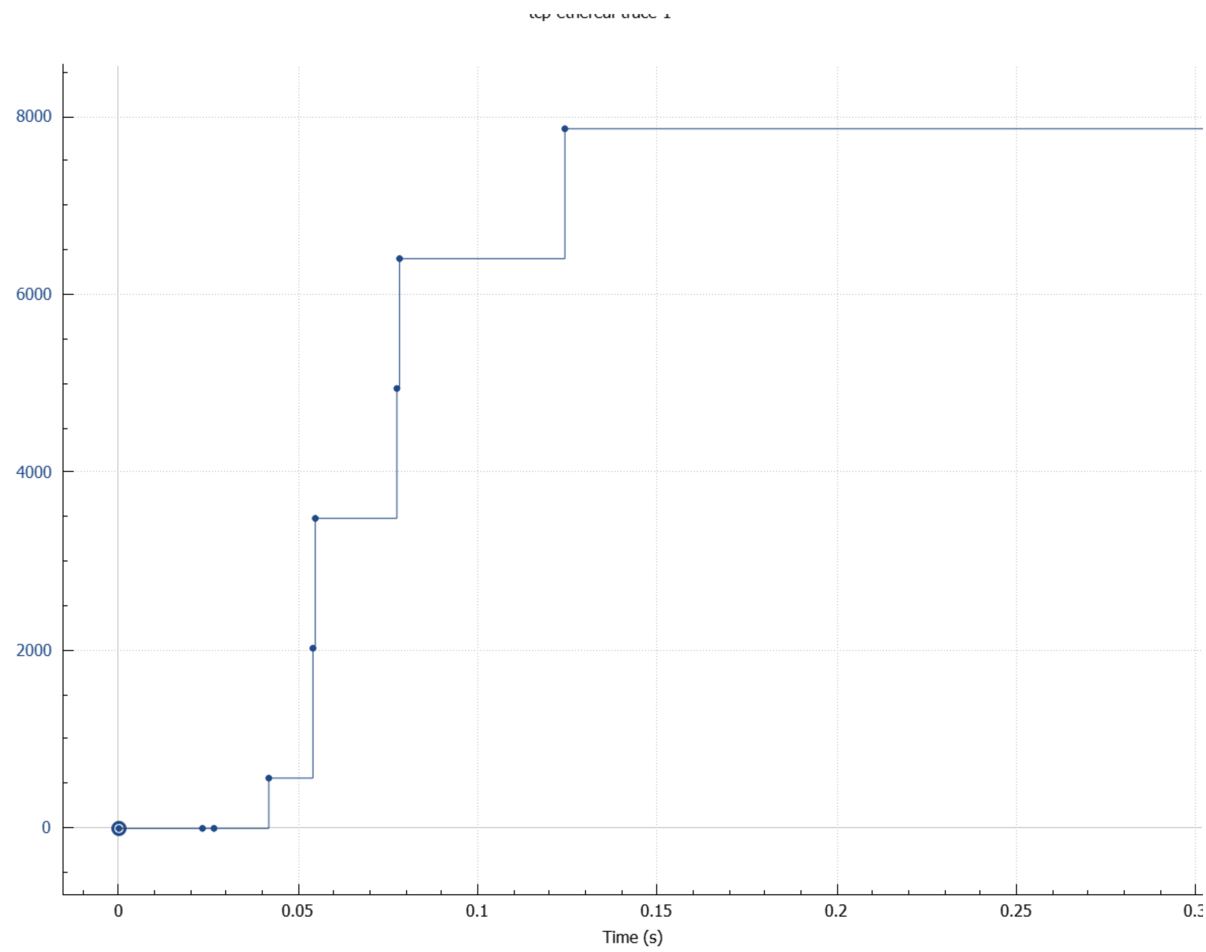




13) Use the Time-Sequence-Graph (Stevens) plotting tool to view the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. Can you identify where TCP's slow start phase begins and ends, and where congestion avoidance takes over? Comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the text. **(Given Trace File)**

TCP's slow start phase begins around zero and ends around 0.15 seconds according to the graph below. Congestion avoidance takes over at around 0.19 seconds because it cut down the amount of data that being sent.





14) Answer each of two questions above for the trace that you have gathered when you transferred a file from your computer to gaia.cs.umass.edu.

3. Conclusion (Discussion of the result & Evaluation of the tool)

In this lab, I have studied and analyzed the performance of TCP Error/Flow Control, Throughput, and Round-Trip Time (RRT). I traced the TCP segments by observing TCP sequence number and acknowledgment number. I also explored and demonstrated how the TCP is capable of providing reliable and dedicated data transmission between my computer and the remote server.

As a packet analyzer, Wireshark are capable of capturing and decoding every packet that are currently-being-transmitted between clients and servers over a real-time network. It also provides practical functionalities such as timing datagram, flow graph, protocols filter, time display formatters, file I/O, and data import/export. On top of that, it's a human-friendly tool for network administrators due to its colorful GUI interface and other interactive built-in statistic toolboxes.