Lachlan Sinclair

11/6/2019

CS 372_400

Lab 3:

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.

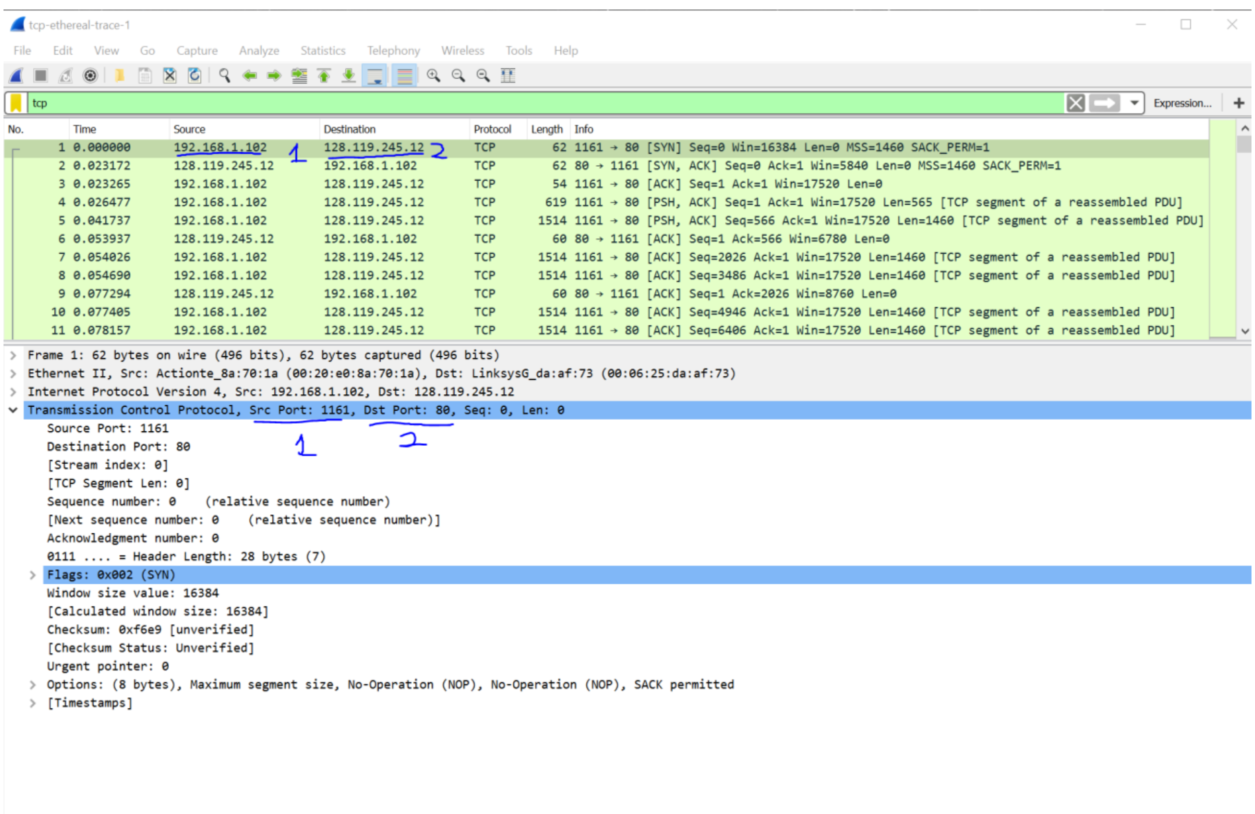
IP addresses of the client: 192.168.1.102

Port number used by the client: 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 addresses of gaia.cs.umass.edu: 128.119.245.12

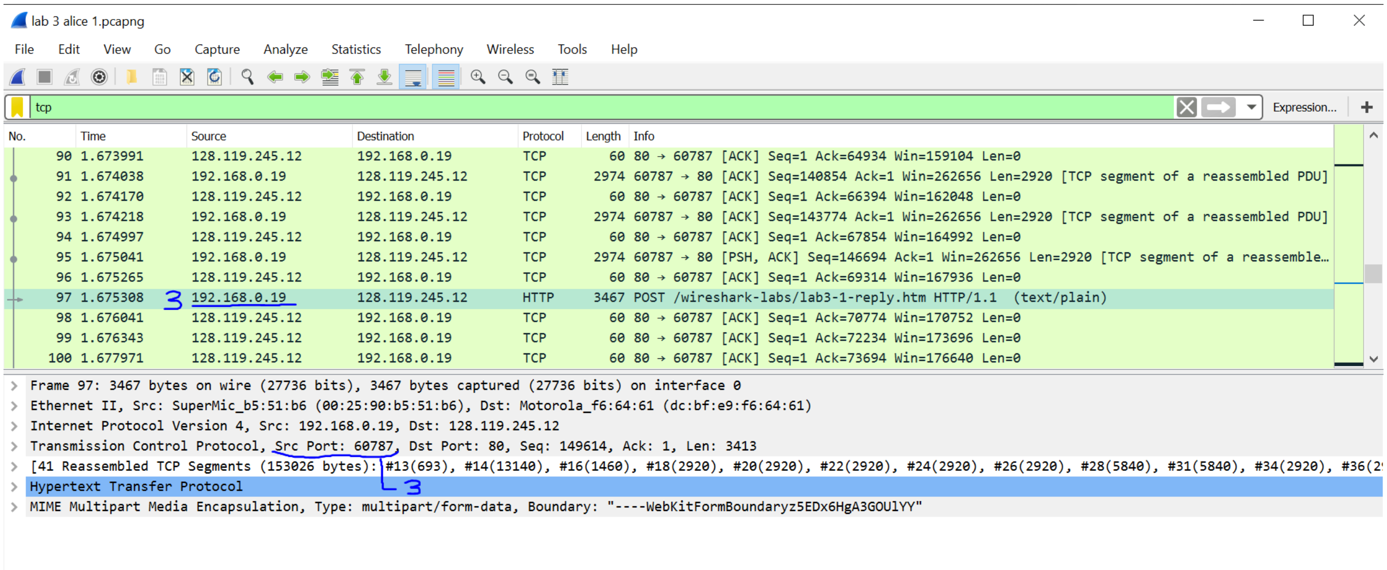
Port number gaia.cs.umass.edu is using: 80



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: 192.168.0.19

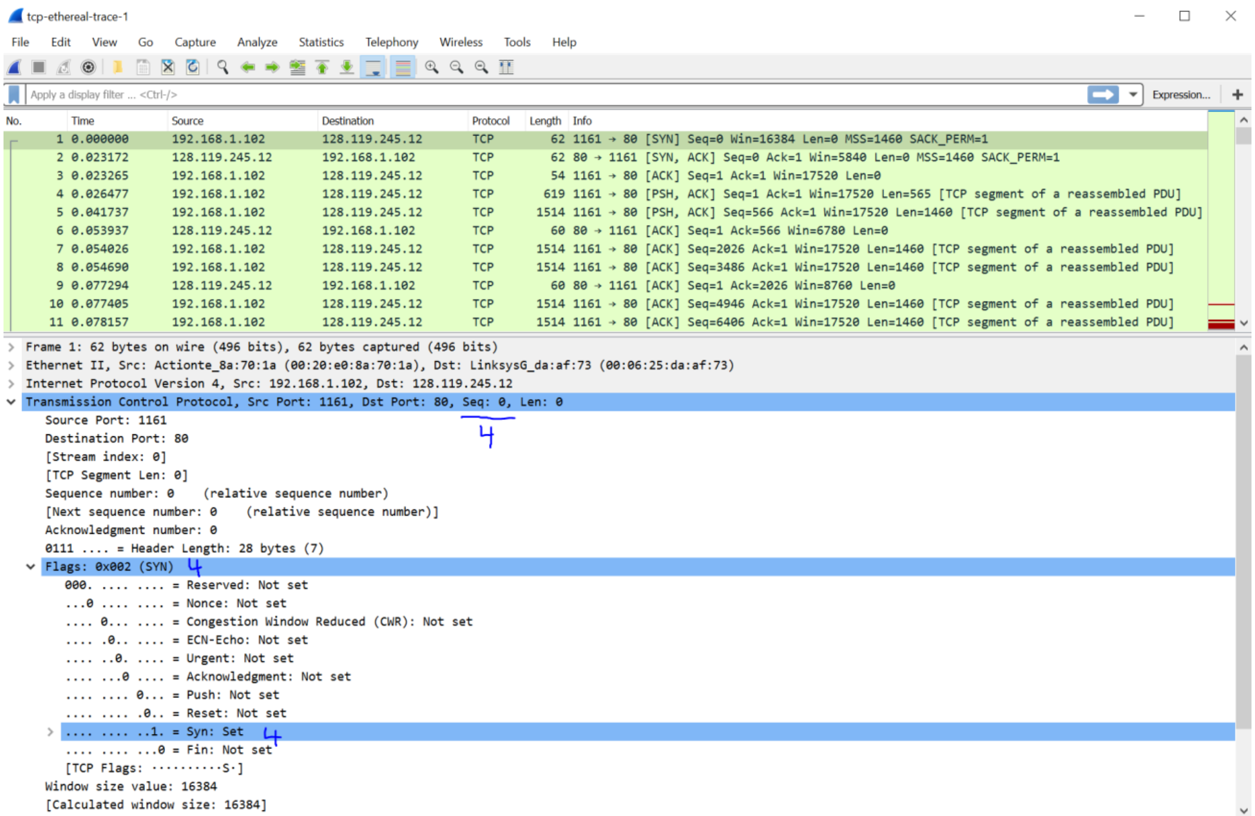
Client TCP port number: 60787



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?

TCP SYN Sequence number: 0

The SYN flag (0x002) being set identifies it as the SYN segment.



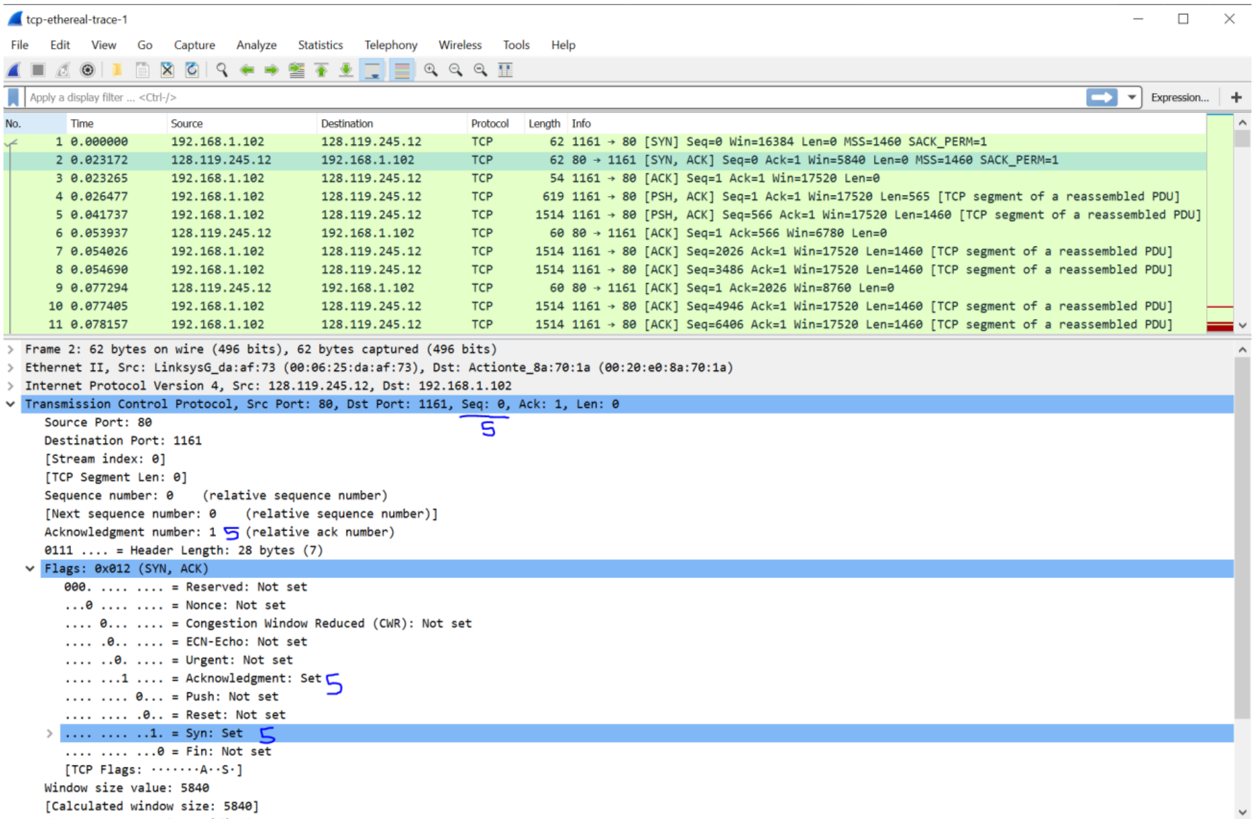
5. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK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

The sequence number of the SYNACK segment is 0.

The acknowledgement field is set to 1.

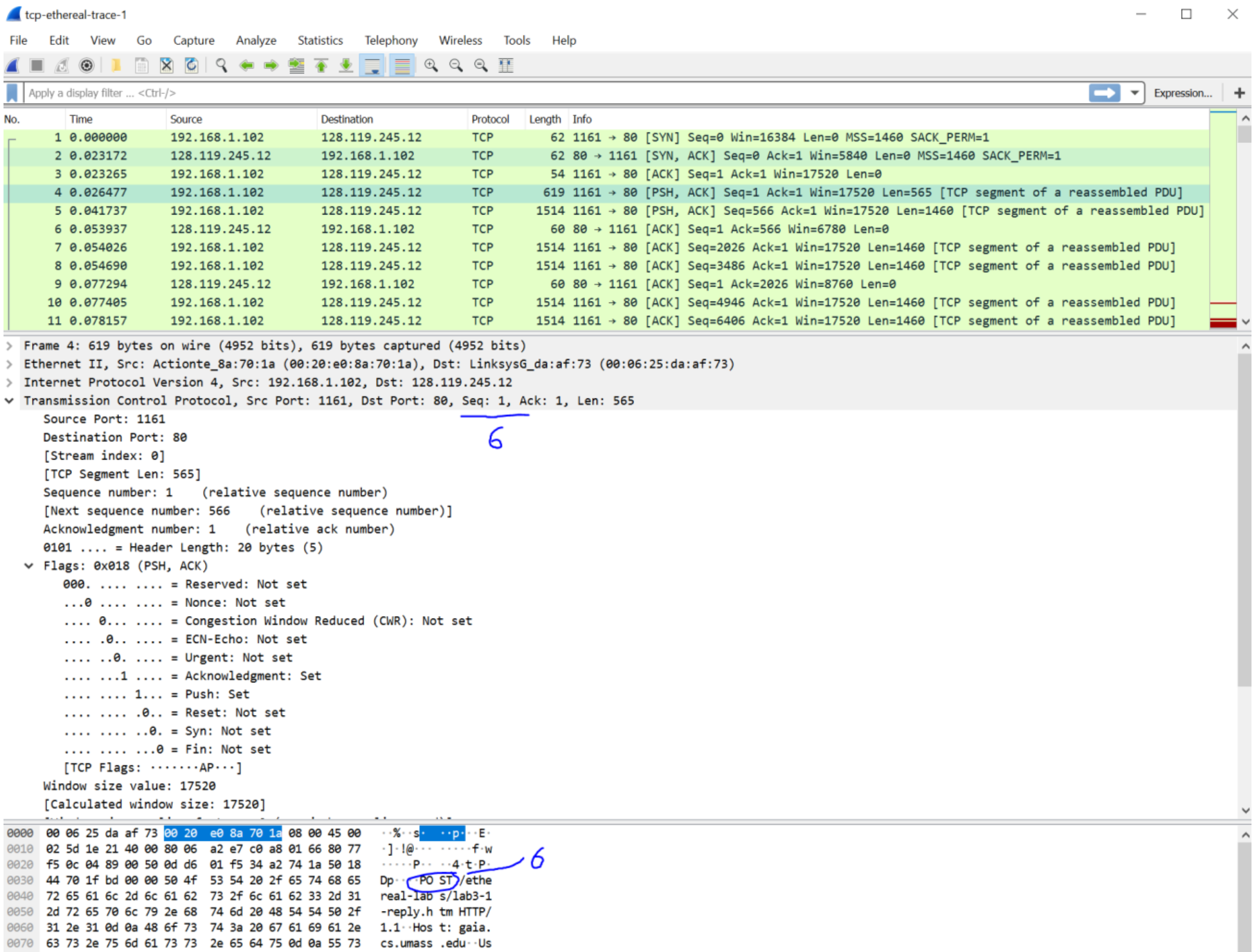
gaia.cs.umass.edu determined that sequence number after successfully receiving the SYN segment, since its sequence number was 0 on the SYN segment it returned 1 since it is the next expected sequence number.

This segment is identified as the SYNACK segment by the SYN and ACK flags being set (0x012)



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 on the TCP segment with the post command is 1.

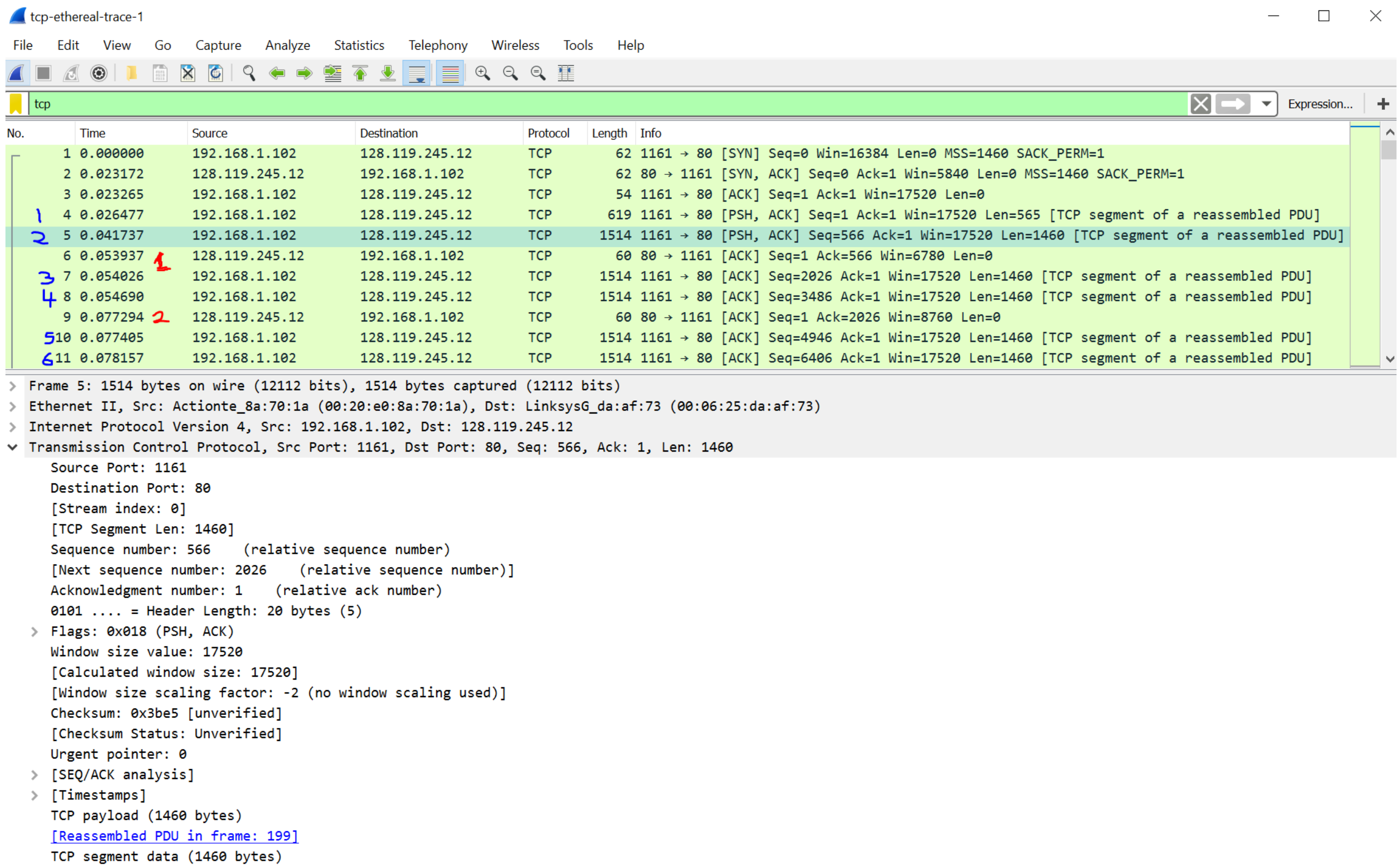


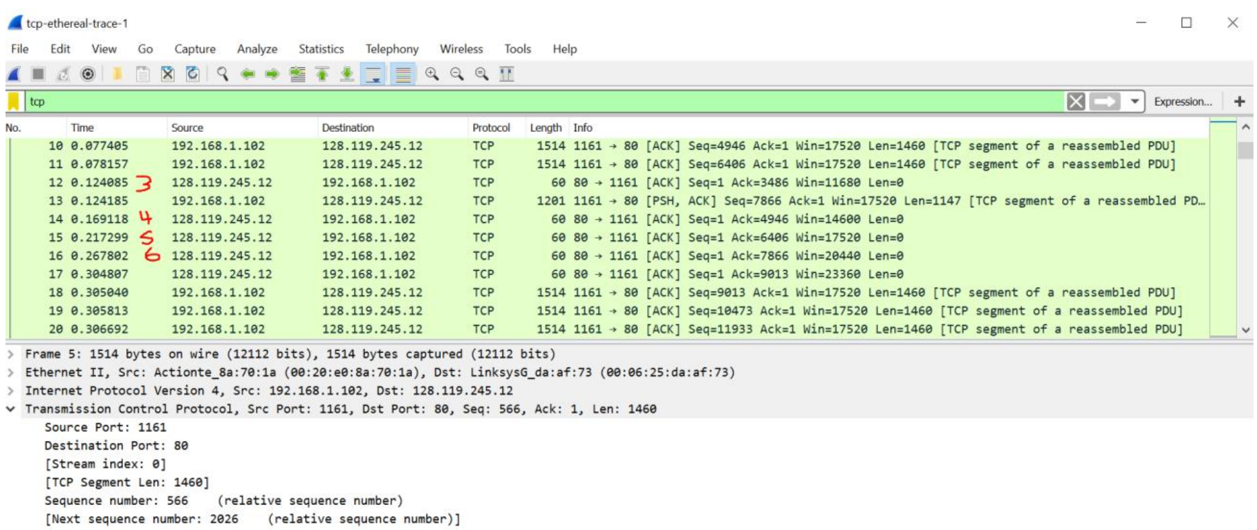
7. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received?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

All times are in seconds (calculated in excel):

Packet #	1	2	3	4	5	6
Sequence #	1	566	2026	3486	4946	6406
Time Sent	.026477	.041737	.054026	.054690	.077405	.078157
Time the Ack Received	.053937	.077294	.124085	.169118	.217299	.267802
Sample RTT	0.02746	0.035557	0.070059	0.114428	0.139894	0.189645
Estimated RTT	0.02746	0.02847213	0.03367	0.043765	0.055781	0.072514

For segment one, the estimated RTT equal the sample RTT. For the rest I used and alpha value of .125 in the formula estimated RTT = (.875*estimatedRTT(n-1)) + (.125*sampleRTT (n))





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

Segment 1: 565 bytes

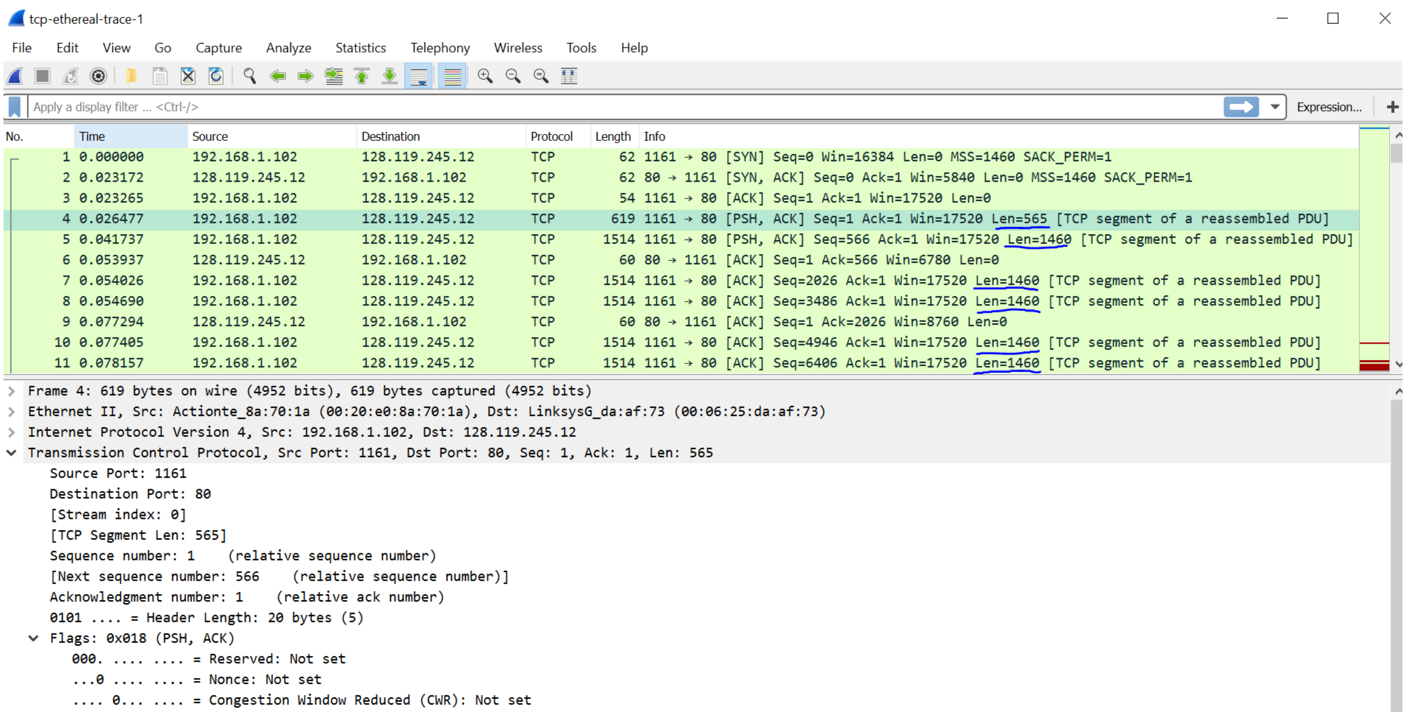
Segment 2: 1460 bytes

Segment 3: 1460 bytes

Segment 4: 1460 bytes

Segment 5: 1460 bytes

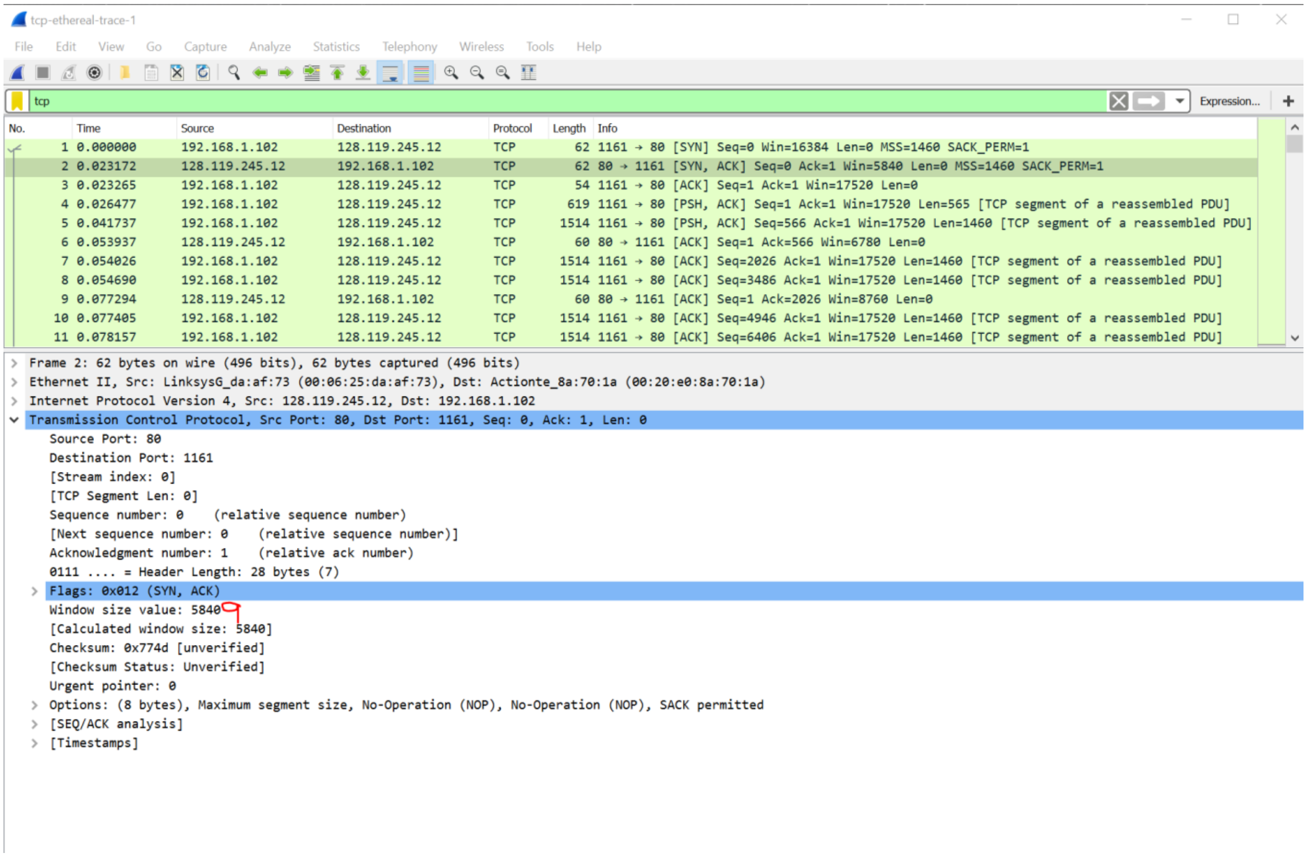
Segment 6: 1460 bytes

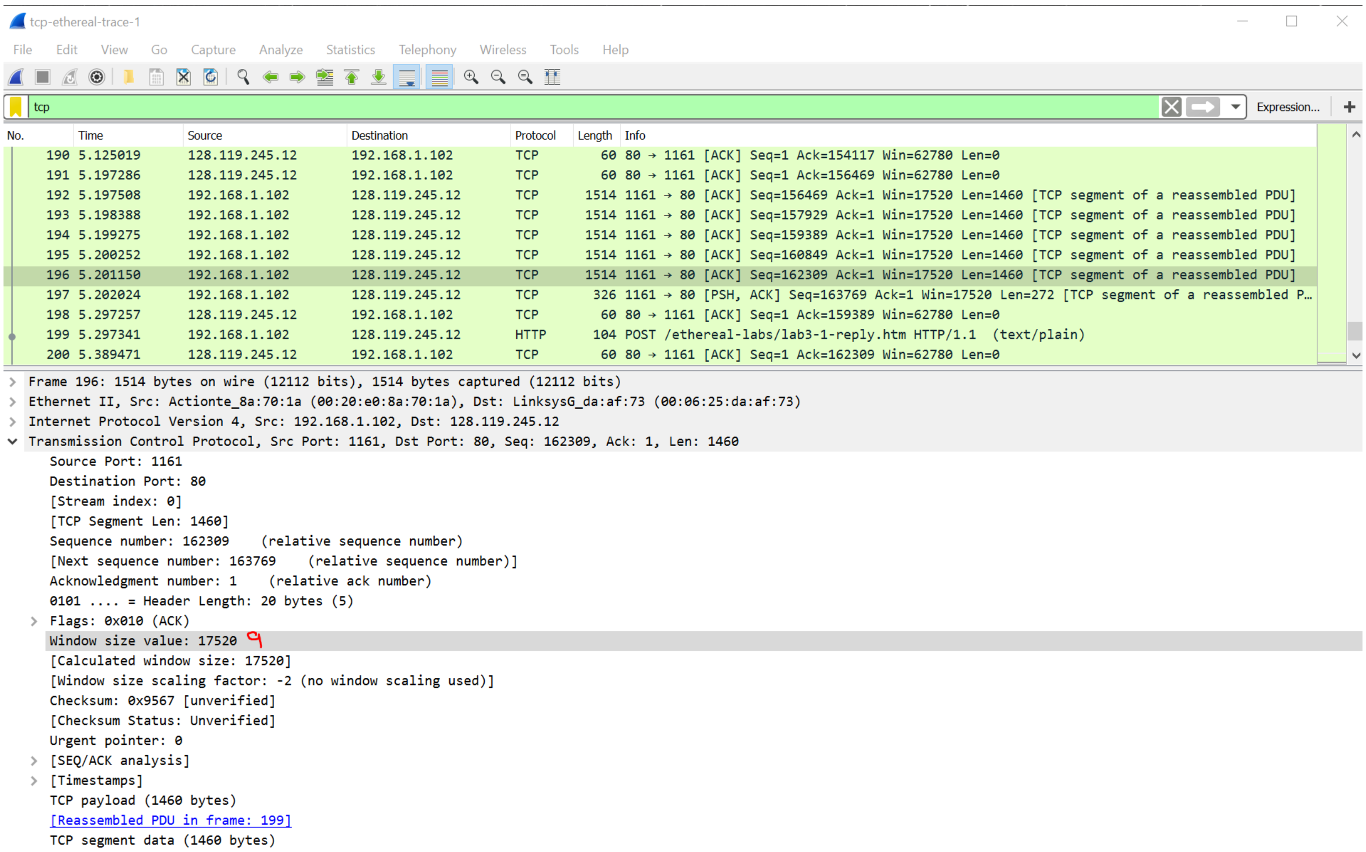


9. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

Looking at the first acknowledgement from the server you can see the window size is listed as 5840 bytes.

Looking through the acknowledgements the window size continually increased, and therefor the server was not throttling the sender. See the screen shot below of one of the last acknowledgements, it has a window size of 17520 bytes.



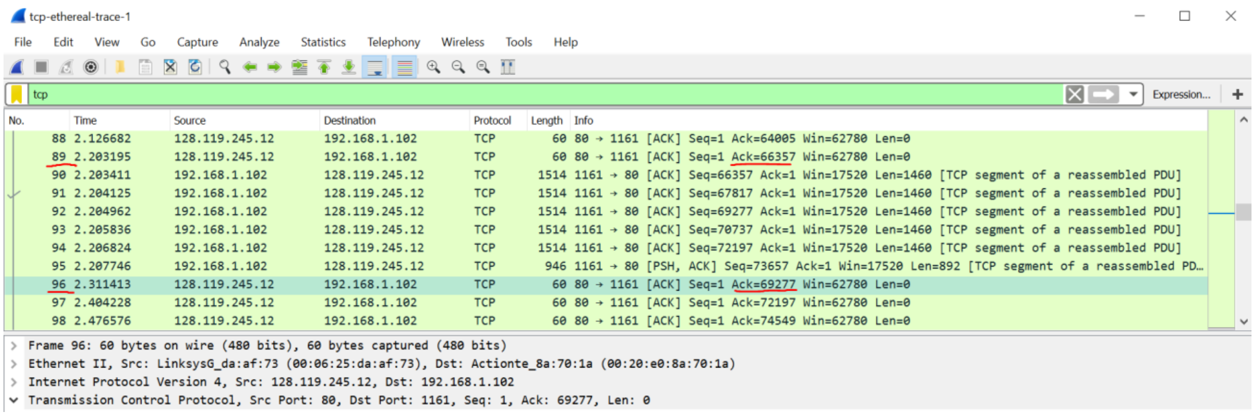


10. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?

There are no retransmissions in the trace files. I checked the sequence numbers used in the sender's packets, the sequence numbers always increased so nothing had to be resent. Also, I have previously used Wireshark and retransmissions show up as super noticeable black entries that are even visible on the scroll bar. The graph in problem 13 also confirms this, since the sequence number always increases.

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 acks for the most part mirror the 1460 byte size of the segments being sent. In quite a few places there is an ack that is acking every other received segment, there sizes are intervals of 1460 bytes, like in the example below 2920 bytes were acked at the same time.



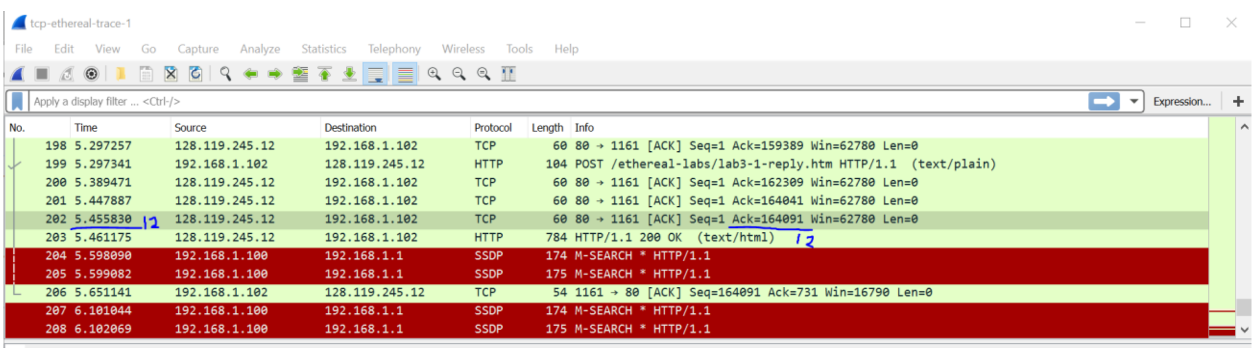
12. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

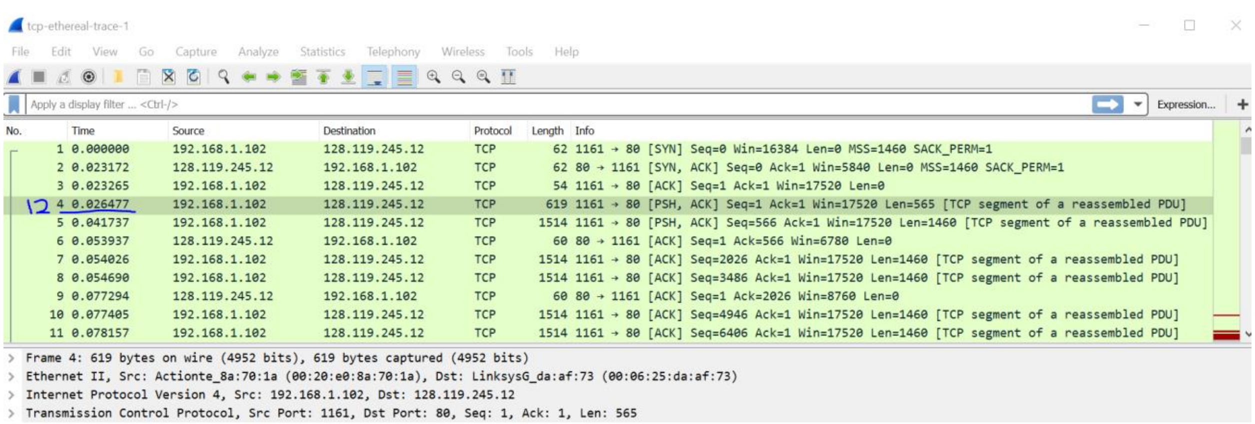
The last ack from the server had an ack value of 164091, which means it had received a total of 164091-1=164090 bytes. This last ack was received at time 5.455830. The first packet containing data was sent at time .026477.

Total time of transmission = 5.455830-.026477=5.429353 seconds

Now divide the total number of bytes transferred by the amount of time it took to get the throughput.

Throughput = 164090/5.429353=30222.75398 Bytes/second

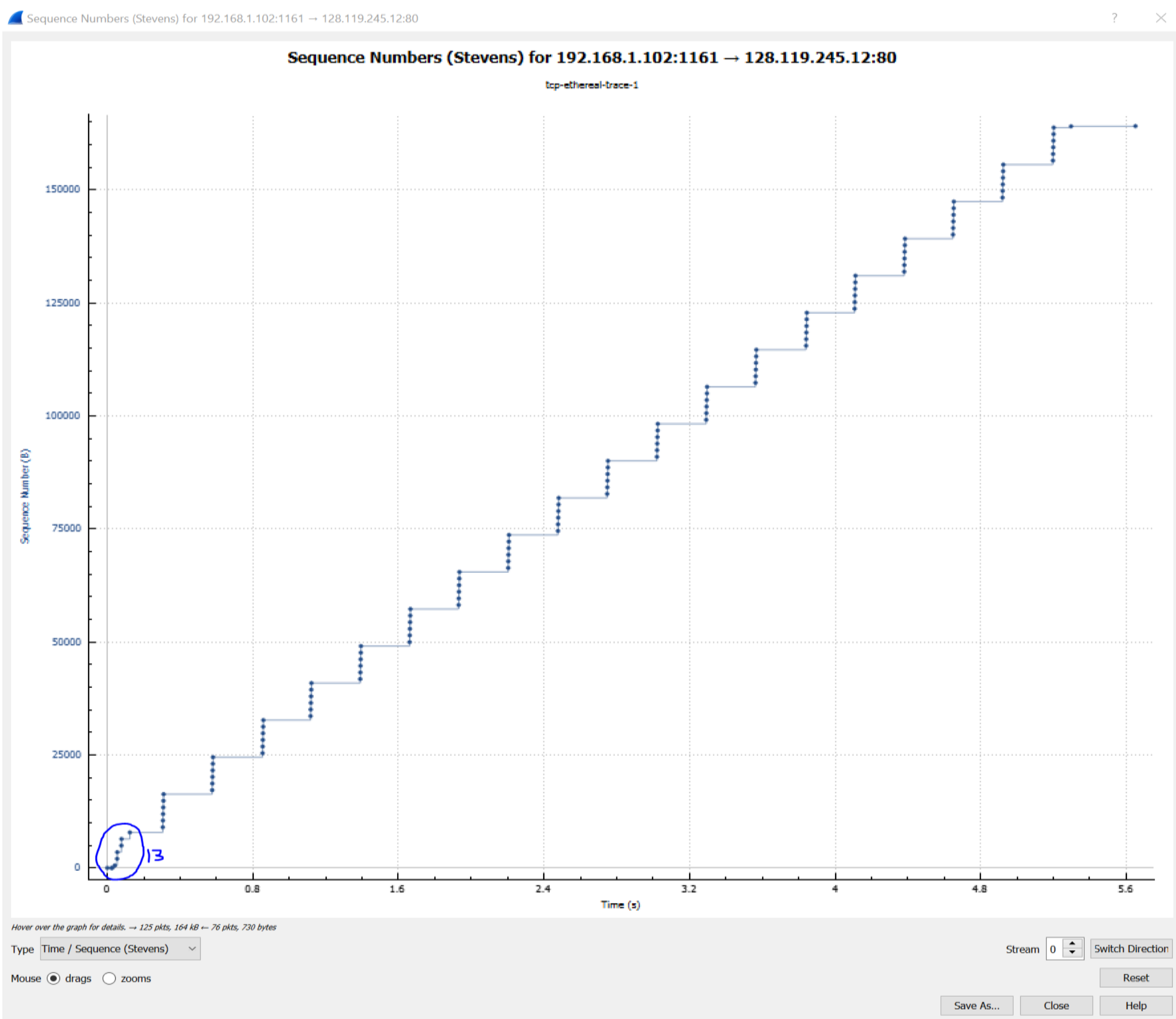




13. Use the Time-Sequence-Graph(Stevens) plotting tool to view the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. Can you identify where TCP's slowstart phase begins and ends, and where congestion avoidance takes over? Comment on ways in which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

By hovering over the values in the graph the slow start begins around .02648 seconds in and ends at .1242 seconds in (exponential growth). At this point congestion avoidance kicks in since the number of bytes waiting to be acked never increases (linear increase).

The idealized version of TCP we have been studying would continue slowstart to use the entire window size the server is advertising rather than keeping it at a steady rate. Which then could be stabilized into a linear growth by congestion control. The congestion avoidance control is the main difference between the graph and the idealized protocol we have been studying.



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 slow start in my graph begins around about .01 seconds (exponential growth). Note, my results matched the issue mentioned earlier in the instructions PDF. My trace recorded the second segment as being over 13 thousand bytes long. The process of continually ramping up the number of bytes sent continues throughout the upload, so the slowstart never really ends. This matches the idealized version of the TCP protocol we have been studying in class since there is no congestion avoidance as seen in problem 13.