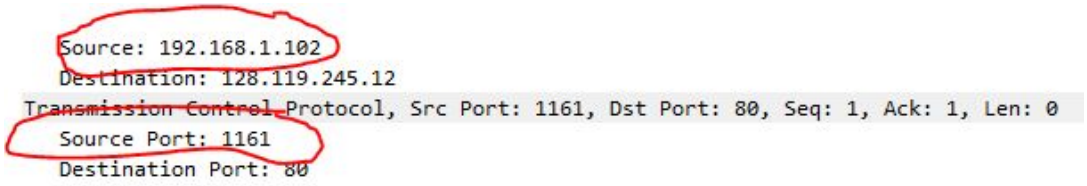
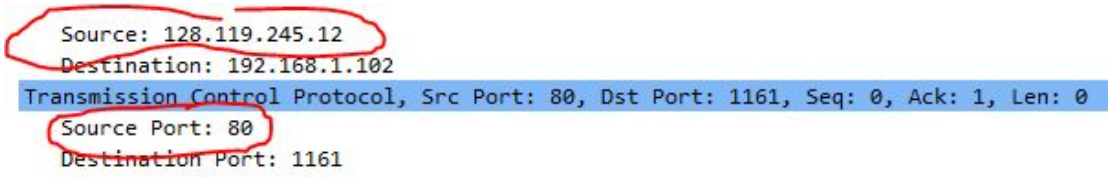
Joel Huffman
CS372-400 Spring 2019

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.**



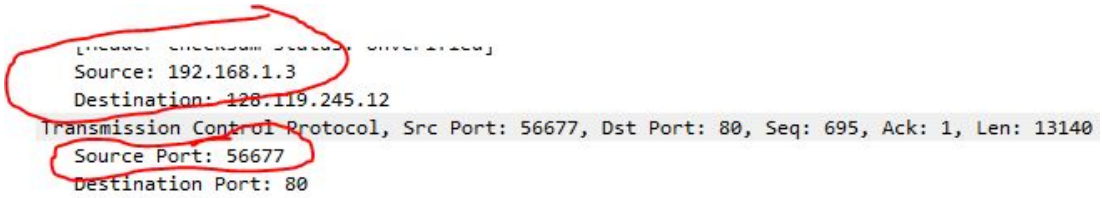
Source IP: 192.168.1.102
Source Port: 1161

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?**



Destination IP: 128.119.245.12
Destination Port: 80

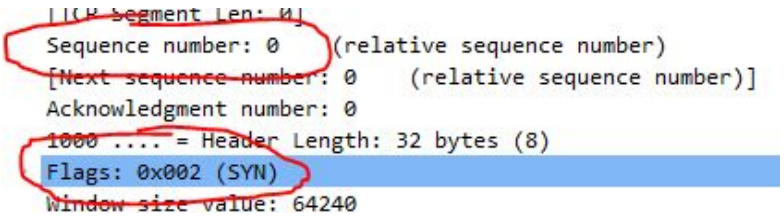
3. **What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?**



Source IP: 192.168.1.3
Source Port: 56677

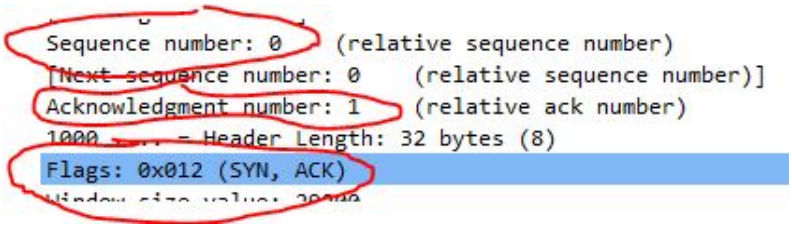
4. **What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?**

Joel Huffman
CS372-400 Spring 2019



Sequence #: 0
The SYN flag identifies it as a SYN segment

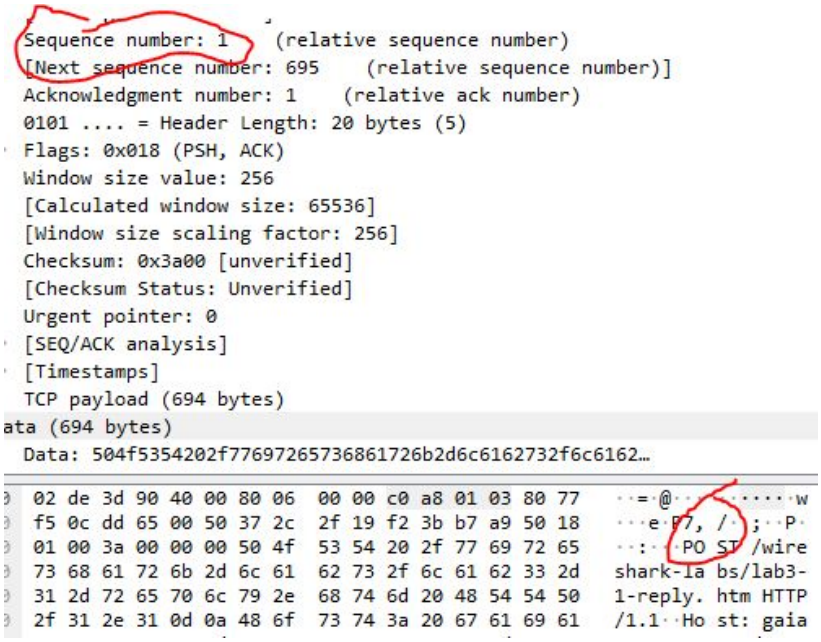
5. **What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a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



Sequence #: 0
ACK value: 1
The ACK value is the client-sent SYN segment number + 1.
The SYN and ACK flags identify the segment as 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.**

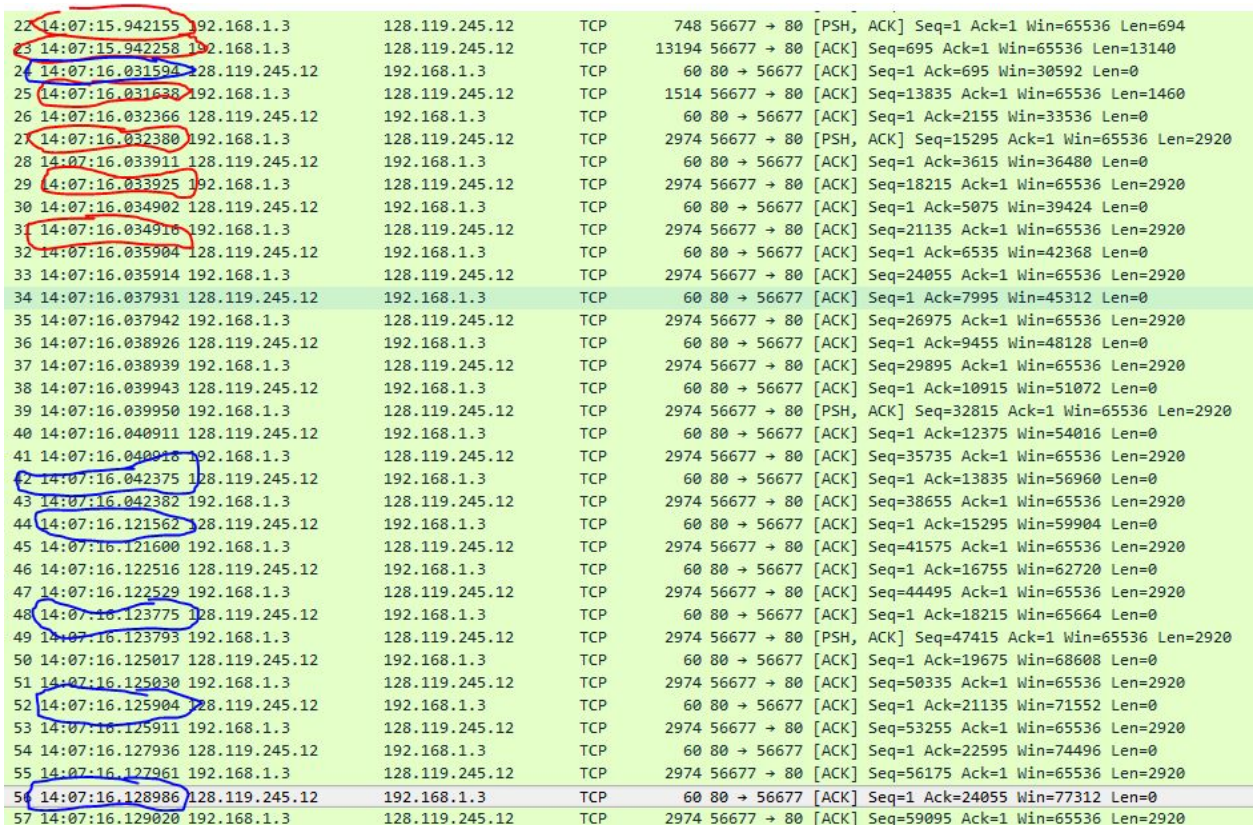
Joel Huffman
CS372-400 Spring 2019



Sequence #: 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**

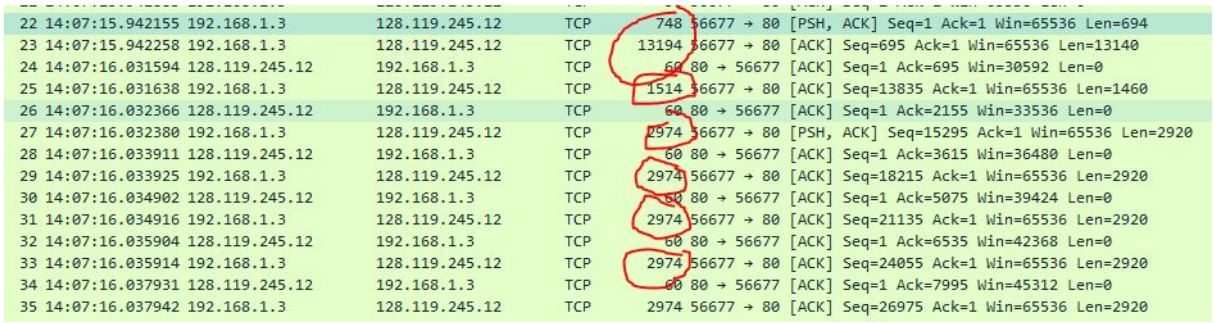
Joel Huffman
CS372-400 Spring 2019



```
22 14:07:15.942155 192.168.1.3      128.119.245.12    TCP      748 56677 → 80 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=694
23 14:07:15.942258 192.168.1.3      128.119.245.12    TCP    13194 56677 → 80 [ACK] Seq=695 Ack=1 Win=65536 Len=13140
24 14:07:16.031594 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=695 Win=30592 Len=0
25 14:07:16.031638 192.168.1.3      128.119.245.12    TCP     1514 56677 → 80 [ACK] Seq=13835 Ack=1 Win=65536 Len=1460
26 14:07:16.032366 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=2155 Win=33536 Len=0
27 14:07:16.032380 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [PSH, ACK] Seq=15295 Ack=1 Win=65536 Len=2920
28 14:07:16.033911 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=3615 Win=36480 Len=0
29 14:07:16.033925 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=18215 Ack=1 Win=65536 Len=2920
30 14:07:16.034902 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=5075 Win=39424 Len=0
31 14:07:16.034916 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=21135 Ack=1 Win=65536 Len=2920
32 14:07:16.035904 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=6535 Win=42368 Len=0
33 14:07:16.035914 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=24055 Ack=1 Win=65536 Len=2920
34 14:07:16.037931 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=7995 Win=45312 Len=0
35 14:07:16.037942 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=26975 Ack=1 Win=65536 Len=2920
36 14:07:16.038926 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=9455 Win=48128 Len=0
37 14:07:16.038939 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=29895 Ack=1 Win=65536 Len=2920
38 14:07:16.039943 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=10915 Win=51072 Len=0
39 14:07:16.039950 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [PSH, ACK] Seq=32815 Ack=1 Win=65536 Len=2920
40 14:07:16.040911 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=12375 Win=54016 Len=0
41 14:07:16.040918 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=35735 Ack=1 Win=65536 Len=2920
42 14:07:16.042375 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=13835 Win=56960 Len=0
43 14:07:16.042382 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=38655 Ack=1 Win=65536 Len=2920
44 14:07:16.121562 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=15295 Win=59904 Len=0
45 14:07:16.121600 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=41575 Ack=1 Win=65536 Len=2920
46 14:07:16.122516 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=16755 Win=62720 Len=0
47 14:07:16.122529 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=44495 Ack=1 Win=65536 Len=2920
48 14:07:16.123775 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=18215 Win=65664 Len=0
49 14:07:16.123793 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [PSH, ACK] Seq=47415 Ack=1 Win=65536 Len=2920
50 14:07:16.125017 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=19675 Win=68608 Len=0
51 14:07:16.125030 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=50335 Ack=1 Win=65536 Len=2920
52 14:07:16.125904 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=21135 Win=71552 Len=0
53 14:07:16.125911 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=53255 Ack=1 Win=65536 Len=2920
54 14:07:16.127936 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=22595 Win=74496 Len=0
55 14:07:16.127961 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=56175 Ack=1 Win=65536 Len=2920
56 14:07:16.128986 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=24055 Win=77312 Len=0
57 14:07:16.129020 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=59095 Ack=1 Win=65536 Len=2920
```

Packet #	22	23	25	27	29	31
Time Sent	14:07:15.942155	14:07:15.942258	14:07:16.031638	14:07:16.032380	14:07:16.033925	14:07:16.034916
Time ACK Received	14:07:16.031594	14:07:16.042375	14:07:16.121562	14:07:16.123775	14:07:16.125904	14:07:16.128986
SampleRTT	0.089439	0.100117	0.089924	0.091395	0.091979	0.09407
EstimatedRTT	0.089439	0.090774	0.090668	0.090759	0.090912	0.091307

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

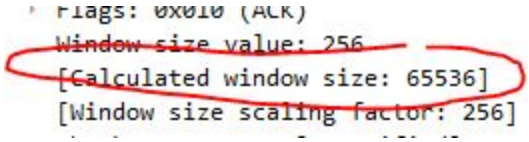


```
22 14:07:15.942155 192.168.1.3      128.119.245.12    TCP      748 56677 → 80 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=694
23 14:07:15.942258 192.168.1.3      128.119.245.12    TCP    13194 56677 → 80 [ACK] Seq=695 Ack=1 Win=65536 Len=13140
24 14:07:16.031594 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=695 Win=30592 Len=0
25 14:07:16.031638 192.168.1.3      128.119.245.12    TCP     1514 56677 → 80 [ACK] Seq=13835 Ack=1 Win=65536 Len=1460
26 14:07:16.032366 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=2155 Win=33536 Len=0
27 14:07:16.032380 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [PSH, ACK] Seq=15295 Ack=1 Win=65536 Len=2920
28 14:07:16.033911 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=3615 Win=36480 Len=0
29 14:07:16.033925 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=18215 Ack=1 Win=65536 Len=2920
30 14:07:16.034902 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=5075 Win=39424 Len=0
31 14:07:16.034916 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=21135 Ack=1 Win=65536 Len=2920
32 14:07:16.035904 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=6535 Win=42368 Len=0
33 14:07:16.035914 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=24055 Ack=1 Win=65536 Len=2920
34 14:07:16.037931 128.119.245.12   192.168.1.3       TCP       60 80 → 56677 [ACK] Seq=1 Ack=7995 Win=45312 Len=0
35 14:07:16.037942 192.168.1.3      128.119.245.12    TCP     2974 56677 → 80 [ACK] Seq=26975 Ack=1 Win=65536 Len=2920
```

Segment	1	2	3	4	5	6

Length	748	13194	1514	2974	2974	2974

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?**

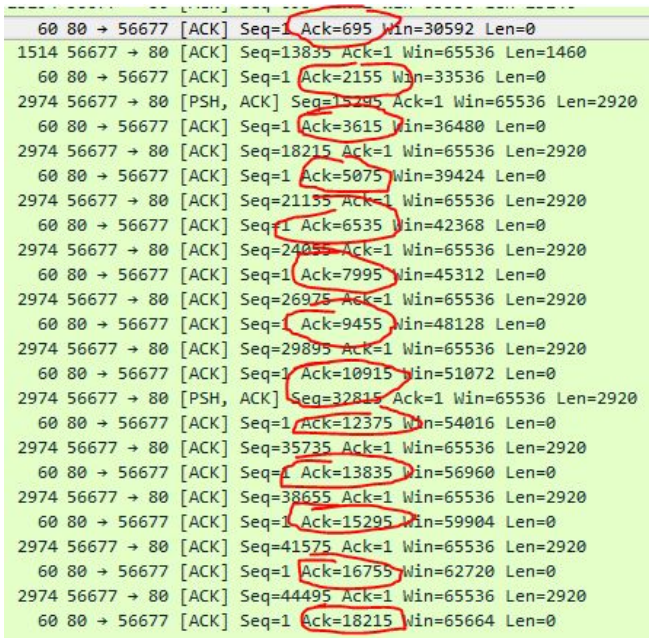


Minimum available buffer: 65536
The sender was never throttled during the span of the trace.

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

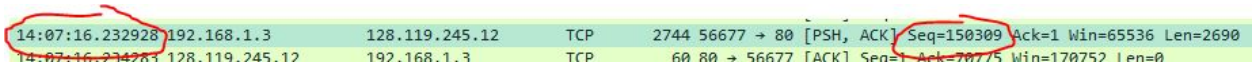
No segments were retransmitted. This was confirmed by finding no segments with duplicate sequence numbers.

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 regularly sent 1460 byte packets. This can be seen in the difference in ACK values.

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



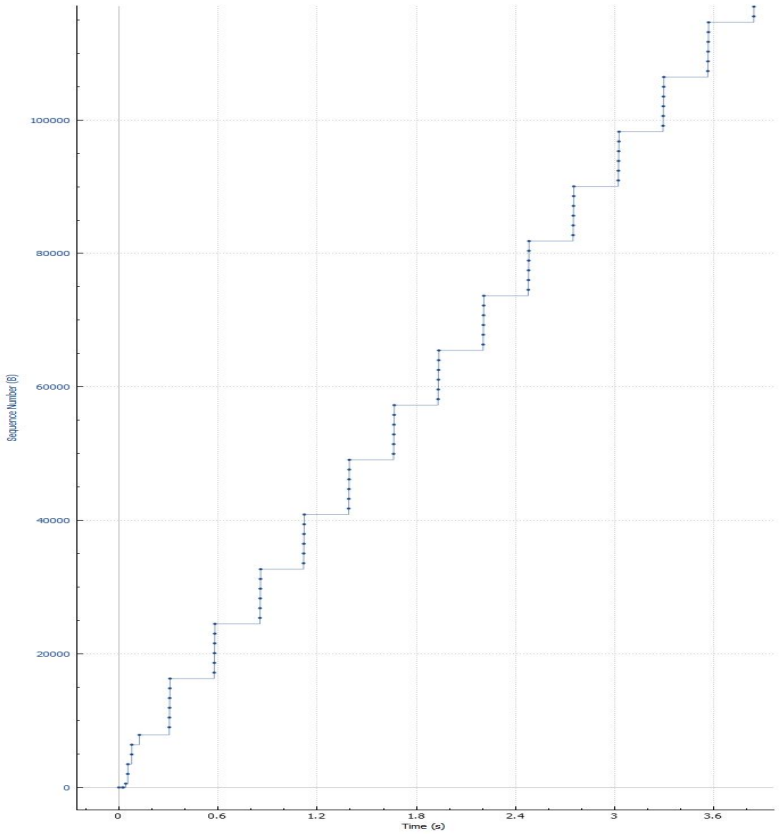
Throughput is calculated by dividing the total amount of data (in bytes) by the duration of time it took to transfer that data. Start time of the transfer can be pulled the the segment that sends the POST data from question 7. Our end time is from the last sent packet of our data as seen in the screenshot above. The sequence number from this last segment can also be used to see how many total bytes we have transferred up to this point.

Start time: 14:07:15.942155
End time: 14:07:16.232928
Bytes transferred: 150309

$$Throughput = \frac{bytes\ transferred}{Elapsed\ time} = \frac{bytes\ transferred}{End\ time - start\ time} = \frac{150309}{16.232928 - 15.942155} = 516929$$

Throughput: 516929 bytes/sec

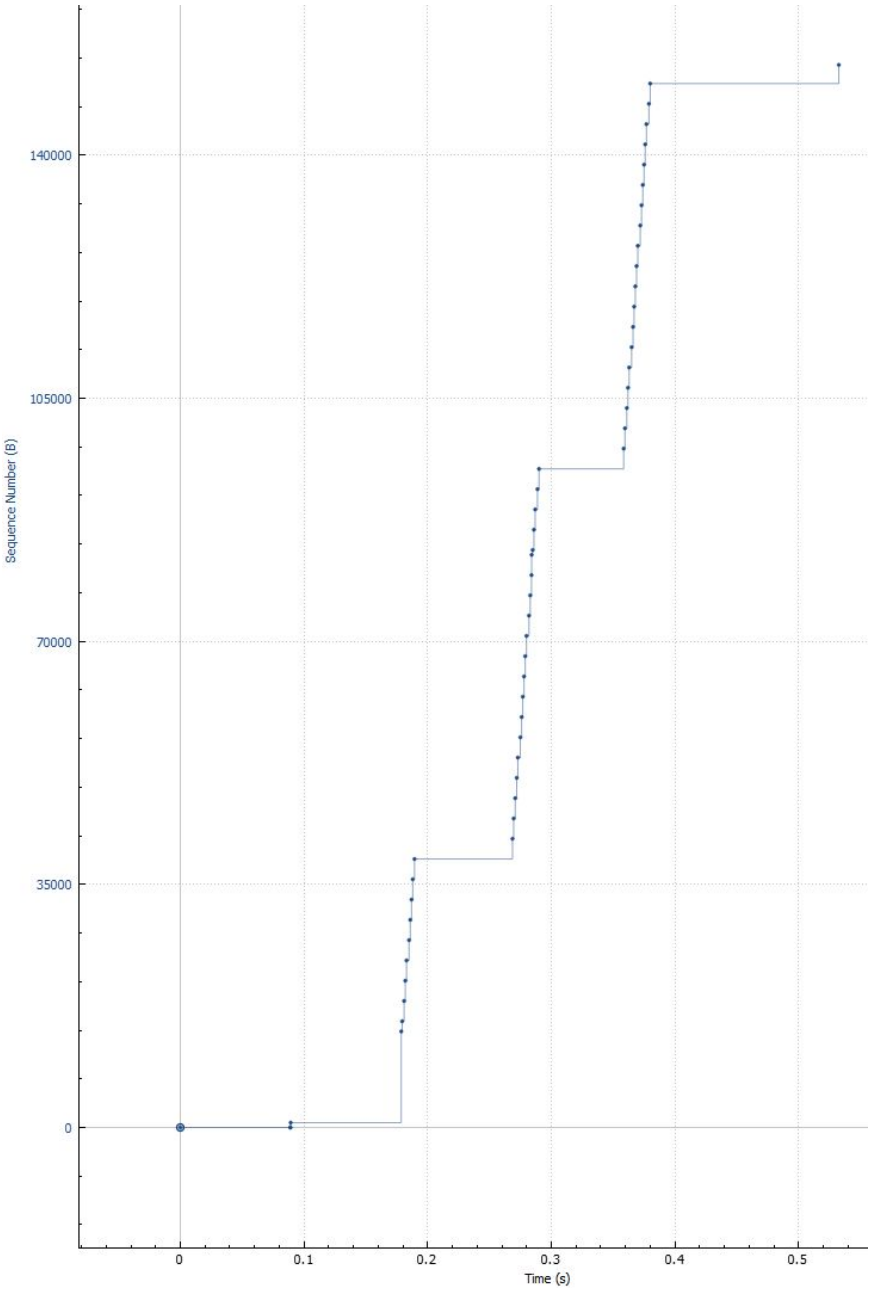
13. **Use the Time-Sequence-Graph(Stevens) plotting tool to view the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. Can you identify where TCP's slowstart phase begins and ends, and where congestion avoidance takes over? Comment on ways in which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



Slow-start begins almost immediately at ~0.05 seconds and continues until ~0.1 seconds. Then congestion avoidance takes over by growing linearly, but because congestion is detected after sending just a few packets it delays sending more for a bit. This differs from ideal behavior because we aren't seeing linear growth of packets sent, but rather these steps at regular intervals.

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

Joel Huffman
CS372-400 Spring 2019



Similar to the provided trace, slow-start begins almost immediately and ends right before 0.09 seconds. The congestion avoidance looks similar to the previous graph as well, but now more packets are being sent before in fewer groupings. This causes only a few, very tall steps to be seen.