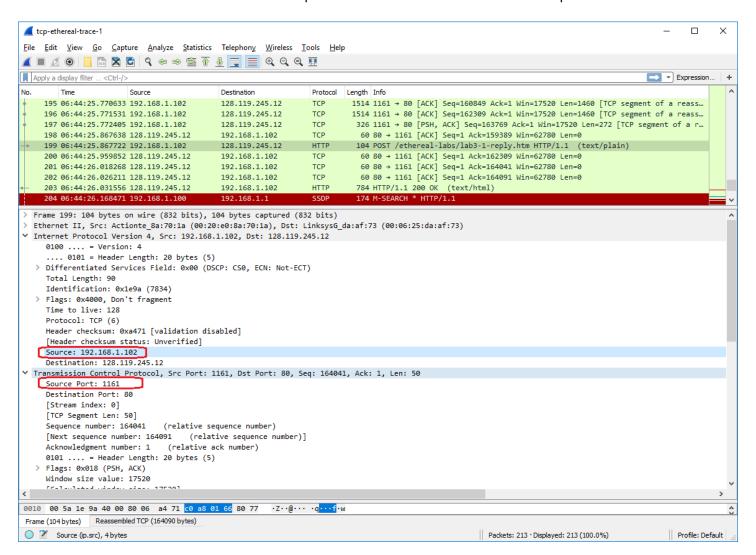
## Lab 3: Wireshark

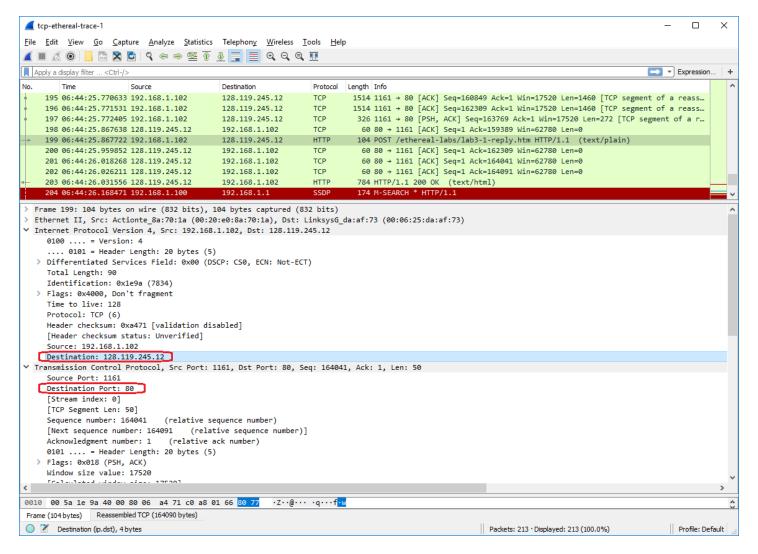
1) What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? To answer this question, it's probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "details of the selected packet header window" (refer to Figure 2 in the "Getting Started with Wireshark" Lab if you're uncertain about the Wireshark windows.

The source IP address for the client computer is 192.168.1.102 and the source TCP port is 1161.



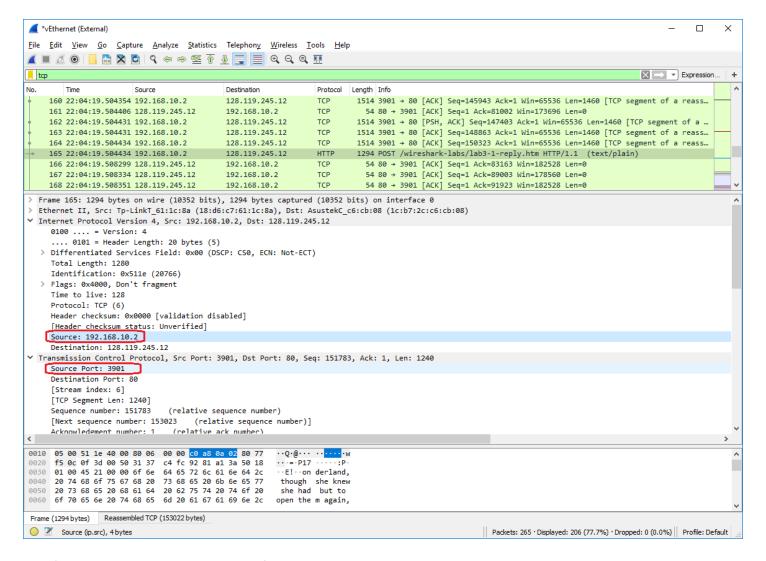
2) What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

The destination IP address for gaia.cs.umass.edu is 128.119.245.12 and the destination TCP port is 80.



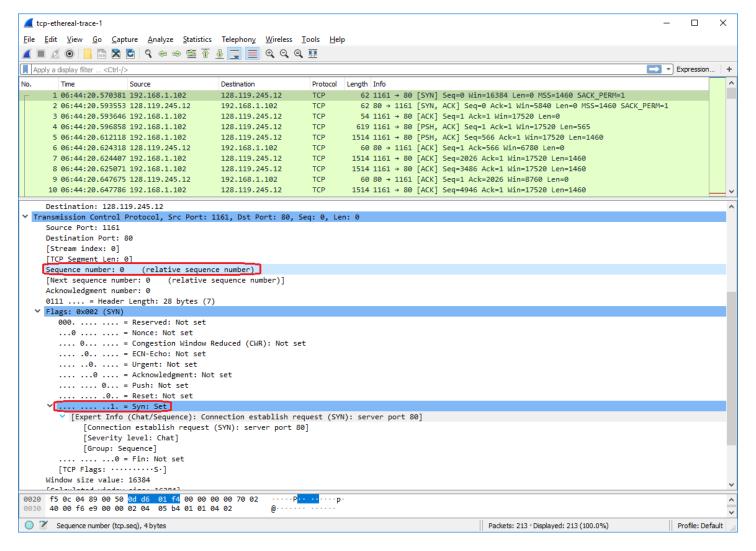
3) If you have been able to create your own trace, answer the following question: What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?

The source IP address for my client computer is 192.168.10.2 and the source TCP port is 3901.



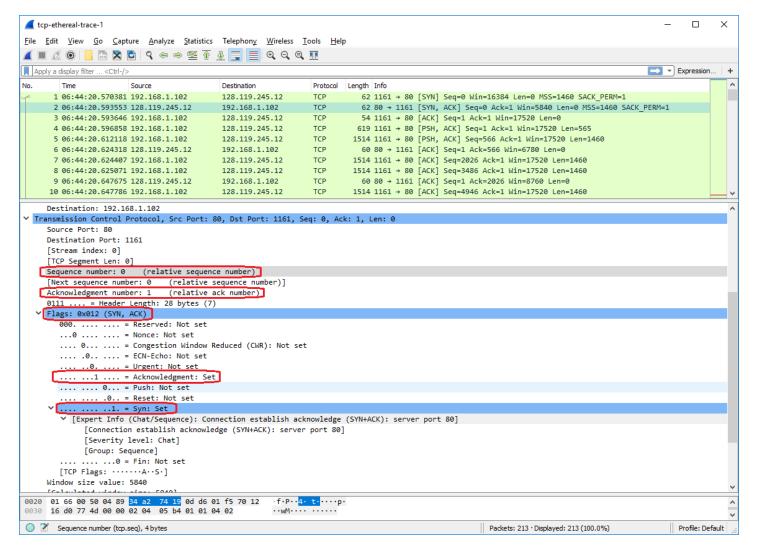
4) What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?

The sequence number of the TCP SYN segment used to initiate the TCP connection is 0. The flags field in the TCP SYN segment has the SYN bit set to 1, which identifies the segment as a SYN segment.



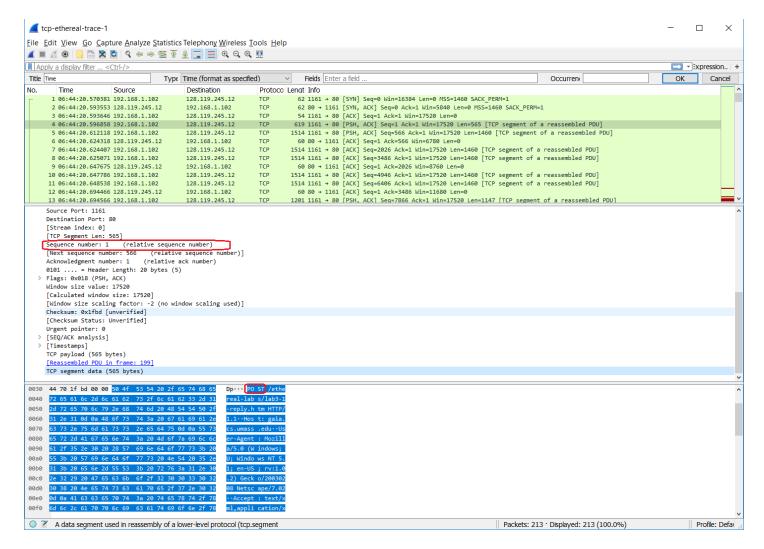
5) What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK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

The sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN is 0. The value of the ACK field in the SYNACK segment is 1. gaia.cs.umass.edu determined the ACK value in the SYNACK segment by incrementing sequence number 0 that was received from the client computer to the ACK value 1. The 10-bit flags field in the TCP SYNACK segment has bit 0 set, which indicates that this is a SYN segment, and bit 3 set, which indicates that this is an ACK segment as well.



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.



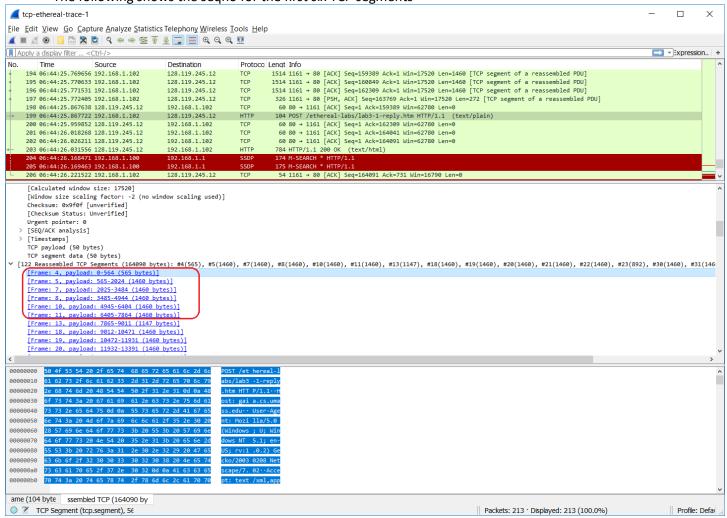
7) Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? 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. You should have a table that looks like this.

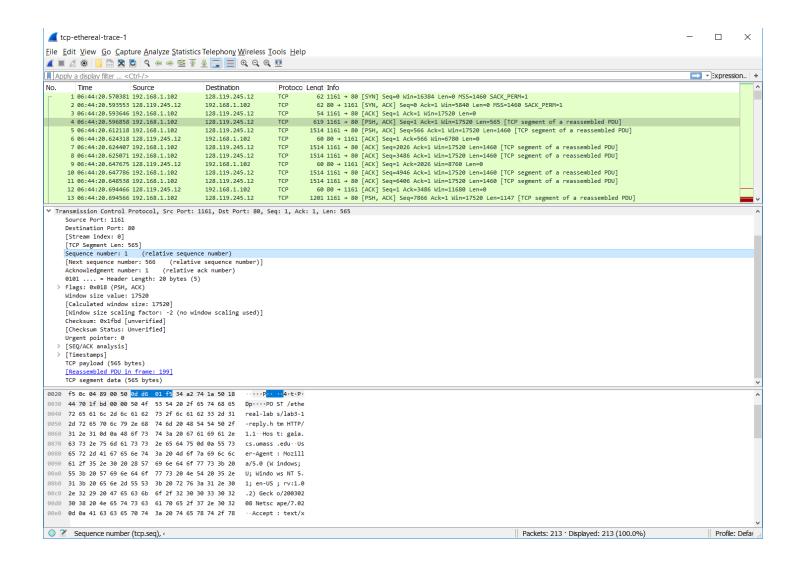
The frame numbers for the first 6 segments in the TCP connection are 4, 5, 7, 8, 10, 11, and the respective sequence numbers of the first six segments in the TCP connection are 1, 566, 2026, 3486, 4946, and 6406. The time for each segment since the capture started, when ACK was received, RTT value, and EstimatedRTT value after ACK receipt is:

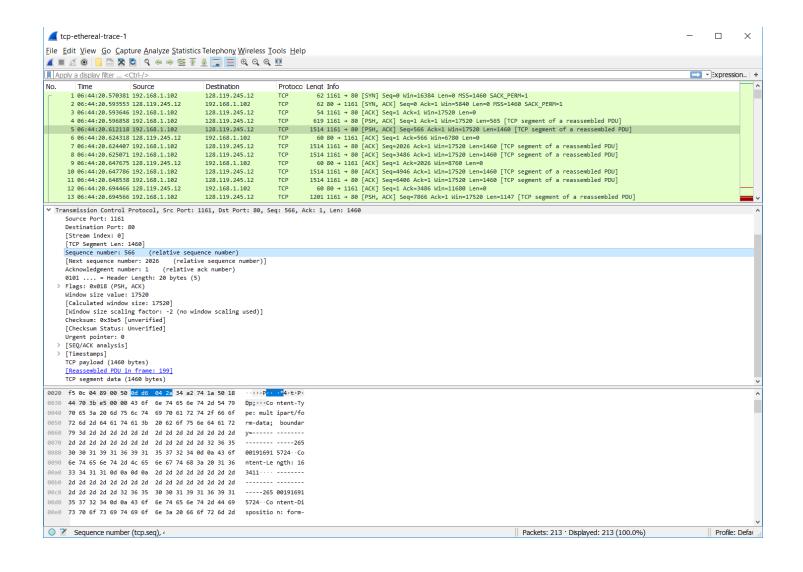
Packet Number #	1	2	3	4	5	6
Time Sent	0.026477	0.041737	0.054026	0.05469	0.077405	0.078157
Time The Ack Received	0.053937	0.077294	0.124085	0.169118	0.217299	0.267802
SampleRTT	0.02746	0.035557	0.070059	0.114428	0.139894	0.189645
EstimatedRTT	0.02746	0.028472	0.03367	0.043765	0.055781	0.072514

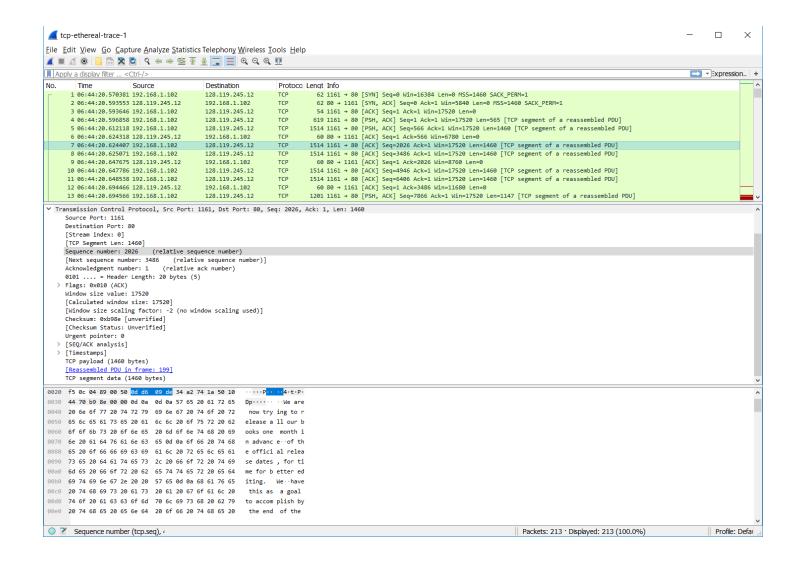
EstimatedRTT =  $(1 - \alpha)$  · EstimatedRTT +  $\alpha$  · SampleRTT

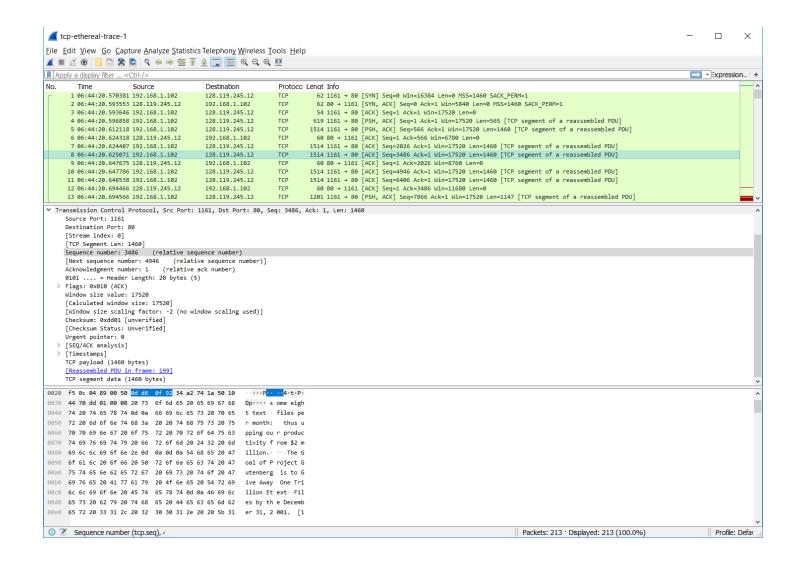
The following shows the seqno for the first six TCP segments -

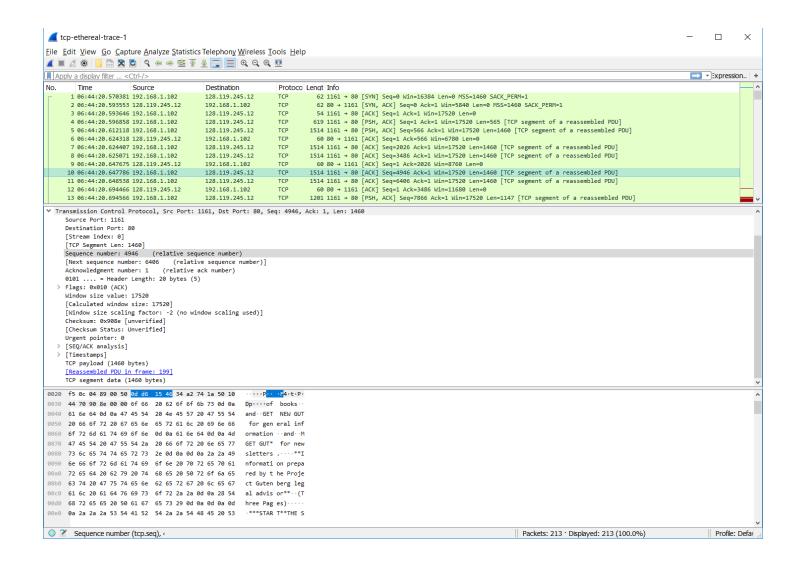


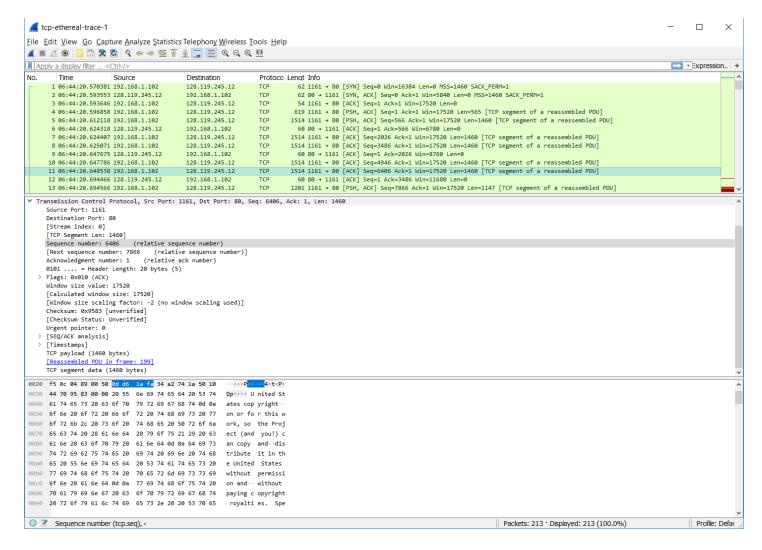




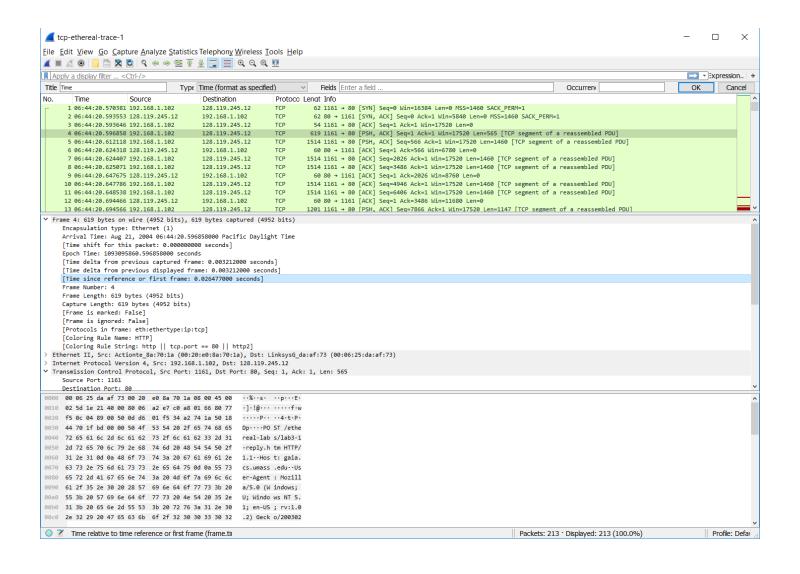


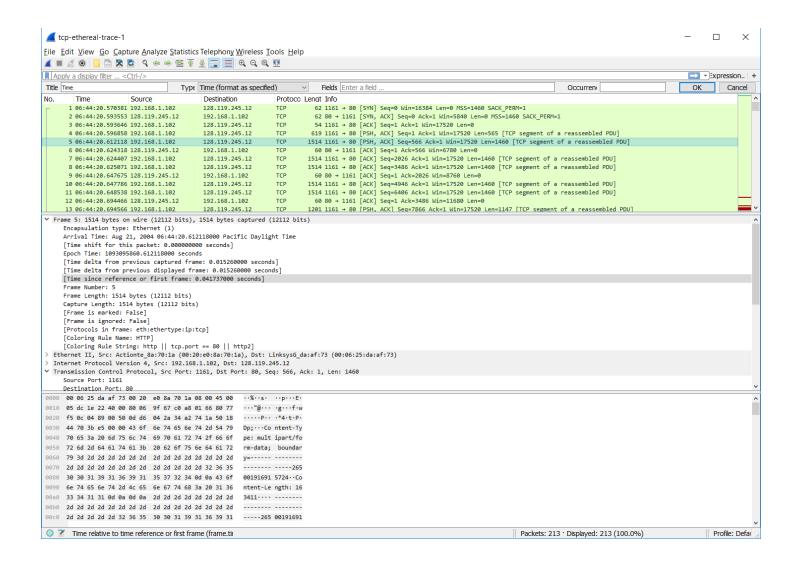


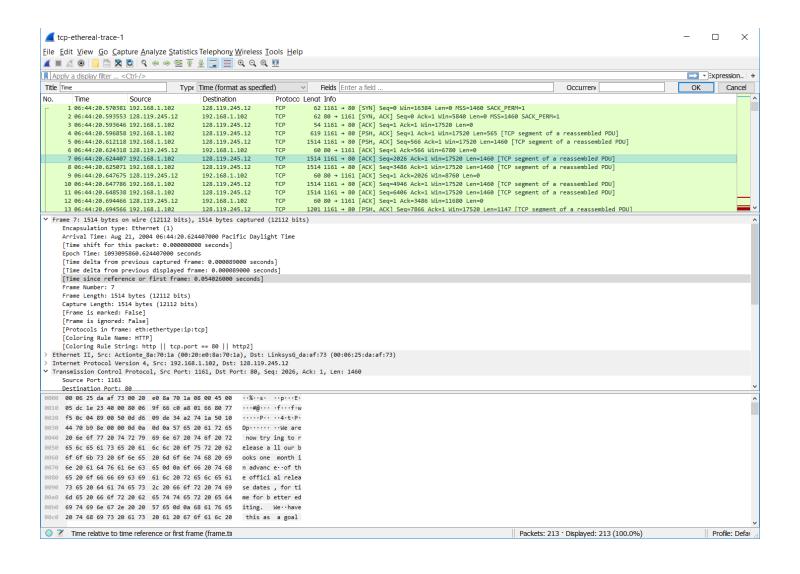


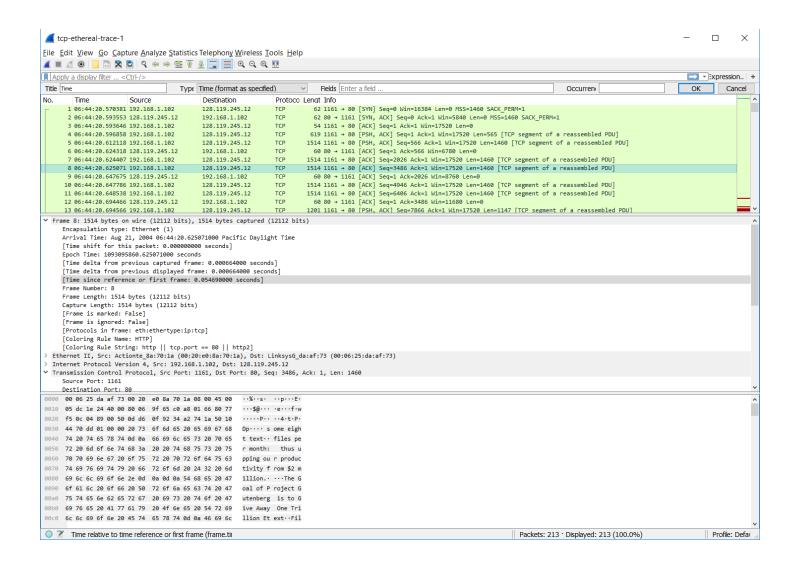


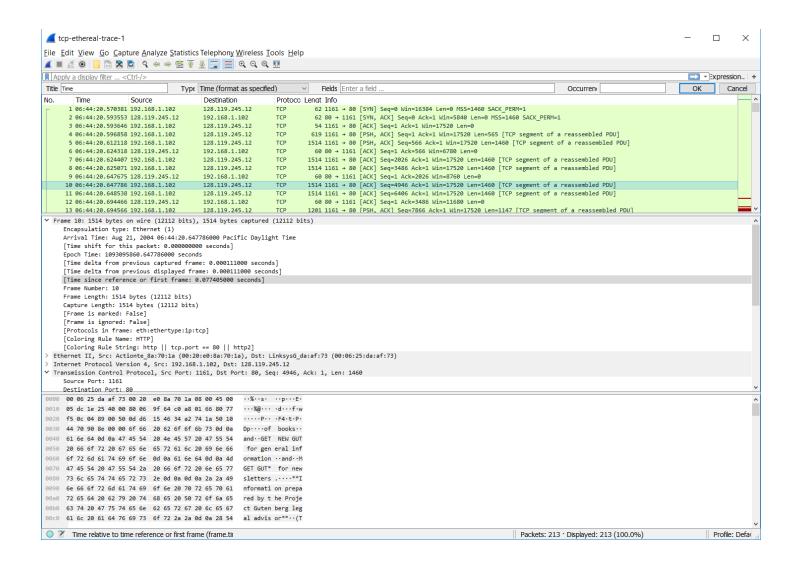
The following shows the time delta since the capture started for the first 6 TCP segments –

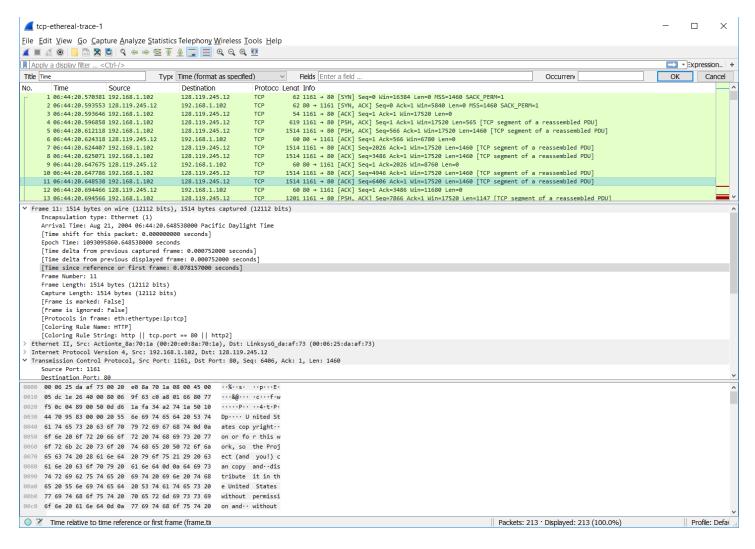




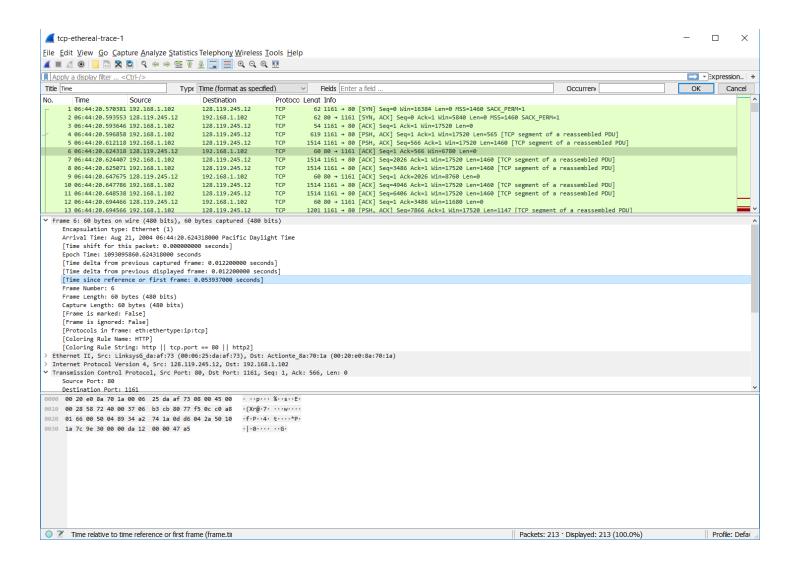


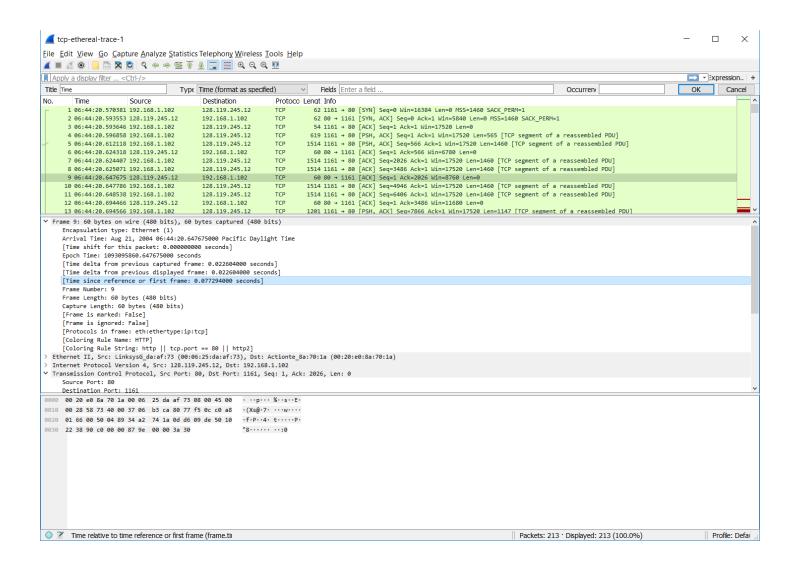


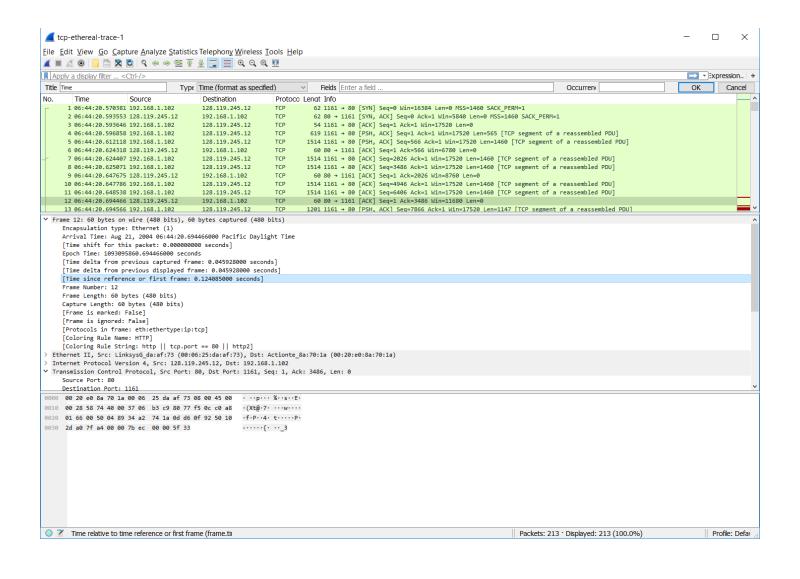


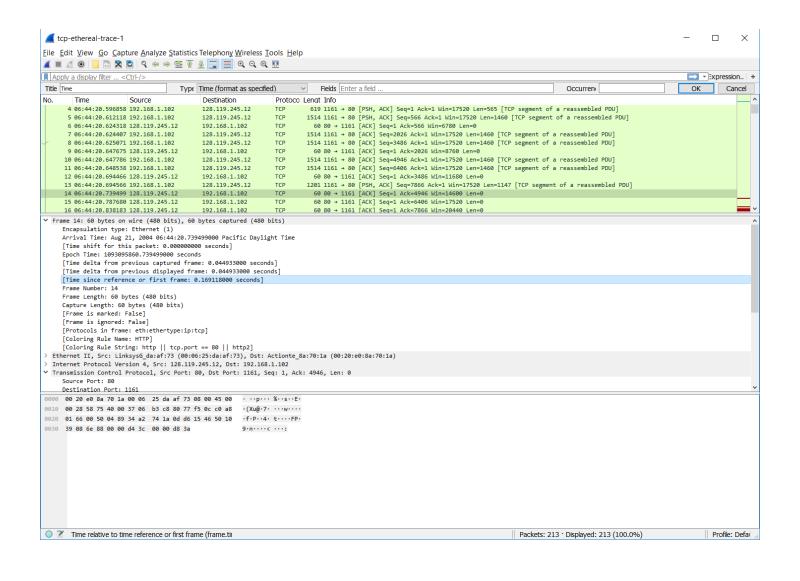


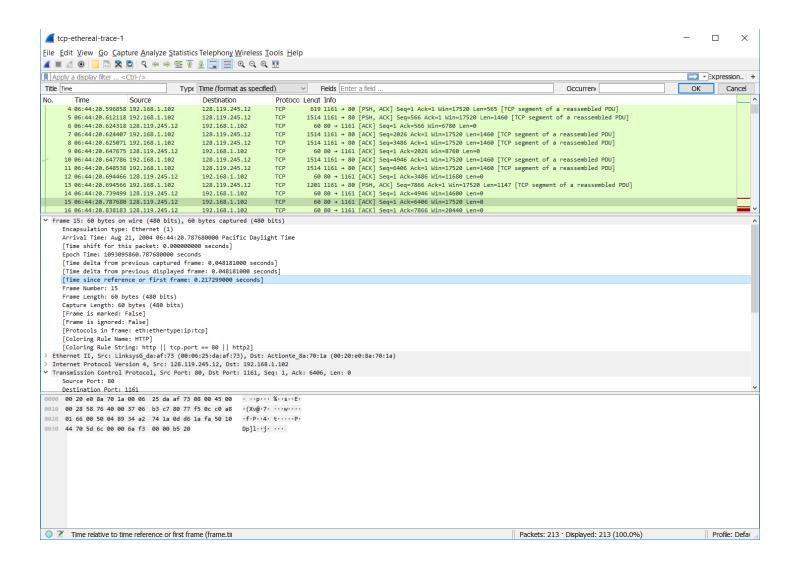
The following shows the time when the ACK was received from the server to client for the first 6 segments -

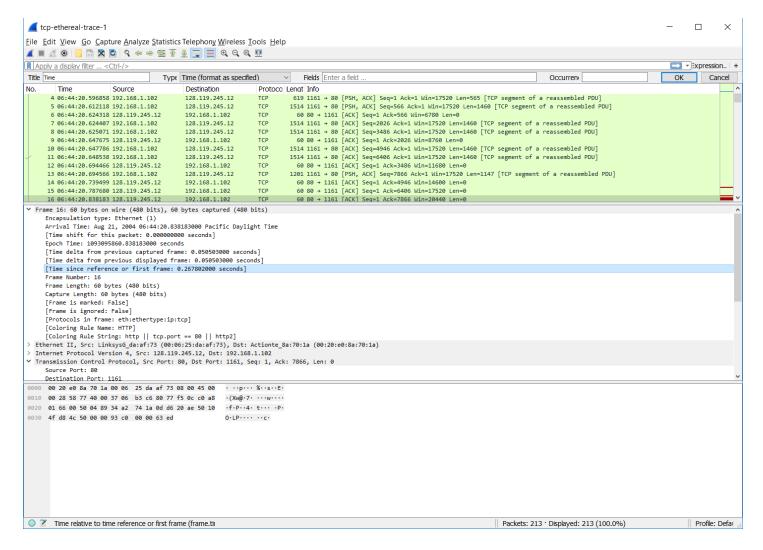












8) What is the length of each of the first six TCP segments?

The length of each of the first six TCP segments are:

Segment 1 – 565 bytes

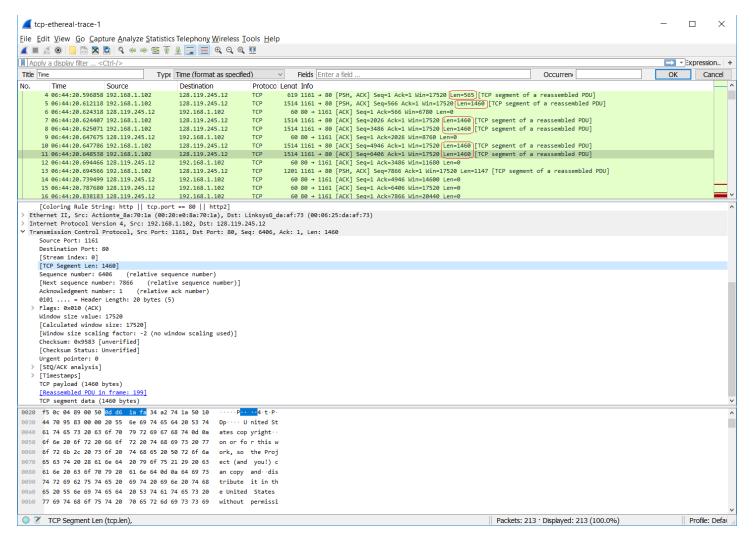
Segment 2 - 1460 bytes

Segment 3 – 1460 bytes

Segment 4 – 1460 bytes

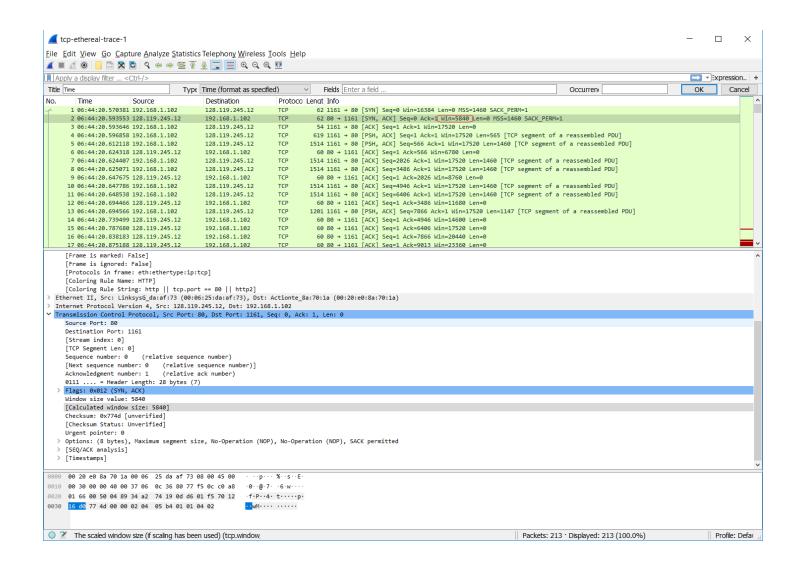
Segment 5 - 1460 bytes

Segment 6 – 1460 bytes



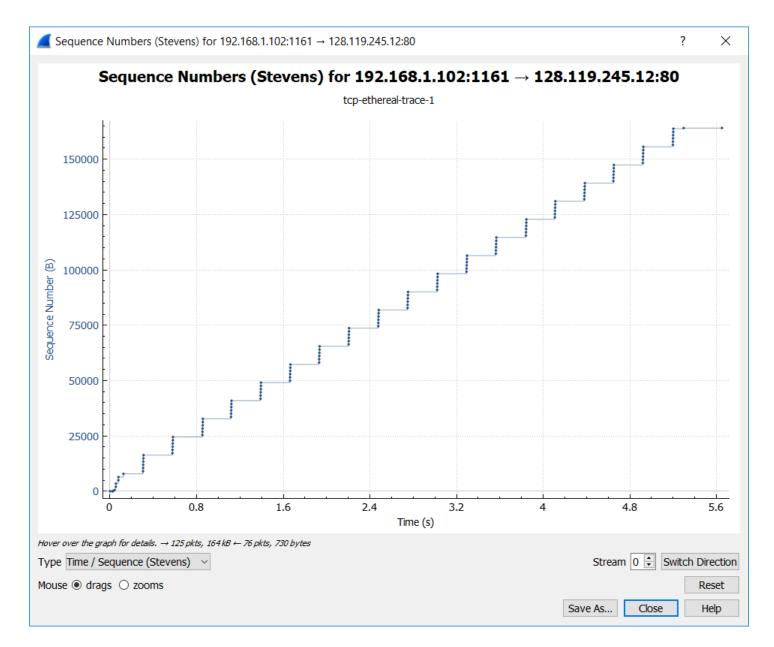
9) What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

The minimum amount of available buffer space advertised at the received for the entire trace is a window size of 5840 bytes. No, the lack of receiver buffer space never throttles the sender, which can be observed by viewing the window size for each of the packets in the RTT Graph shown below. In the RTT graph below, the window size never drops after reaching a maximum window size of 62780 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?

No, there are no retransmitted segments in the trace file. I verified this by manually viewing all of the sequence numbers in each of the TCP packets sent from the client to server. If there was a retransmission, then the client would send a TCP packet with a sequence number that is less than any previous TCP segments' sequence numbers. I also verified this by observing that the Sequence Numbers graph below never decreases, which verifies that the sequence number transmitted from client to server never decreases.



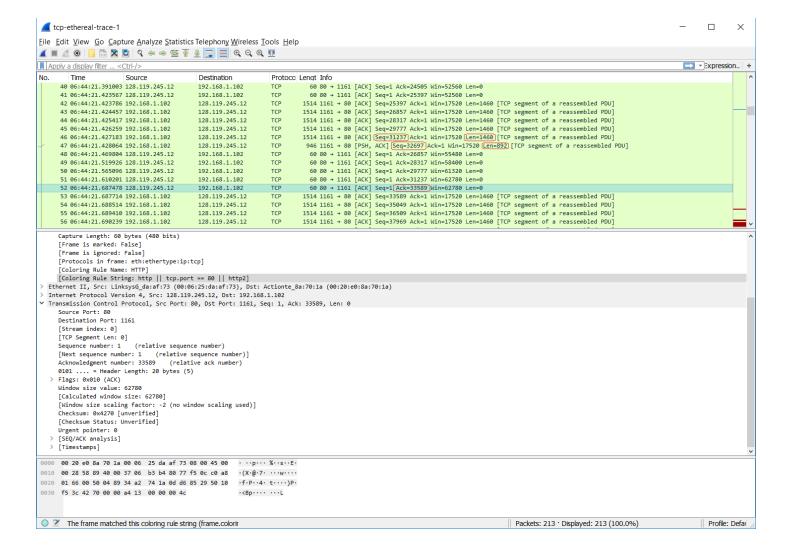
11) How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 250 in the text).

The receiver typically acknowledges 1460 bytes in an ACK. Since the ACK value is the number of bytes received since the last ACK, I determined the amount of acknowledged data by calculating the difference  $ACK_n - ACK_{n-1}$ , whereas the first ACK is just 1 based on a typical 3-way handshake.

One case where the receiver is ACKing every other segments is TCP Frame No 52, which acknowledges 2352 (= 1460 + 892) bytes. In this case, client sent 1460 bytes in TCP frame no 46 and 892 bytes in TCP frame no 47. When TCP frame 52 arrived, the server received an in-order segment with the expected sequence number and all data up to the expected sequence number were already acknowledged, so the server performed a delayed ACK that lasted 77.277 ms (= (1.117097000 - 1.039820000) \* 1000) but could have waited up to 500 ms. TCP frame 52 is ACKing the TCP frame no 46 and 47, so TCP segment 52 is ACKing every other segment in this case.

TCP Frame No	ACK value	ACK data
3	1	1
6	566	565
9	2026	1460

1		,
12	3486	1460
14	4946	1460
15	6406	1460
16	7866	1460
17	9013	1147
24	10473	1460
25	11933	1460
26	13393	1460
27	14853	1460
28	16313	1460
29	17205	892
36	18665	1460
37	20125	1460
38	21585	1460
39	23045	1460
40	24505	1460
41	25397	892
48	26857	1460
49	28317	1460
50	29777	1460
51	31237	1460
<mark>52</mark>	<mark>33589</mark>	<mark>2352</mark>



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

If the throughput for the TCP connection is bytes transferred per unit of time, then throughput can be calculated as throughput\_rate = (total\_length) / (time\_elapsed). Total length would be calculated as total\_length =  $TCP_{segmentN\_ack} - TCP_{segment1\_seqno}$ , where  $TCP_{segmentN\_ack}$  is the ACK value of the last TCP segment sent by the server minus the sequence number of the first TCP segment sent by the client. The time elapsed would be calculated as time\_elapsed =  $TCP_{segmentN\_time} - TCP_{segment1\_time}$ , where  $TCP_{segmentN\_time}$  is the time elapsed since the capture first started for the last TCP segment and  $TCP_{segment1\_time}$  is the time elapsed since the capture first TCP segment. Therefore, the throughput is 30222.75 bytes per second.

```
throughput_rate = (TCP_{segmentN\_ack} - TCP_{segment1\_seqno}) / (TCP_{segmentN\_time} - TCP_{segment1\_time}) throughput_rate = (164091 - 1) bytes / (5.455830000 - 0.026477000) seconds

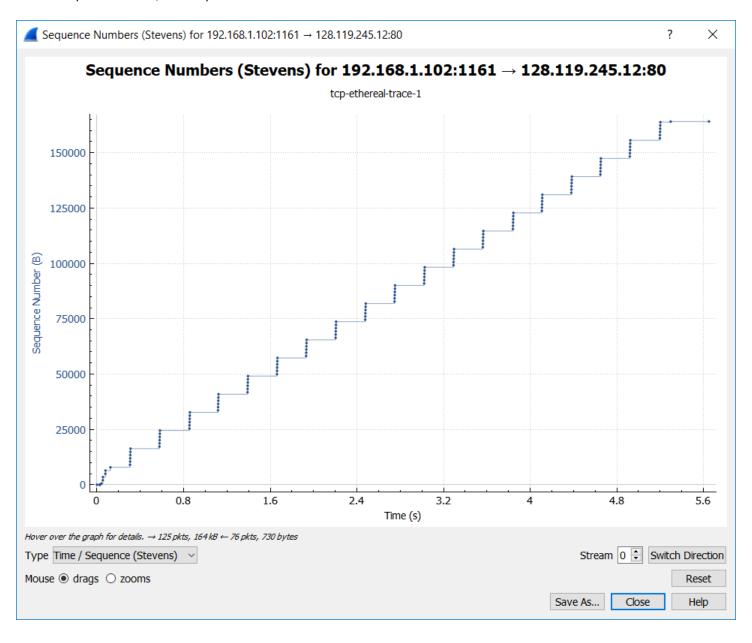
Total_length = 164090 bytes

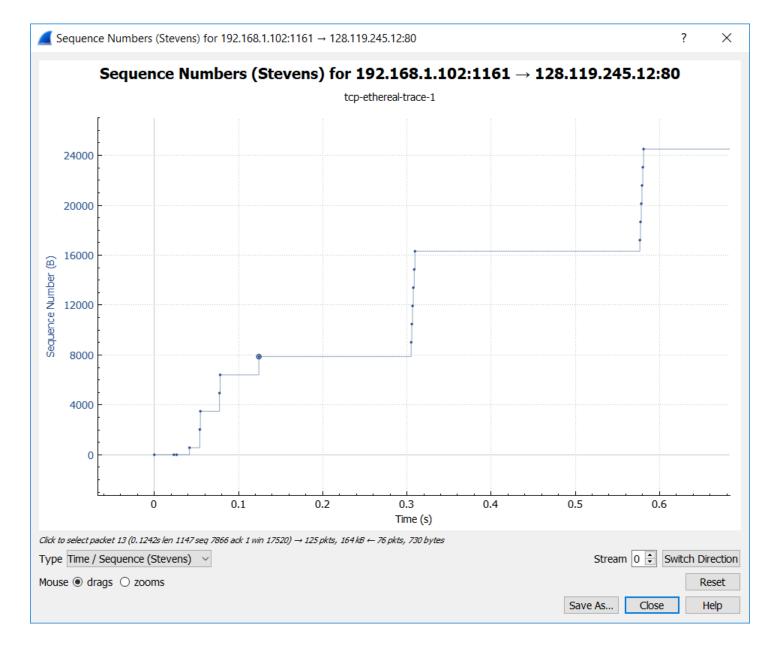
Time_elapsed = 5.429353 seconds

throughput_rate = 164090 bits / 5.429353 s
= 30222.753981920129341378245253164 bytes per second
= 241782.03185536103473102596202531 bps
= 241.78203185536103473102596202531 Mbps
```

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 slowstart 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.

The TCP's slowstart phase begins after sending the first TCP segment at 0.02648s and ends after sending packet 13 around 0.1242s. Congestion avoidance takes over around 0.3s where we see 5 TCP packets transmitted. The measured data differs from the ideal behavior of TCP that is in the text because we do not observe the typical exponential increase in CWND that's described in the text. The first 8 packets look as if slow-start is just beginning to increase the CWND at 0.0s to 0.15s. After this point in time, the measured data transmits in batches of 6 packets even though the window size in the ACK packets are typically much greater than the number of packets transmitted. This is unlike the expected method of transmission described in the text where slow-start CWND increases exponentially until an event occurs that triggers moving to congestion avoidance mode. These slow-start to congestion avoidance event triggers are a loss event indicated a timeout, CWND equals ssthresh, or 3 duplicate ACKs are received.





14) Answer Question 13 for the trace that you captured when you transferred a file from your *own* computer to gaia.cs.umass.edu.

The TCP's slowstart phase begins after sending the first TCP segment at 0.10s and ends after sending packet 13 around 0.33s. Congestion avoidance takes over at this point after 0.33s. The measured data is more in line with the ideal behavior of TCP that is in the text because we observed the exponential increase in in MSS until CWND equals the SSTHRESH around 0.33s. At around 0.10s, 8 TCP packets are transmitted. At around 0.22s, 16 TCP packets are transmitted. At around 0.32s, 32 TCP packets are transmitted. This doubling in CWND size matches what is described in the text.