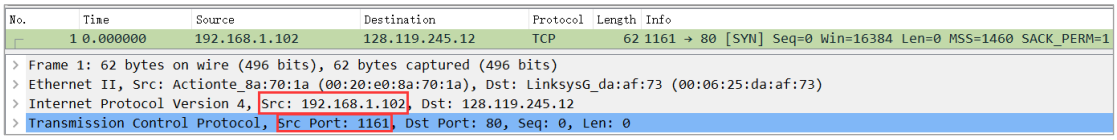
Wireshark Lab 3 – TCP
Xiaoying Li

NOTE:
I used the downloaded trace to answer questions 1 – 13, and my personal trace to answer question 14.

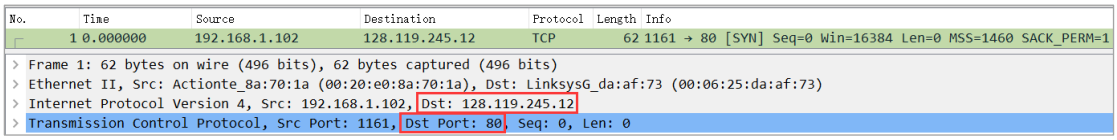
1. What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu?

IP address used by the client computer: 192.168.1.102
TCP port number used by the client computer: 1161



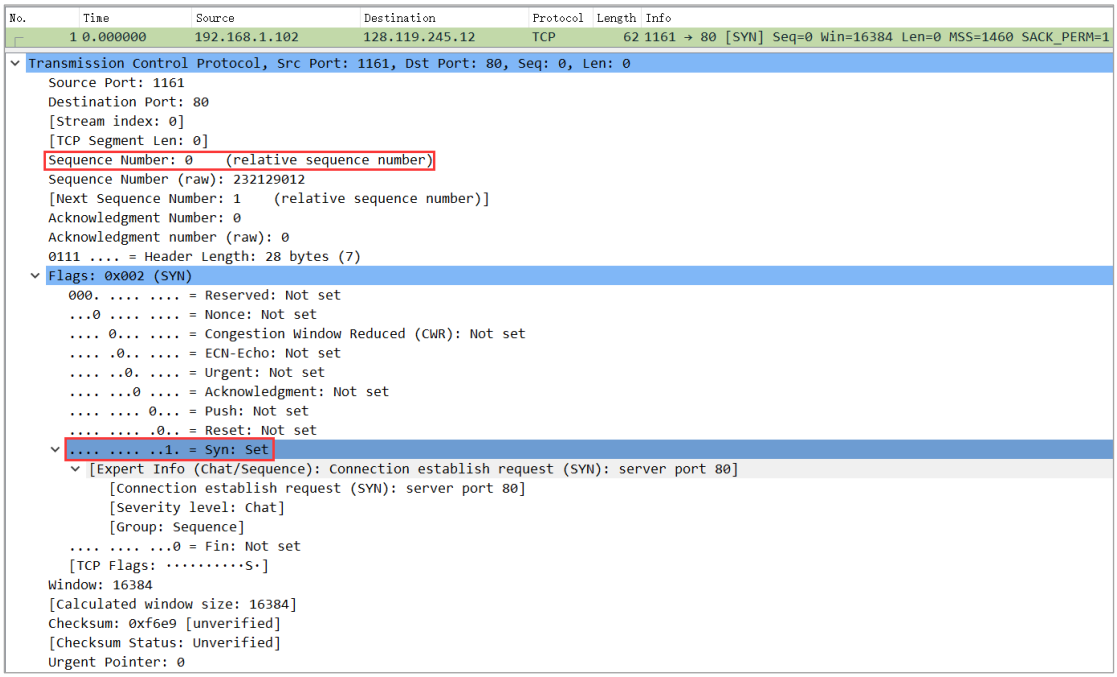
2. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

IP address of gaia.cs.umass.edu: 128.119.245.12
Port number: 80



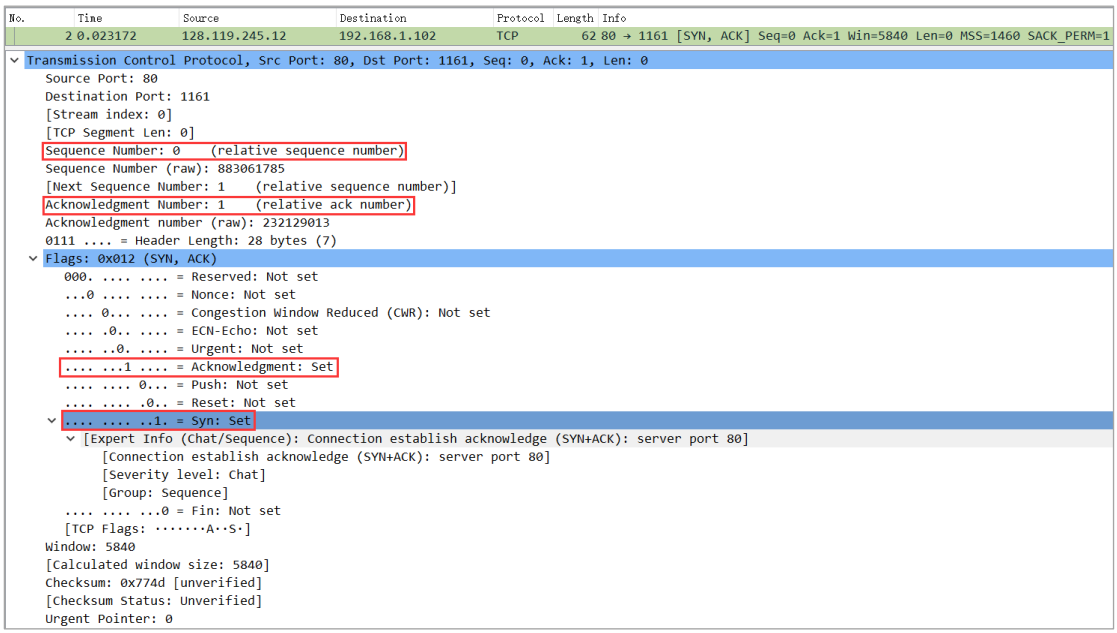
4. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?

Sequence number of the TCP SYN segment: 0 (relative sequence number)
The Syn flag in the segment is set to 1 identifies the segment as a SYN segment.



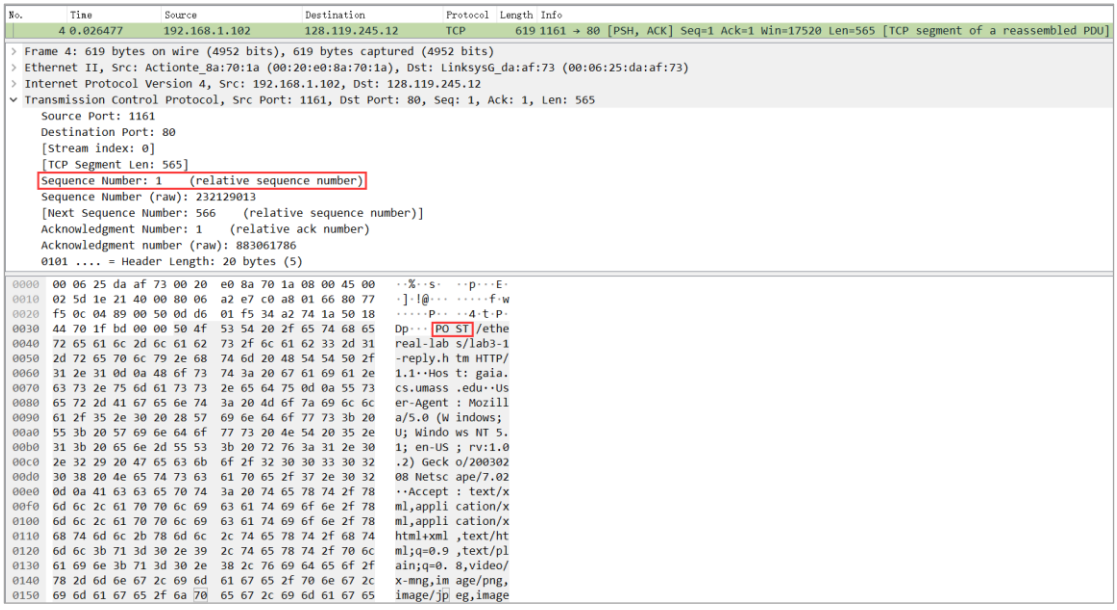
5. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK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

Sequence number of the SYNACK segment: 0 (relative sequence number)
Value of the Acknowledgement field in the SYNACK segment: 1 (relative ack number)
gaia.cs.umass.edu determine the value of the Acknowledgement field in the SYNACK segment by adding 1 to the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN, which is 0+1=1.
The Ack flag and Syn flag in the segment is set to 1 identifies the segment as a SYNACK segment.

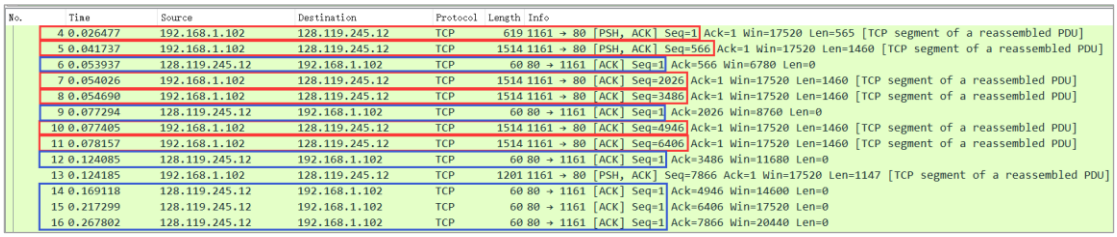


6. What is the sequence number of the TCP segment containing the HTTP POST command?

The No.4 segment contains the HTTP POST command, and its sequence number is 1 (relative sequence number).



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.



No.	Time	Source	Destination	Protocol	Length	Info
4	0.026477	192.168.1.102	128.119.245.12	TCP	619	1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565 [TCP segment of a reassembled PDU]
5	0.041737	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [PSH, ACK] Seq=566 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
6	0.053937	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=566 Win=6780 Len=0
7	0.054026	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
8	0.054690	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
9	0.077294	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=2026 Win=8760 Len=0
10	0.077405	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
11	0.078157	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
12	0.124085	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=3486 Win=11680 Len=0
13	0.124185	192.168.1.102	128.119.245.12	TCP	1201	1161 → 80 [PSH, ACK] Seq=7866 Ack=1 Win=17520 Len=1147 [TCP segment of a reassembled PDU]
14	0.169118	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=4946 Win=14600 Len=0
15	0.217299	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=6406 Win=17520 Len=0
16	0.267802	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=7866 Win=20440 Len=0

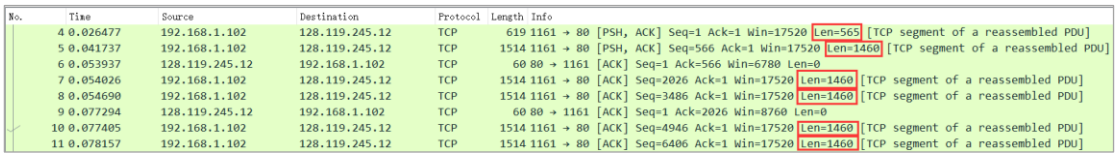
The first six segments in the TCP connection (including the segment containing the HTTP POST) are No.4, 5, 7, 8, 10, 11 segments. And the ACKs for these six segments are segments No.6, 9, 12, 14, 15, 16. Based on this, the sequence numbers (relative sequence number) of the first six segments in the TCP connection, the time of each segment was sent, the time of ACK for each segment was received, and the RTT value for each of the six segments are listed in the table below:

	Sequence Number	Segment Sent Time	ACK Received Time	RTT (seconds)
Segment 1 (No.4)	1	0.026477	0.053937	0.02746
Segment 2 (No.5)	566	0.041737	0.077294	0.035557
Segment 3 (No.7)	2026	0.054026	0.124085	0.070059
Segment 4 (No.8)	3486	0.054690	0.169118	0.114428
Segment 5 (No.10)	4946	0.077405	0.217299	0.139894
Segment 6 (No.11)	6406	0.078157	0.267802	0.189645

The EstimatedRTT value after the receipt of each ACK:
The first segment's EstimatedRTT is equal to its measured RTT. And based on section 3.5.3, page 242 in text, EstimatedRTT = (1−α) * EstimatedRTT + α * SampleRTT, where α = 0.125 recommended, thus, EstimatedRTT = 0.875 * EstimatedRTT + 0.125 * SampleRTT.
Segment 1: EstimatedRTT = the measured RTT = 0.02746 second
Segment 2: EstimatedRTT = 0.875 * 0.02746 + 0.125 * 0.035557 = 0.028472 second
Segment 3: EstimatedRTT = 0.875 * 0.028472 + 0.125 * 0.070059 = 0.03367 second
Segment 4: EstimatedRTT = 0.875 * 0.03367 + 0.125 * 0.114428 = 0.043765 second
Segment 5: EstimatedRTT = 0.875 * 0.043765 + 0.125 * 0.139894 = 0.055781 second
Segment 6: EstimatedRTT = 0.875 * 0.055781 + 0.125 * 0.189645 = 0.072514 second

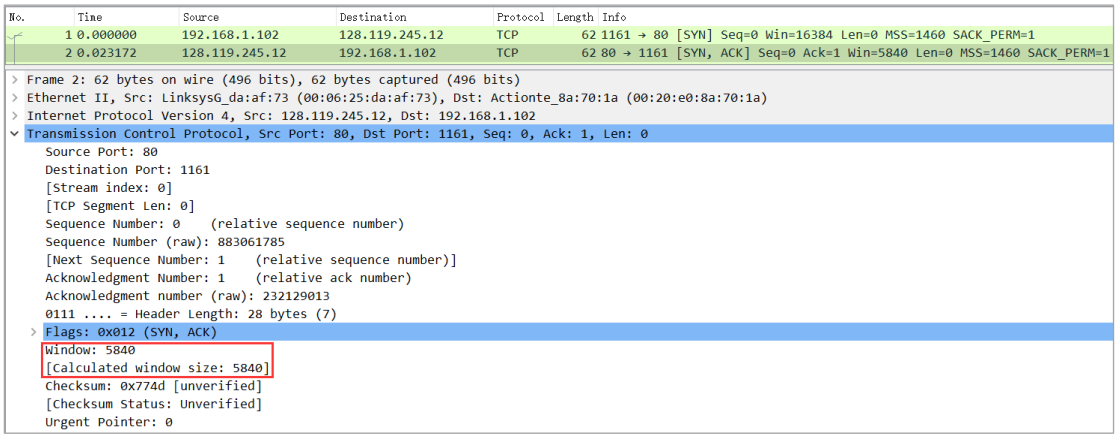
8. What is the length of each of the first six TCP segments?

Length of TCP segment 1 (No.4): 565
Length of TCP segment 2 (No.5): 1460
Length of TCP segment 3 (No.7): 1460
Length of TCP segment 4 (No.8): 1460
Length of TCP segment 5 (No.10): 1460
Length of TCP segment 6 (No.11): 1460

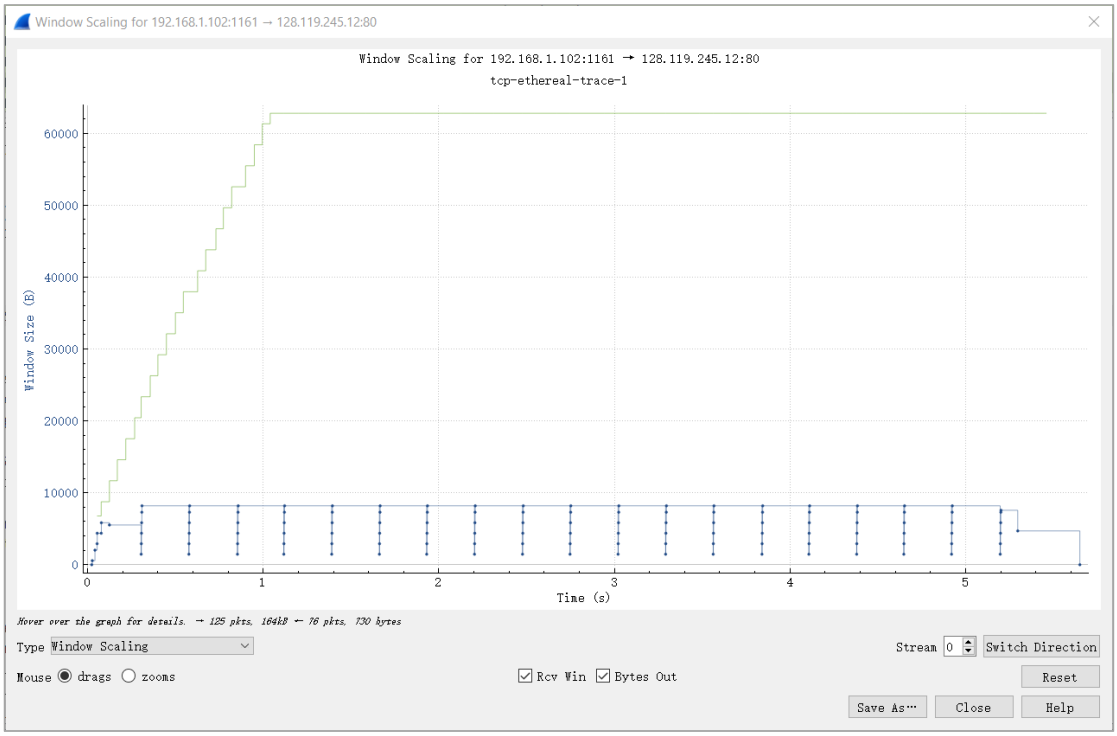


9. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

The minimum amount of available buffer space advertised at the received for the entire trace is 5840, which is the window size shown in the SYNACK segment.

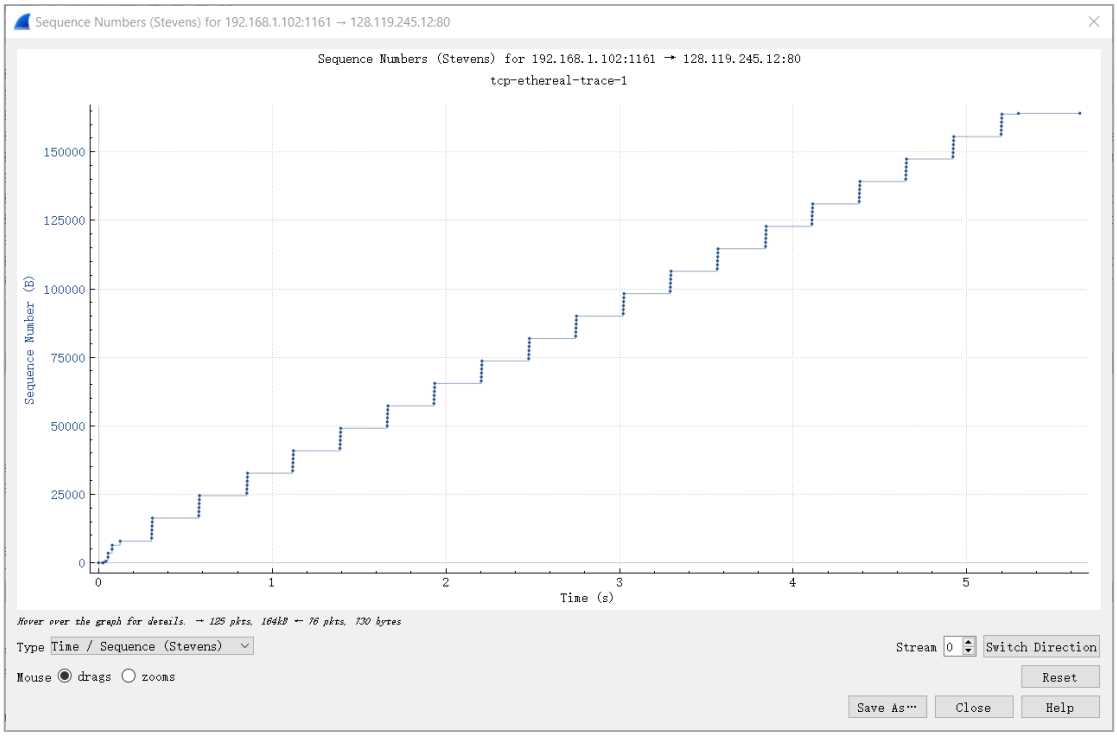


The lack of receiver buffer space never throttles the sender. Because the amounts of bytes out are always less than the receiver buffer space, as shown in the window scaling graph below.



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. In order to answer the question, I checked sequence numbers of TCP segments in the trace by looking at the Time / Sequence (Stevens) graph. As was shown in the graph, all sequence numbers for 192.168.1.102: 1161 → 128.119.245.12: 80 are increasing over time. If there was any retransmitted segment, its sequence number would be smaller than the previous sequence number.



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?

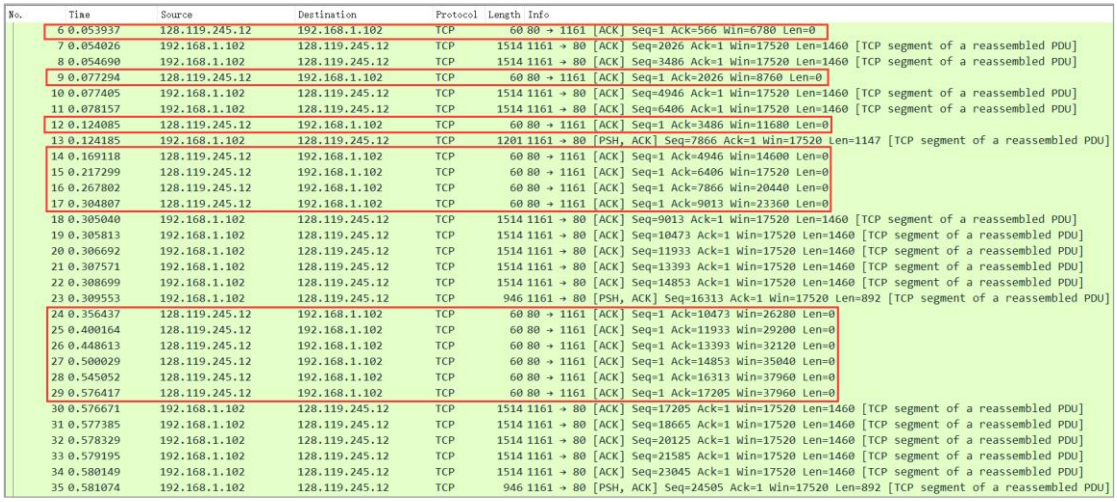
In order to know the size of data the receiver typically acknowledge in an ACK, I listed the relative acknowledgement numbers of the ACKs in the table below. The difference between one ACK's acknowledgment number and its previous ACK's acknowledgment number is the data size the receiver acknowledges in this ACK. Therefore, the size of data the receiver typically acknowledge in an ACK is 1460 bytes.

	Relative Acknowledgment Number	Acknowledged Data Size
ACK 1 (No.6)	566	566
ACK 2 (No.9)	2026	1460
ACK 3 (No.12)	3486	1460
ACK 4 (No.14)	4946	1460
ACK 5 (No.15)	6406	1460
ACK 6 (No.16)	7866	1460
ACK 7 (No.17)	9013	1147
ACK 8 (No.24)	10473	1460
ACK 9 (No.25)	11933	1460
ACK 10 (No.26)	13393	1460
ACK 11 (No.27)	14853	1460
ACK 12 (No.28)	16313	1460
ACK 13 (No.29)	17205	892

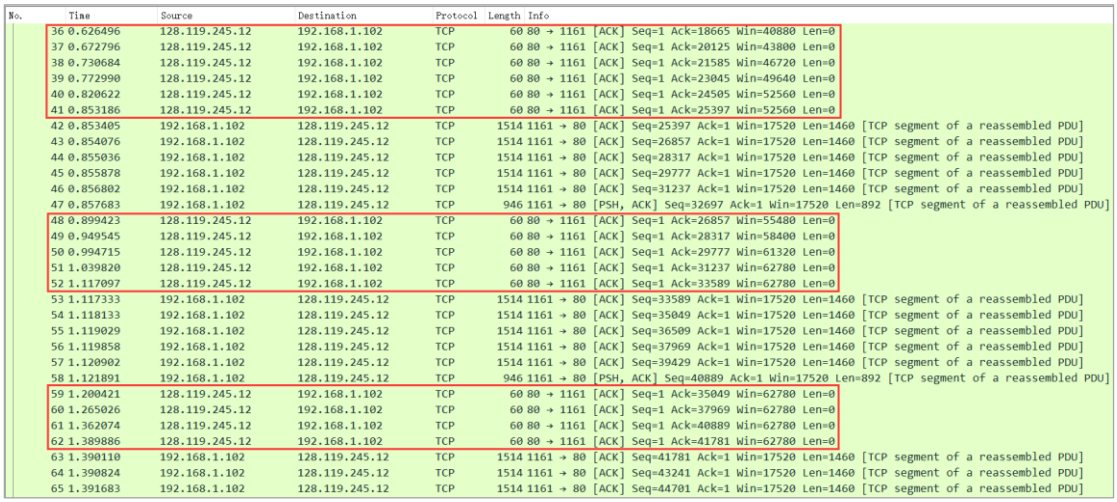
ACK 14 (No.36)	18665	1460
ACK 15 (No.37)	20125	1460
ACK 16 (No.38)	21585	1460
ACK 17 (No.39)	23045	1460
ACK 18 (No.40)	24505	1460
ACK 19 (No.41)	25397	892
ACK 20 (No.48)	26857	1460
ACK 21 (No.49)	28317	1460
ACK 22 (No.50)	29777	1460
ACK 23 (No.51)	31237	1460
ACK 24 (No.52)	33589	2352
ACK 25 (No.59)	35049	1460
ACK 26 (No.60)	37969	2920
ACK 27 (No.61)	40889	2920
ACK 28 (No.62)	41781	892

……

Yes, I can identify cases where the receiver is ACKing every other received segment. In both ACK 26 and ACK 27 listed in the above table, the receiver is ACKing 2920 = 1460 * 2 bytes data, which is two times of the data size the receiver typically acknowledges in an ACK. Therefore, in ACK 26 (segment No.60) and ACK 27 (segment No.61), the receiver is ACKing every other received segment



No.	Time	Source	Destination	Protocol	Length	Info
6	0.053937	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=566 Win=6780 Len=0
7	0.054026	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
8	0.054690	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
9	0.077294	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=2026 Win=8760 Len=0
10	0.077405	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
11	0.078157	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
12	0.124085	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=3486 Win=11680 Len=0
13	0.124185	192.168.1.102	128.119.245.12	TCP	1201	1161 → 80 [PSH, ACK] Seq=7866 Ack=1 Win=17520 Len=1147 [TCP segment of a reassembled PDU]
14	0.169118	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=4946 Win=14600 Len=0
15	0.217299	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=6406 Win=17520 Len=0
16	0.267802	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=7866 Win=20440 Len=0
17	0.304807	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=9013 Win=23360 Len=0
18	0.305040	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=9013 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
19	0.305813	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=10473 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
20	0.306692	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=11933 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
21	0.307571	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=13393 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
22	0.308699	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=14853 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
23	0.309553	192.168.1.102	128.119.245.12	TCP	946	1161 → 80 [PSH, ACK] Seq=16313 Ack=1 Win=17520 Len=892 [TCP segment of a reassembled PDU]
24	0.356437	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=10473 Win=26280 Len=0
25	0.400164	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=11933 Win=29200 Len=0
26	0.448613	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=13393 Win=32120 Len=0
27	0.500029	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=14853 Win=35040 Len=0
28	0.545052	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=16313 Win=37960 Len=0
29	0.576417	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=17205 Win=37960 Len=0
30	0.576671	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=17205 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
31	0.577385	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=18665 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
32	0.578329	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=20125 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
33	0.579195	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=21585 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
34	0.580149	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=23045 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
35	0.581074	192.168.1.102	128.119.245.12	TCP	946	1161 → 80 [PSH, ACK] Seq=24505 Ack=1 Win=17520 Len=892 [TCP segment of a reassembled PDU]

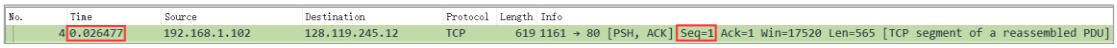


No.	Time	Source	Destination	Protocol	Length	Info
36	0.626496	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=18665 Win=40880 Len=0
37	0.672796	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=20125 Win=43800 Len=0
38	0.730684	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=21585 Win=46720 Len=0
39	0.772990	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=23045 Win=49640 Len=0
40	0.820622	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=24505 Win=52560 Len=0
41	0.853186	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=25397 Win=52560 Len=0
42	0.853405	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=25397 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
43	0.854076	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=26857 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
44	0.855036	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=28317 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
45	0.855878	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=29777 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
46	0.856802	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=31237 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
47	0.857683	192.168.1.102	128.119.245.12	TCP	946	1161 → 80 [PSH, ACK] Seq=32697 Ack=1 Win=17520 Len=892 [TCP segment of a reassembled PDU]
48	0.899423	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=26857 Win=55480 Len=0
49	0.949545	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=28317 Win=58400 Len=0
50	0.994715	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=29777 Win=61320 Len=0
51	1.039820	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=31237 Win=62780 Len=0
52	1.117097	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=33589 Win=62780 Len=0
53	1.117333	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=33589 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
54	1.118133	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=35049 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
55	1.119029	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=36509 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
56	1.119858	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=37969 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
57	1.120902	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=39429 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
58	1.121891	192.168.1.102	128.119.245.12	TCP	946	1161 → 80 [PSH, ACK] Seq=40889 Ack=1 Win=17520 Len=892 [TCP segment of a reassembled PDU]
59	1.200421	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=35049 Win=62780 Len=0
60	1.265026	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=37969 Win=62780 Len=0
61	1.362074	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=40889 Win=62780 Len=0
62	1.389886	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=41781 Win=62780 Len=0
63	1.390110	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=41781 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
64	1.390824	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=43241 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
65	1.391683	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=44701 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]

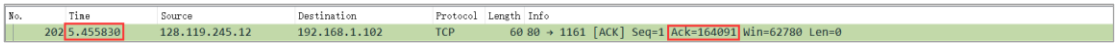
12. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

In order to calculate the throughput (bytes transferred per unit time) for the TCP connection, we need to know the total bytes transferred and the total transfer time. The TCP connection starts transfer data in segment No.4 (first TCP segment) and ends in segment No.202 (last ACK segment). So, the total bytes transferred = acknowledgment number of segment No.202 – sequence number of segment No.4; the total transfer time = time of segment No.202 – time of segment No.4. Therefore, the value of throughput can be calculated as below:

Throughput (bytes transferred per unit time)
= total bytes transferred / total transfer time
= (164091 bytes – 1 byte) / (5.455830 second – 0.026477 second)
= 30223 bytes / second



No.	Time	Source	Destination	Protocol	Length	Info
4	0.026477	192.168.1.102	128.119.245.12	TCP	619	1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565 [TCP segment of a reassembled PDU]



No.	Time	Source	Destination	Protocol	Length	Info
202	5.455830	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=164091 Win=62780 Len=0

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.

Slowstart should look like an exponential increase, so slowstart phase begins at the start of the TCP connection and ends at packet No.13 (0.1242 second). Congestion avoidance should look like a linear increase, except when congestion is detected, so congestion avoidance takes over at packet No.18 (0.305 second).
The measured data shows that there is no data being transferred during some periods of time. But in the text that we've studied, TCP tends to transfer as much data as possible, which means ideally, the sequence numbers should increase smoothly with no break over time, not the stair-step curve shown in the measured data.

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

Slowstart should look like an exponential increase, but I didn't see any exponential increase in my captured trace, so I think there is no slowstart in my captured trace. Congestion avoidance should look like a linear increase, except when congestion is detected, so congestion avoidance starts at the beginning of my captured TCP connection.

The measured data of my captured TCP trace also shows that there is no data being transferred during some periods of time. But in the text that we've studied, TCP tends to transfer as much data as possible, which means ideally, the sequence numbers should increase smoothly with no break over time, not the stair-step curve shown in my measured data.