VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

COMPUTER NETWORKS (LAB)

Report lab 4a

Wireshark Lab: TCP v8.0

Advisor(s): Nguyễn Mạnh Thìn
Student(s): Vũ Nguyễn Lan Vi ID 2153094

HO CHI MINH CITY, APRIL 2024

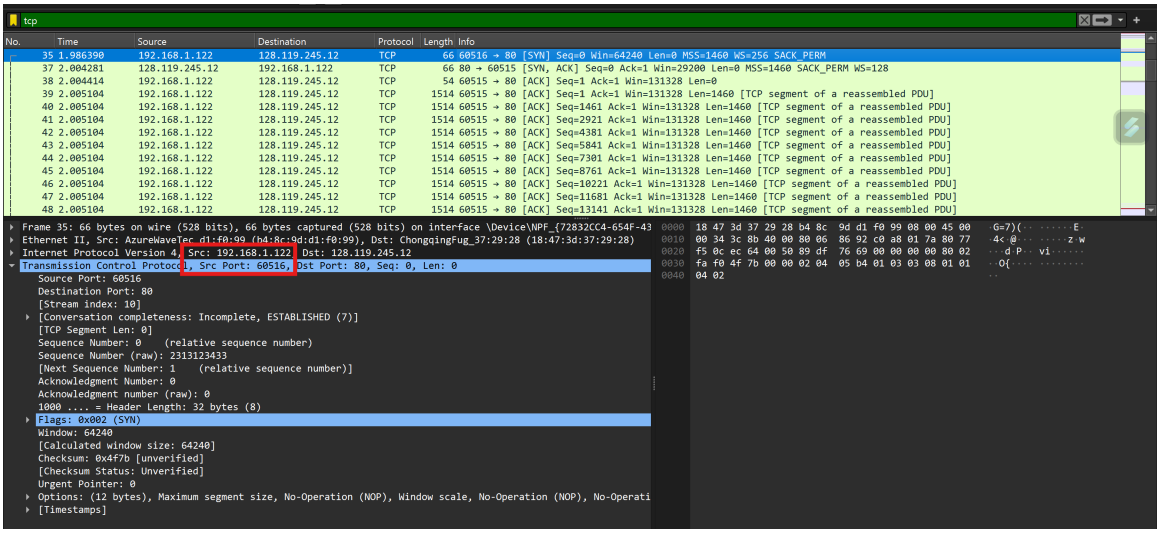
Contents

1 Exercise

- **Question 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 address: 192.168.1.102
  Port: 1161

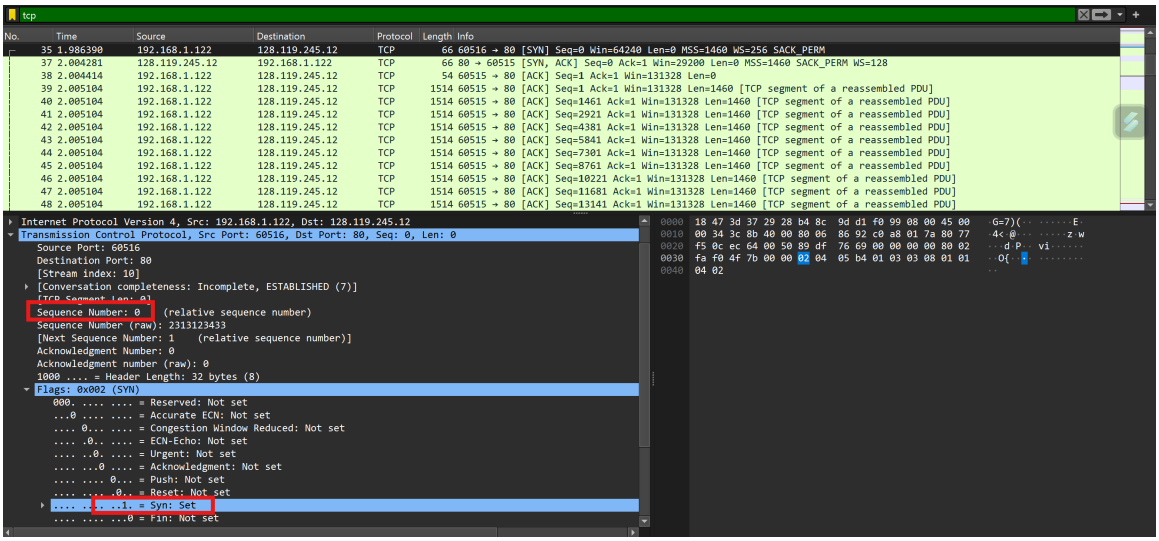
- **Question 2:** What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?
  IP address: 128.119.245.12
  Port: 80

- **Question 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?
  IP address: 192.168.1.122
  Port: 60516



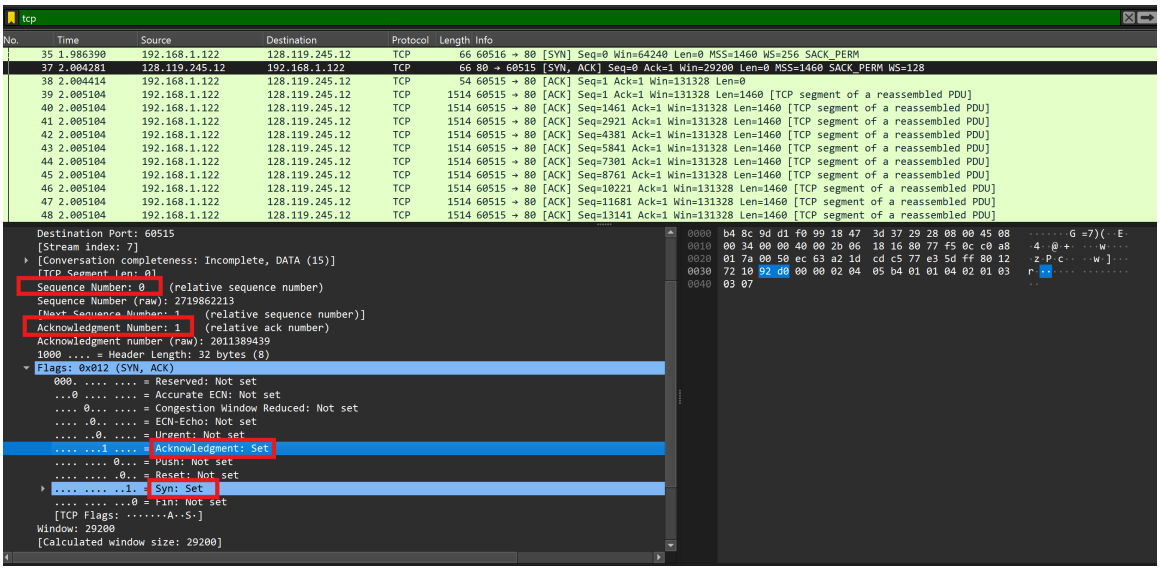
- **Question 4:** What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?
  The sequence number of the TCP SYN segment that is used to initiate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

connection between the client computer and gaia.cs.umass.edu is **0**. The **Syn flag** is set indicates that the segement is SYN segment.

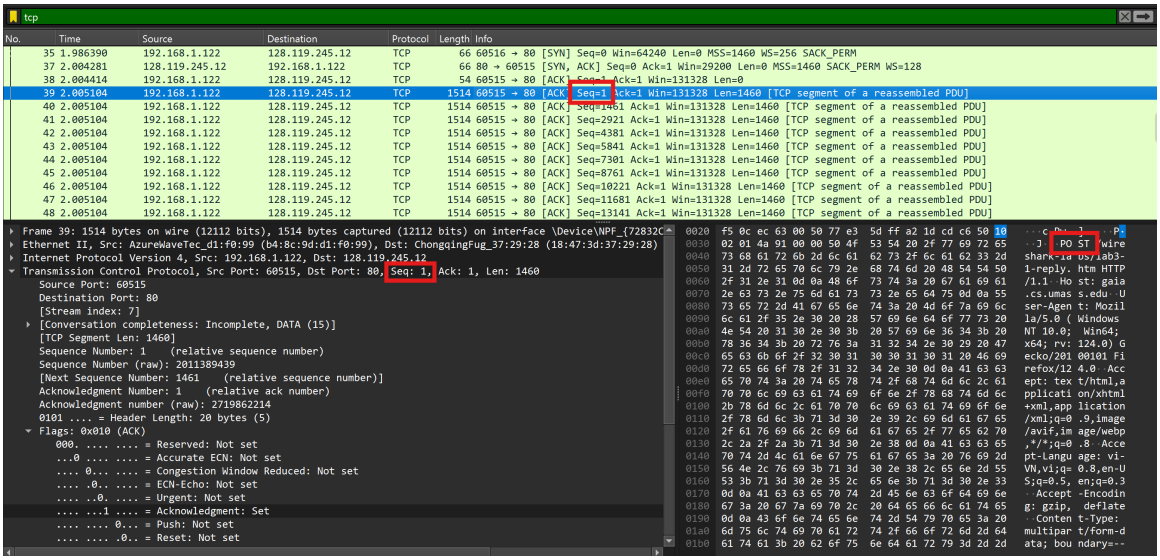


- **Question 5:** What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the 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 Acknowledgement field in the SYNACK segment is 1. In TCP 3-way handshake, the acknowledgment number inc ACK/SYN packet equals the sequence number of the SYN packet incremented by 1. The **Syn flag** and **Acknowladgement flag** is set indicates that the segement is SYN segment.

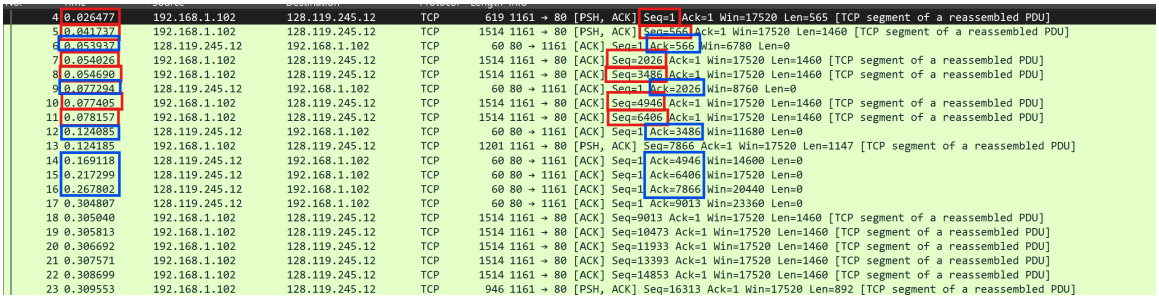


- **Question 6:** What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.
  The sequence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



- **Question 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.



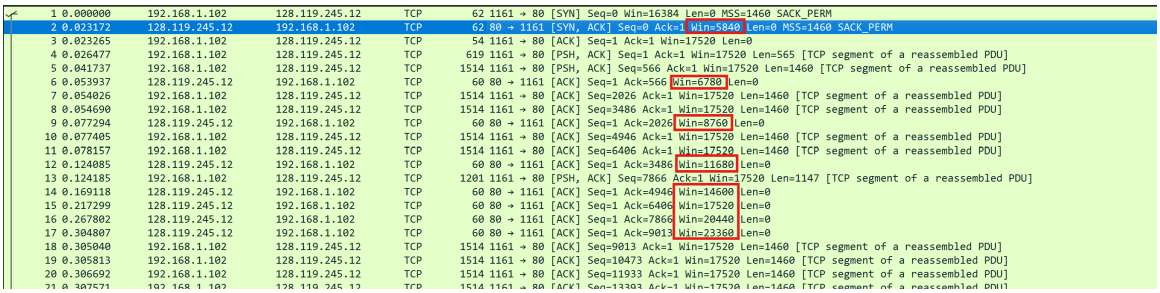
Seq Number	Sent Time	ACK received time	RTT value = ACK received time - Sent time
1	0.026477	0.053937	0.02746
566	0.041727	0.077294	0.035567
2026	0.054026	0.124085	0.070059
3486	0.05469	0.169118	0.114428
4946	0.077405	0.217299	0.139894
6406	0.078157	0.267802	0.189645

Segment	SampleRTT	Estimated	EstimatedRTT= (1-a)*Estimated + a*SampleRTT (a=0.125)
1	0.02746	0.02746	0.02746
2	0.035567	0.02746	0.028473375
3	0.070059	0.02847338	0.033671578
4	0.114428	0.03367158	0.043766131
5	0.139894	0.04376613	0.055782115
6	0.189645	0.05578211	0.072514975

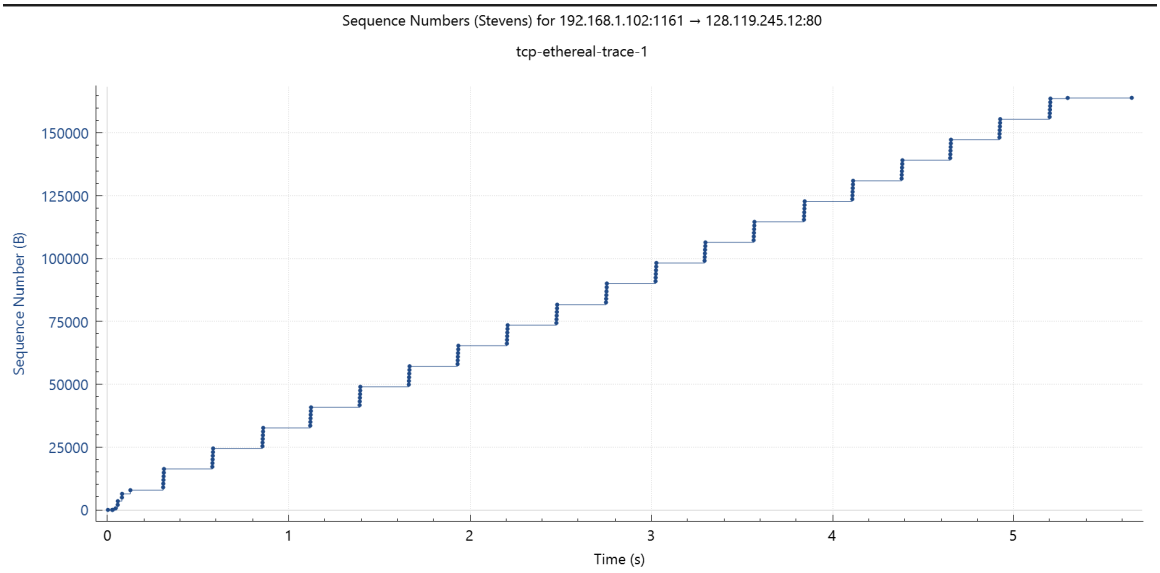
- **Question 8:** What is the length of each of the first six TCP segments?
  The length of the first segment is 565 bytes, and the legth of the next 5 TCP segments are 1460 bytes.

- **Question 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?

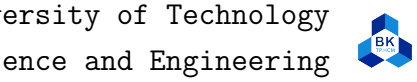
the minimum amount of available buffer space advertised at the received for the entire trace is 5480. As the minimum number of bytes receiver willing to accept is 5480, there is no lack of receiver buffer space that throttle the sender.



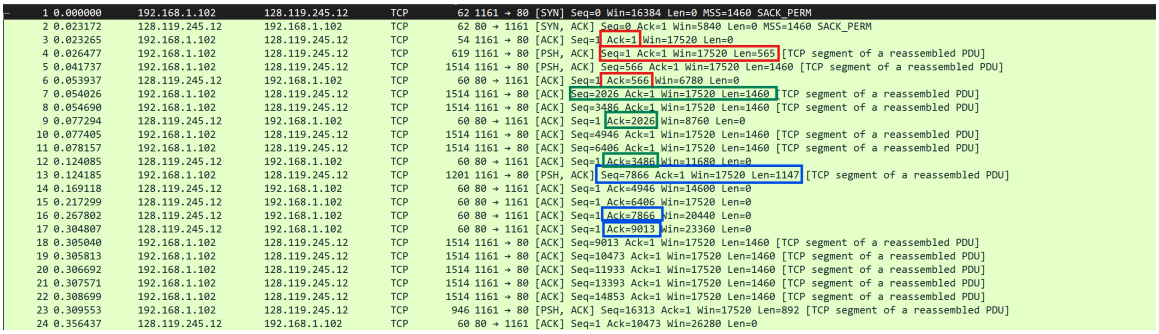
- **Question 10:** Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?
  There are no packet with same sequence number at different time (See Tme/Sequence graph below). Hence, there are no retransmitted segments in the trace file.



- **Question 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 number (amount) of data transfered can be indicated by the different betweens 2 consecutive acknowledged sequence numbers. We can infer it from the figure below:



- 2 consecutive ACKs: 1 and 566, the number of bytes transfered is 565 bytes (566 - 1).
- 2 consecutive ACKs: 2026 and 3486, the number of bytes transfered is 1460 bytes (3486 - 2026).
- 2 consecutive ACKs: 7866 and 9013, the number of bytes transfered is 1147 bytes (9013 - 7866).



1	0.000000	192.168.1.102	128.119.245.12	TCP	62 1161 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM
2	0.023172	128.119.245.12	192.168.1.102	TCP	62 80 → 1161 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM
3	0.023265	192.168.1.102	128.119.245.12	TCP	54 1161 → 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0
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
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

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

$$\begin{aligned}\text{Total data transfer} &= \text{ACK of the last Acknowledged segment (\#202)} \\ &\quad - \text{ACK of the segment containing the HTTP POST command (\#4)} \\ &= 164091 - 1 \\ &= 164090 \text{ (bytes)}\end{aligned}$$

$$\begin{aligned}\text{Total amount of time} &= \text{ACK received time of the last Acknowledged segment (\#202)} \\ &\quad - \text{ACK received time of the HTTP POST command segment(\#4)} \\ &= 5.455830 - 0.026477 \\ &= 5.429353 \text{ (s)}\end{aligned}$$

$$\begin{aligned}\text{Throughput} &= \frac{\text{total data transfer}}{\text{total amount of time}} \\ &= \frac{164090}{5.429353} \\ &= 30.223 \text{ KBytes/s}\end{aligned}$$

- **Question 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.
  TCP sender limits transmission: $LastByteSent - LastByteAcked \leq cwnd$

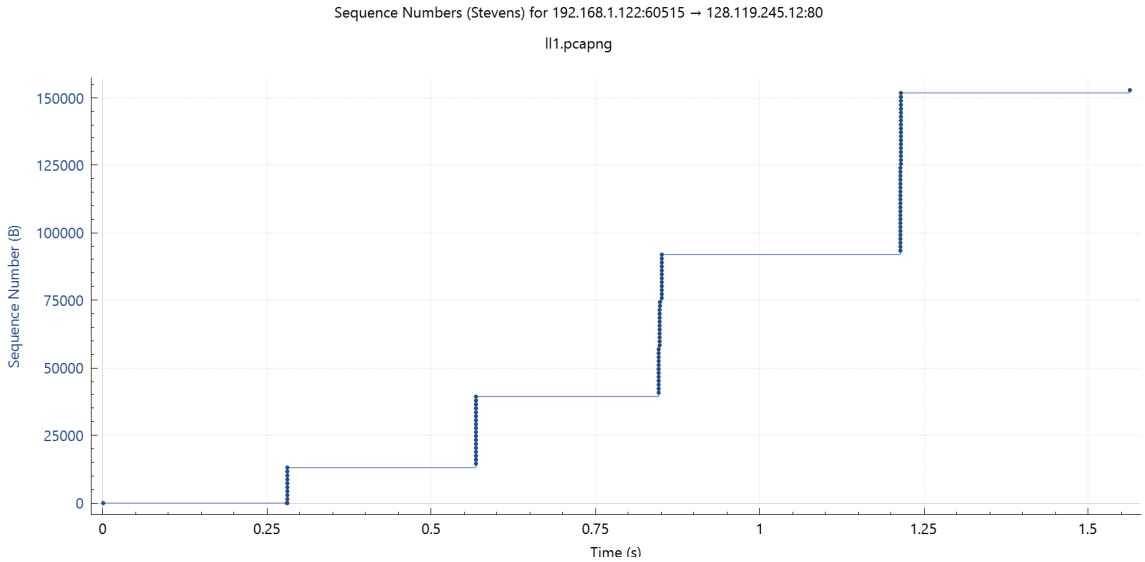
  Here, we can observe that the amount of data transferred never exceeds the receiver window size (the receiver buffer can act as the upper bound of the TCP window size). The amount of available buffer space advertised at the receiver increases up to 62780, showing that this TCP sender is not sending data aggressively enough to push the congestion rate. However, the amount of outstanding data (data sent but not yet acknowledged) is always less than 8192 bytes, indicating that before receiving the ACK for the whole 6 blocks (8192 bytes) here, the application will not send more data. Hence, the amount of data transmission did not grow linearly not because of the flow control since the advertised window is significantly larger than 6 packets.

  Upon an ACK arrival, if the congestion window size increases by one MSS, the TCP sender still stays in the slow start phase. In the congestion avoidance phase, the congestion window size increases at $\frac{1}{\text{current congestion window size}}$. By inspecting the change of the congestion window upon the arrival of ACKs, we can say that: The slow start phase of congestion avoidance begins at the start of the connection when the sender sends the segment containing the POST command. Before the end of the slow start phase, the application already stops transmission temporarily.

  TCP's ideal behavior assumes aggressive data transmission from TCP senders, which can lead to network congestion under heavy traffic loads. To mitigate this issue, TCP senders are advised to adhere to the AIMD algorithm. This algorithm triggers a reduction in the sending window size upon detecting network congestion, typically

signaled by packet loss. In practice, TCP behavior varies depending on the specific application. For instance, in scenarios where the TCP sender is ready to transmit data, it may encounter a lack of available data for transmission. This situation is common in web applications where certain web objects have small sizes. Consequently, due to TCP's slow start phase, the transmission of these small web objects experiences unnecessary delays because of the slow start phase of TCP.

- **Question 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
  In the trace data I collected, the slow start phase ended earlier than expected, and the sending window began to double in size after 0.25 seconds. Although the advertised window continued to increase, after 1.25 seconds, the amount of outstanding data (sending window) began to increase linearly instead of doubling as observed earlier.



243	3.226496	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=124101 Win=232832 Len=0
244	3.226496	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=125561 Win=235776 Len=0
245	3.227796	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=127021 Win=238720 Len=0
246	3.227796	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=128481 Win=241664 Len=0
247	3.227796	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=129941 Win=244608 Len=0
248	3.229023	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=131401 Win=247424 Len=0
249	3.229023	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=132861 Win=250368 Len=0
250	3.229023	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=134321 Win=253312 Len=0
251	3.230661	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=135781 Win=256256 Len=0
252	3.230661	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=137241 Win=259200 Len=0
253	3.230661	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=138701 Win=262144 Len=0
254	3.230661	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=140161 Win=264960 Len=0
255	3.230661	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=141621 Win=267904 Len=0
256	3.233090	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=143081 Win=270848 Len=0
257	3.233090	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=144541 Win=273792 Len=0
258	3.233090	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=146001 Win=276736 Len=0
259	3.233090	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=147461 Win=279552 Len=0
260	3.233255	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=148921 Win=282496 Len=0
261	3.233255	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=150381 Win=285440 Len=0
262	3.233255	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=151841 Win=288384 Len=0
263	3.233255	128.119.245.12	192.168.1.122	TCP	60 80 → 60515 [ACK] Seq=1 Ack=152905 Win=291328 Len=0