LAB-2

DATA COMMUNICATIONS NETWORKS

Prof. Dr. Karim Banawan - Prof. Dr. Noha ElKorany
Communication & Electronics Department

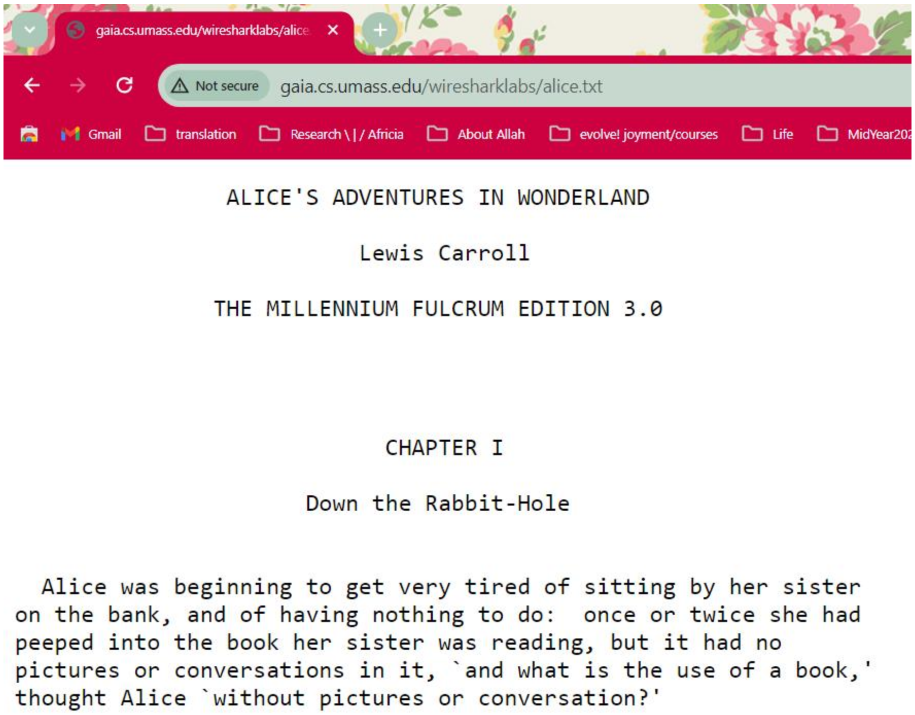
Asmaa Gamal Abdel-Halem Mabrouk Nagy
أسماء جمال عبد الحليم مبروك ناجي
15010473 - section 8

Wireshark Lab: TCP v6.0

By: “Computer Networking: A Top Down Approach, 6th edition”

<u>Section 1</u> : Capturing a bulk TCP transfer from your computer to a remote Server

The Task:



ALICE'S ADVENTURES IN WONDERLAND

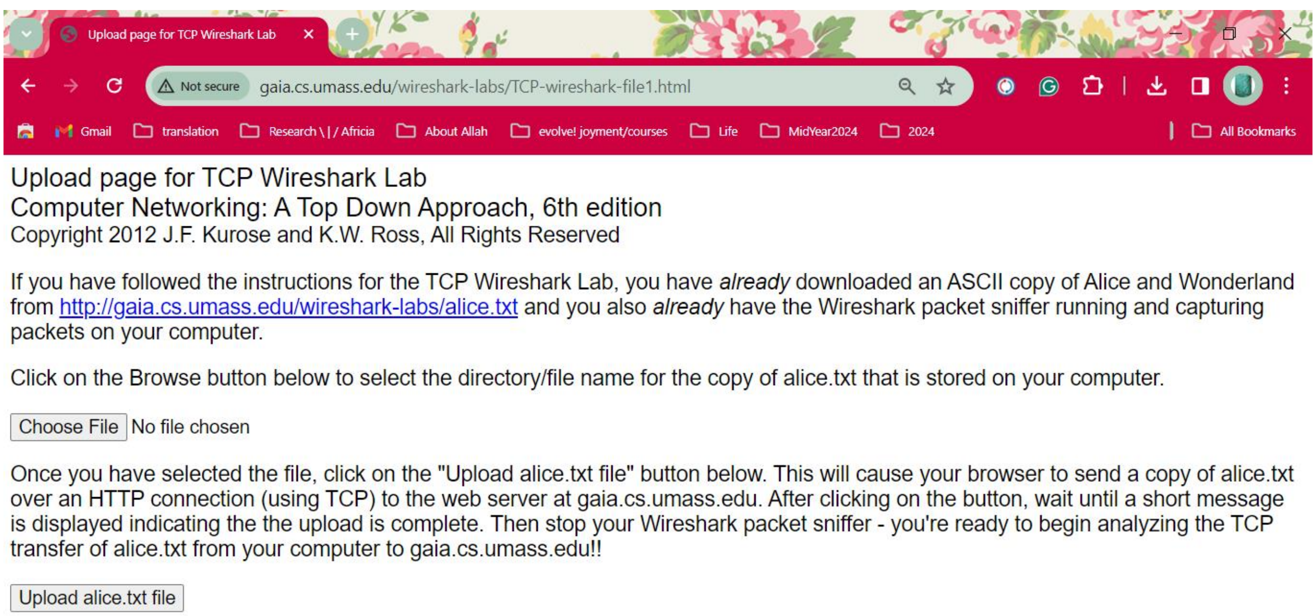
Lewis Carroll

THE MILLENNIUM FULCRUM EDITION 3.0

CHAPTER I

Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, `and what is the use of a book,' thought Alice `without pictures or conversation?'



Upload page for TCP Wireshark Lab
Computer Networking: A Top Down Approach, 6th edition
Copyright 2012 J.F. Kurose and K.W. Ross, All Rights Reserved

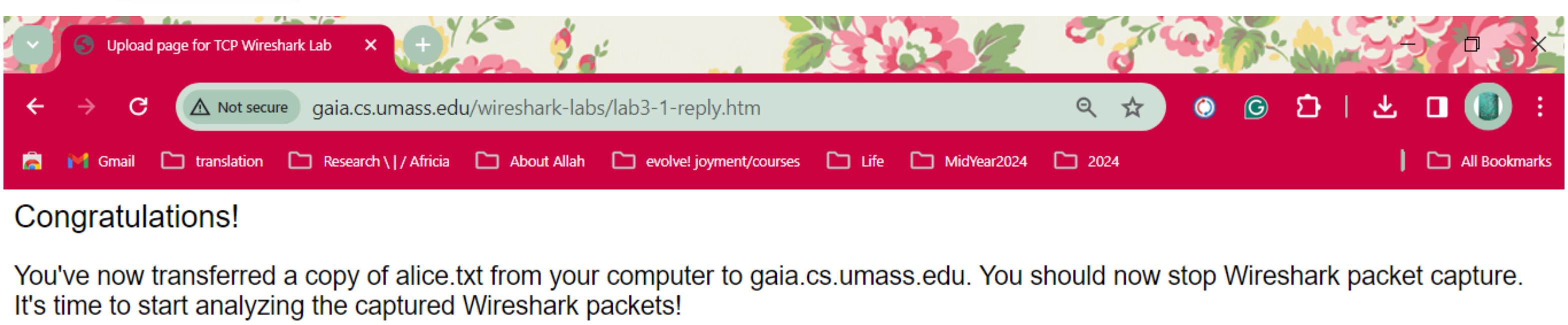
If you have followed the instructions for the TCP Wireshark Lab, you have *already* downloaded an ASCII copy of Alice and Wonderland from http://gaia.cs.umass.edu/wireshark-labs/alice.txt and you also *already* have the Wireshark packet sniffer running and capturing packets on your computer.

Click on the Browse button below to select the directory/file name for the copy of alice.txt that is stored on your computer.

Choose File No file chosen

Once you have selected the file, click on the "Upload alice.txt file" button below. This will cause your browser to send a copy of alice.txt over an HTTP connection (using TCP) to the web server at gaia.cs.umass.edu. After clicking on the button, wait until a short message is displayed indicating the the upload is complete. Then stop your Wireshark packet sniffer - you're ready to begin analyzing the TCP transfer of alice.txt from your computer to gaia.cs.umass.edu!!

Upload alice.txt file



Congratulations!

You've now transferred a copy of alice.txt from your computer to gaia.cs.umass.edu. You should now stop Wireshark packet capture. It's time to start analyzing the captured Wireshark packets!

<u>Section 2</u>: A first look at the captured trace:

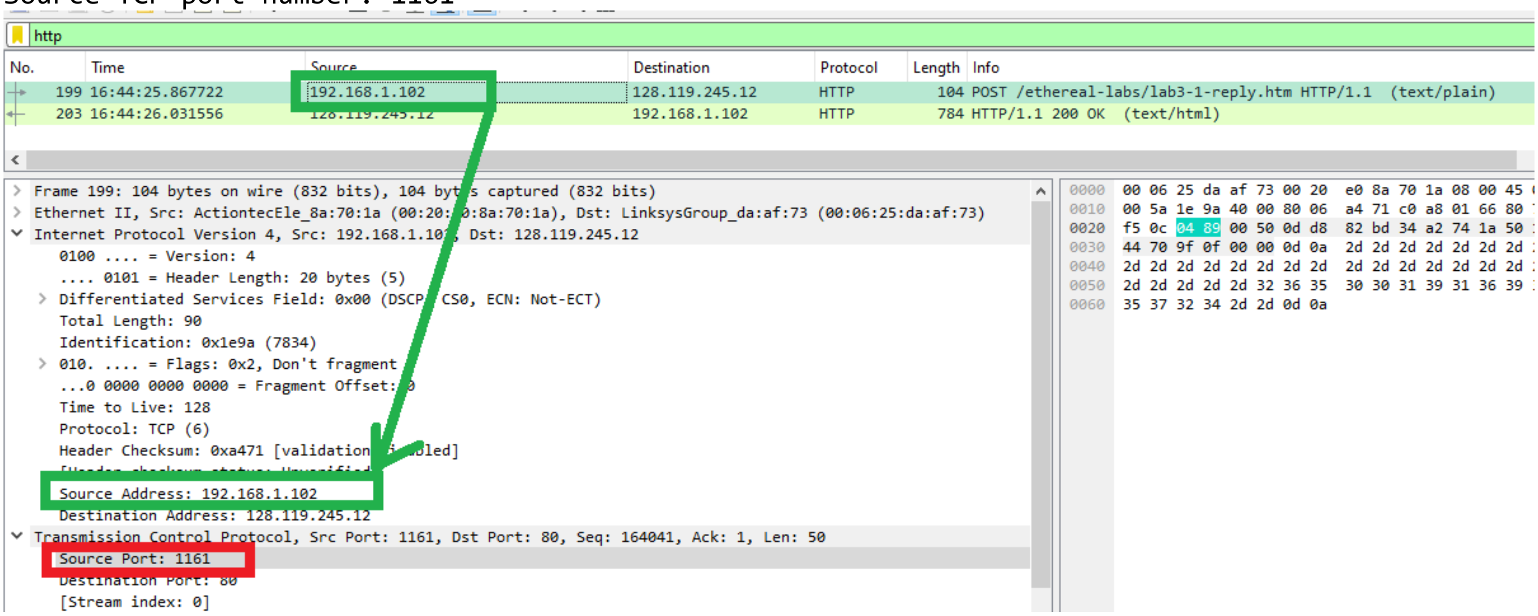
The Questions & Answers:

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?

<u>The book author's client computer (source):</u>

Source IP Address: 192.168.1.102

Source TCP port number: 1161

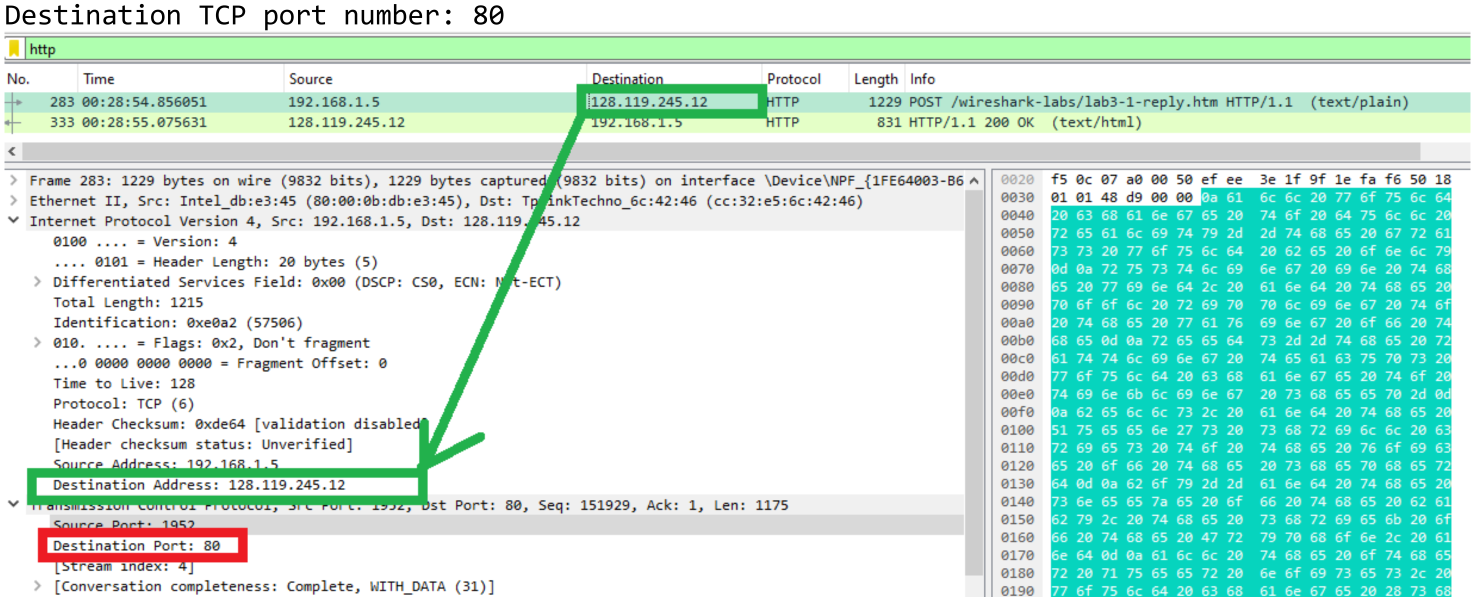


2. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

<u>Destination Server:</u> gaia.cs.umass.edu

Destination IP Address: 128.119.245.12

Destination TCP port number: 80

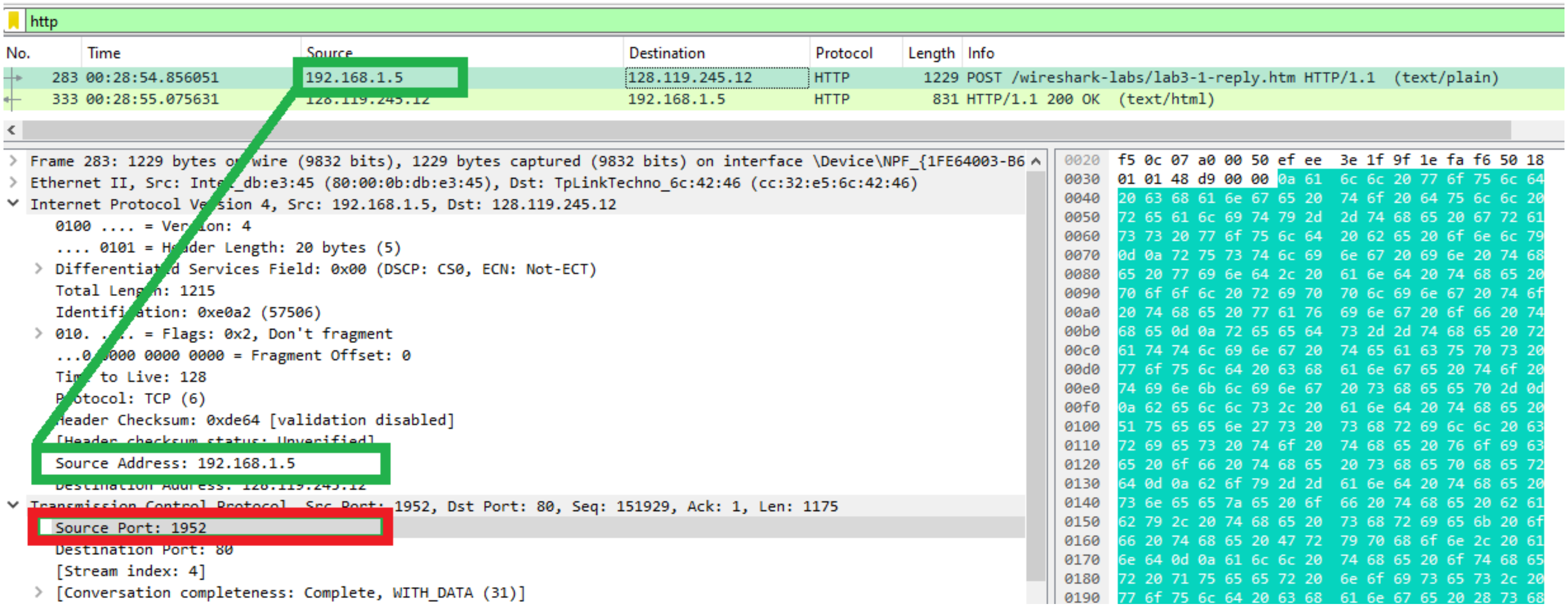


3. What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?

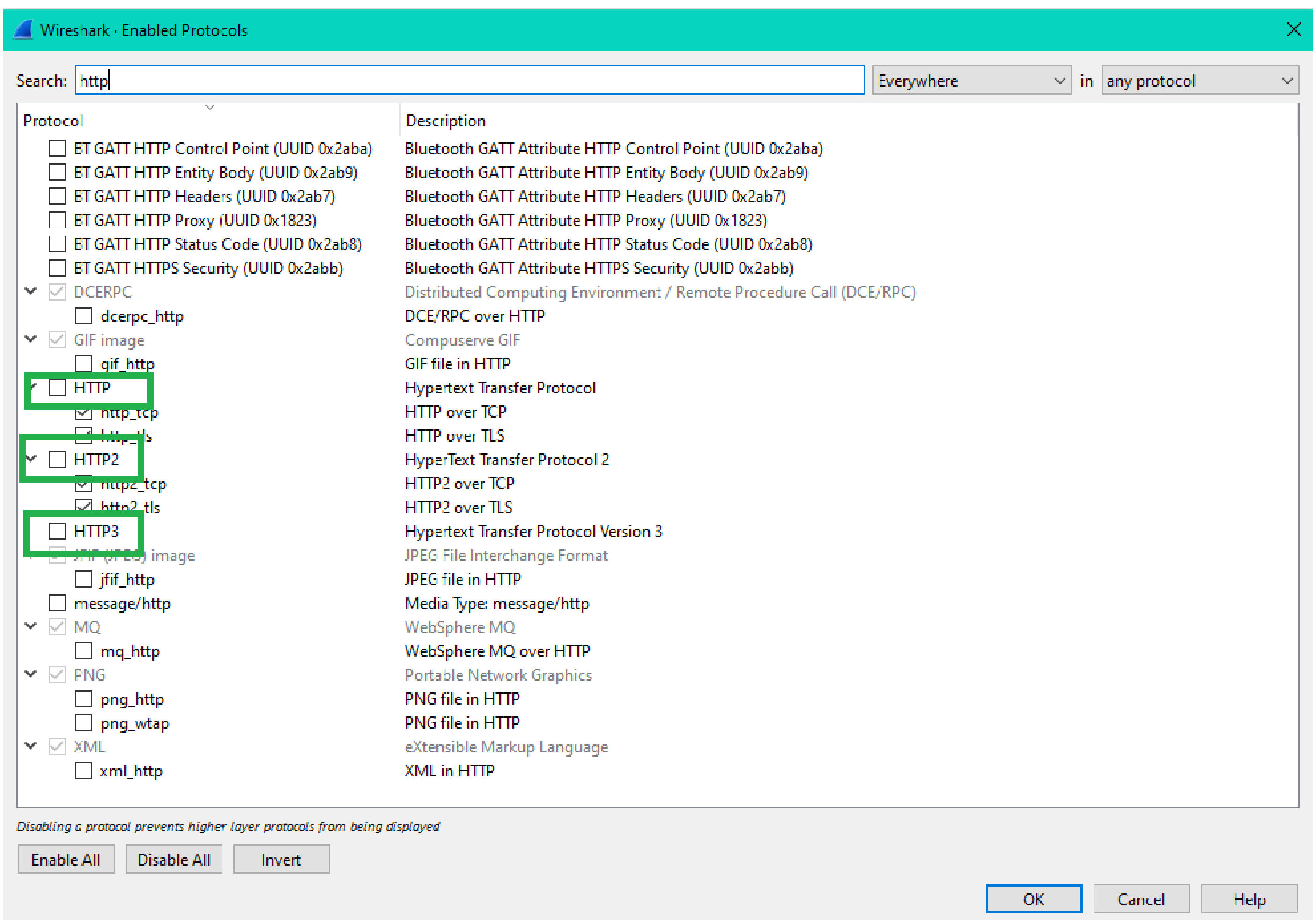
<u>My client computer (source):</u>

Source IP Address: 192.168.1.5

Source TCP port number: 1952



The Task:

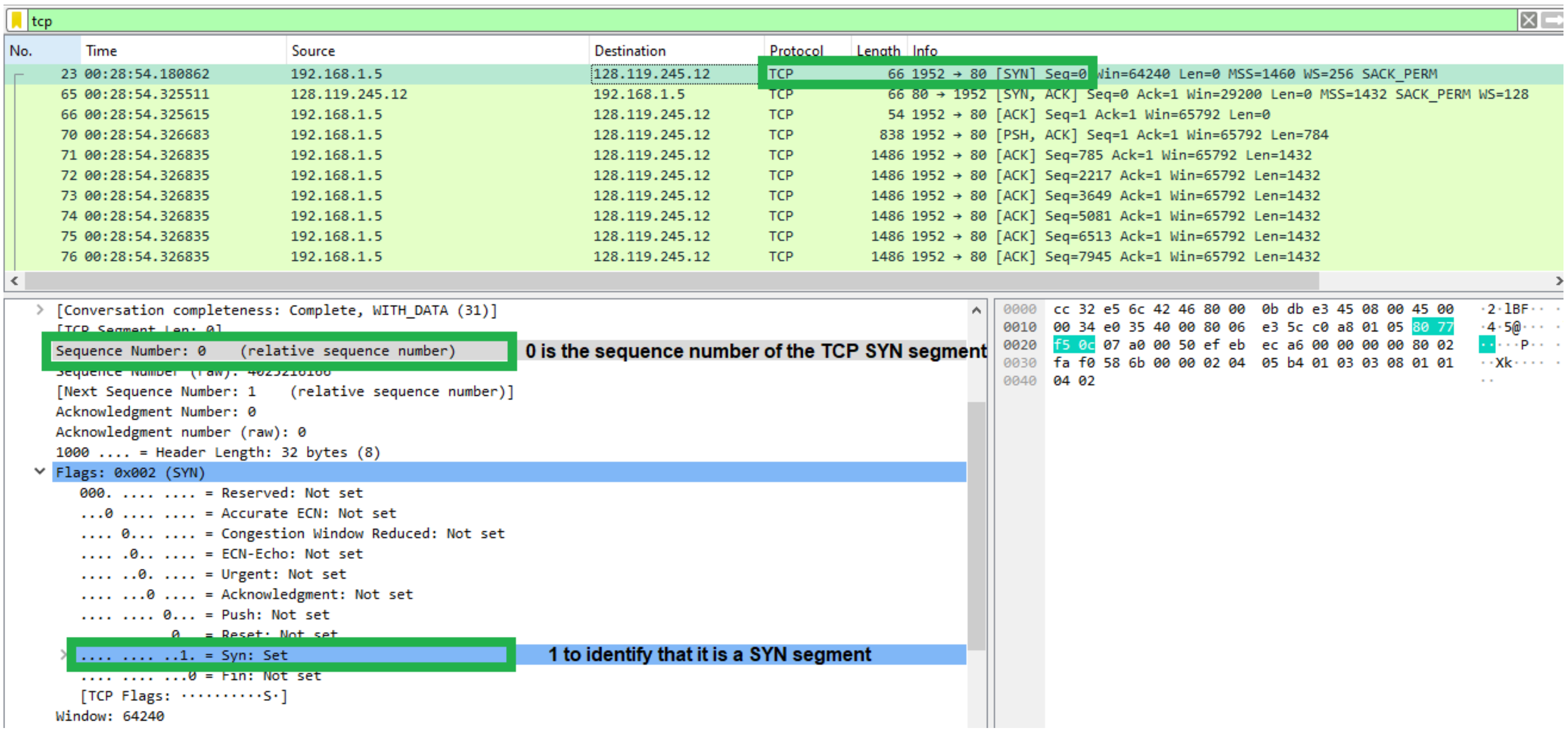


<u>Section 3</u>: TCP Basics

The Questions & Answers:

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?

- **<u>Sequence number</u>** of the TCP SYN segment is equal to 0 here and it is used to initiate the TCP connection between the client computer and the server.
- **<u>The SYN flag</u>** is equal to 1 (set) and it indicates that this segment is a SYN segment.

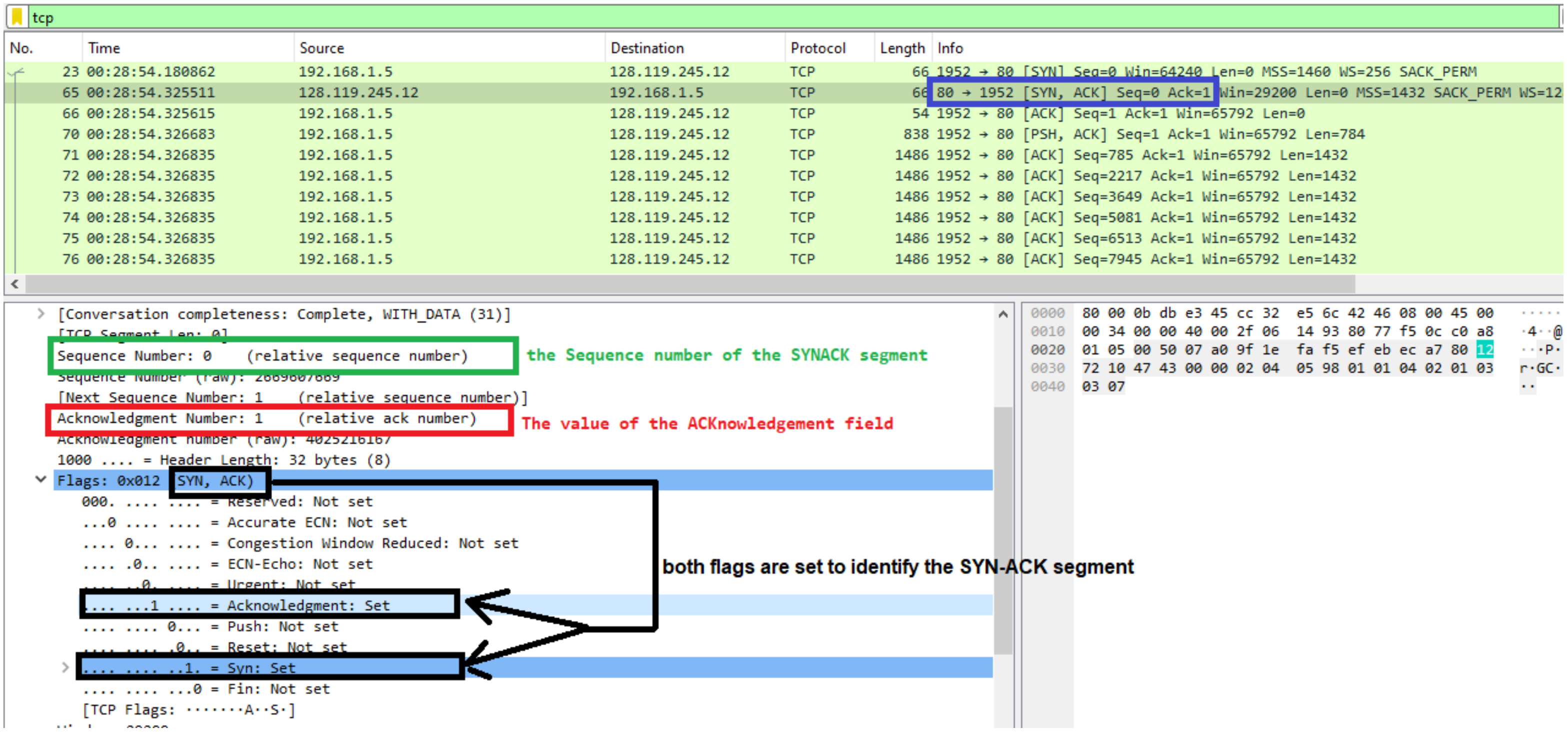


5. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK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

<u>Sequence number</u> of the SYNACK segment from server to the client in replying to the SYN is equal to 0.

<u>The value of the ACKnowledgement field</u> in the SYN-ACK segment is 1. The value of the ACKnowledgement field in the SYN-ACK segment is determined by the server **by adding 1** to the initial sequence number of SYN segment from the client (i.e. the sequence number of the original SYN segment initiated by the client is 0 as shown before).

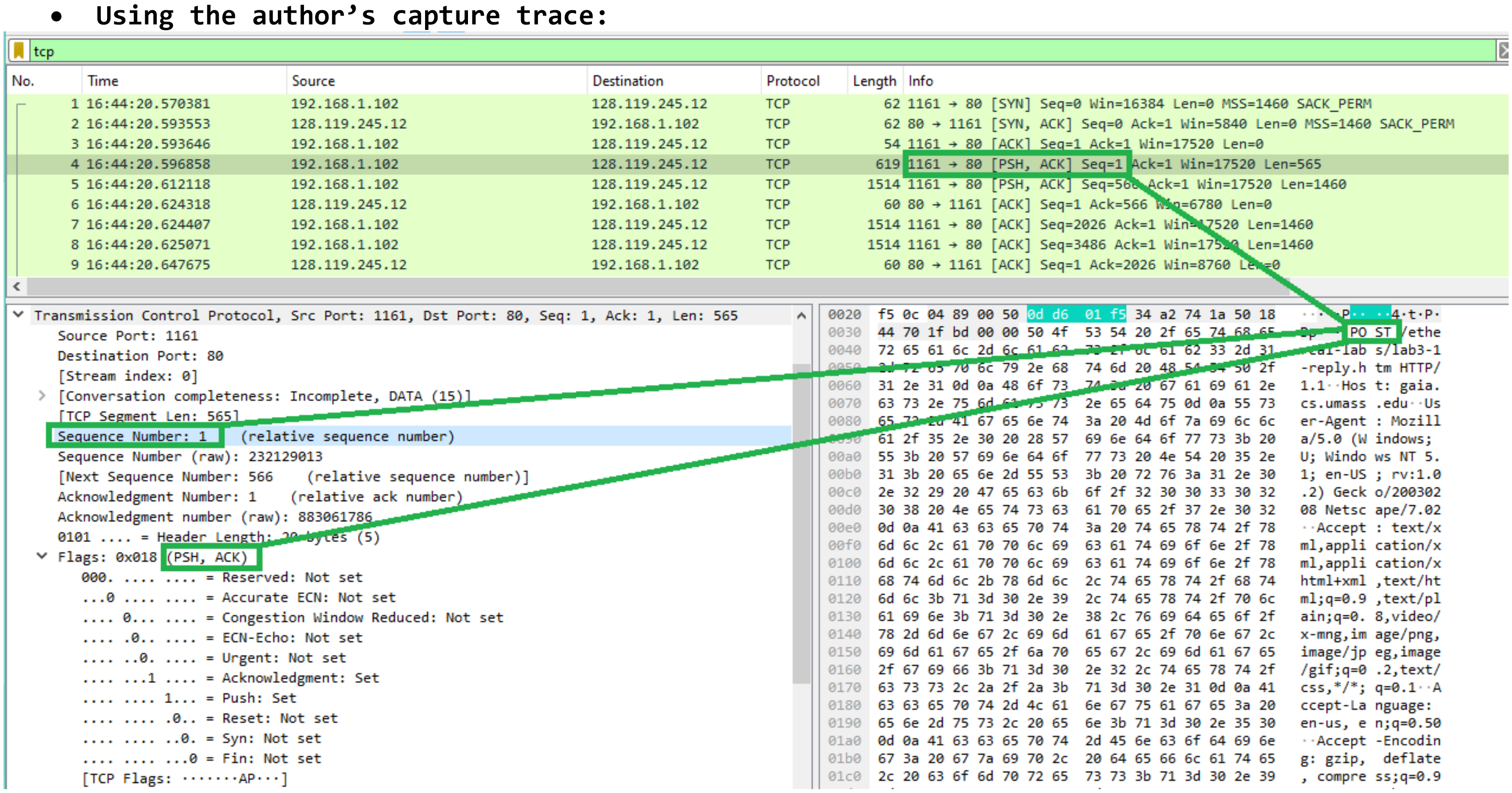
<u>To indicate/identify the segment as a SYN-ACK segment</u> : the SYN flag and Acknowledgement flag in the segment are both set to 1.



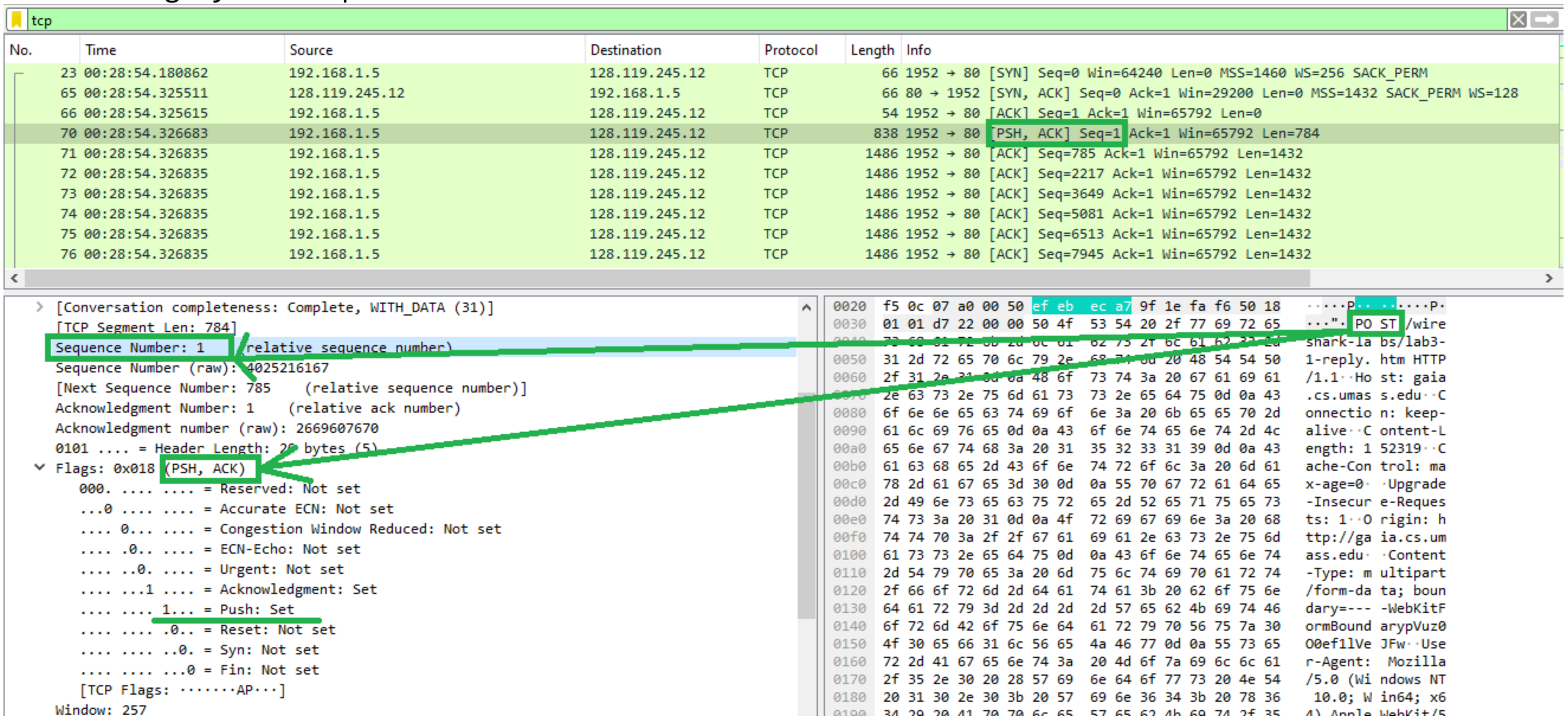
6. What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.

After investigating packets , the below packet contains POST method, its sequence number is 1.

- **Using the author's capture trace:**



- Using my own capture trace:



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 239 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 239 for all subsequent segments.

Note: Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the "listing of captured packets" window that is being sent from the client to the gaia.cs.umass.edu server. Then select: Statistics->TCP Stream Graph->Round Trip Time Graph.

The HTTP POST segment is considered as the first segments from the client computer to the server in the TCP connection. Segments 1 – 6 are No. 4, 5, 7, 8, 10, and 11 in this trace respectively.

The ACKs of segments 1 – 6 are No. 6, 9, 12, 14, 15, and 16 in this trace.

1	16:44:20.570381	192.168.1.102	128.119.245.12	TCP	62	1161 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM
2	16:44:20.593553	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	16:44:20.593646	192.168.1.102	128.119.245.12	TCP	54	1161 → 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0
4	16:44:20.596858	192.168.1.102	128.119.245.12	TCP	619	1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565
5	16:44:20.612118	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [PSH, ACK] Seq=566 Ack=1 Win=17520 Len=1460
6	16:44:20.624318	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=566 Win=6780 Len=0
7	16:44:20.624407	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460
8	16:44:20.625071	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460
9	16:44:20.647675	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=2026 Win=8760 Len=0
10	16:44:20.647786	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460
11	16:44:20.648538	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460
12	16:44:20.694466	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=3486 Win=11680 Len=0
13	16:44:20.694566	192.168.1.102	128.119.245.12	TCP	1201	1161 → 80 [PSH, ACK] Seq=7866 Ack=1 Win=17520 Len=1147
14	16:44:20.739499	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=4946 Win=14600 Len=0
15	16:44:20.787680	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=6406 Win=17520 Len=0
16	16:44:20.838183	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=7866 Win=20440 Len=0

The sequence number , sending time, and the received time of ACKs are in the following table:

Seq num	Sent time	ACK time	RTT (seconds)
1	0.026477	0.053937	0.02746
566	0.041737	0.077294	0.035557
2026	0.054026	0.124085	0.070059
3486	0.054690	0.169118	0.11443
4946	0.077405	0.217299	0.13989
6406	0.078157	0.267802	0.18964

$$RTT_{est_i} = \frac{7}{8} RTT_{est_{(i-1)}} + \frac{1}{8} RTT_{sample}$$

EstimatedRTT after the receipt of the ACK of segment 1:
EstimatedRTT = RTT for Segment 1 = 0.02746 seconds

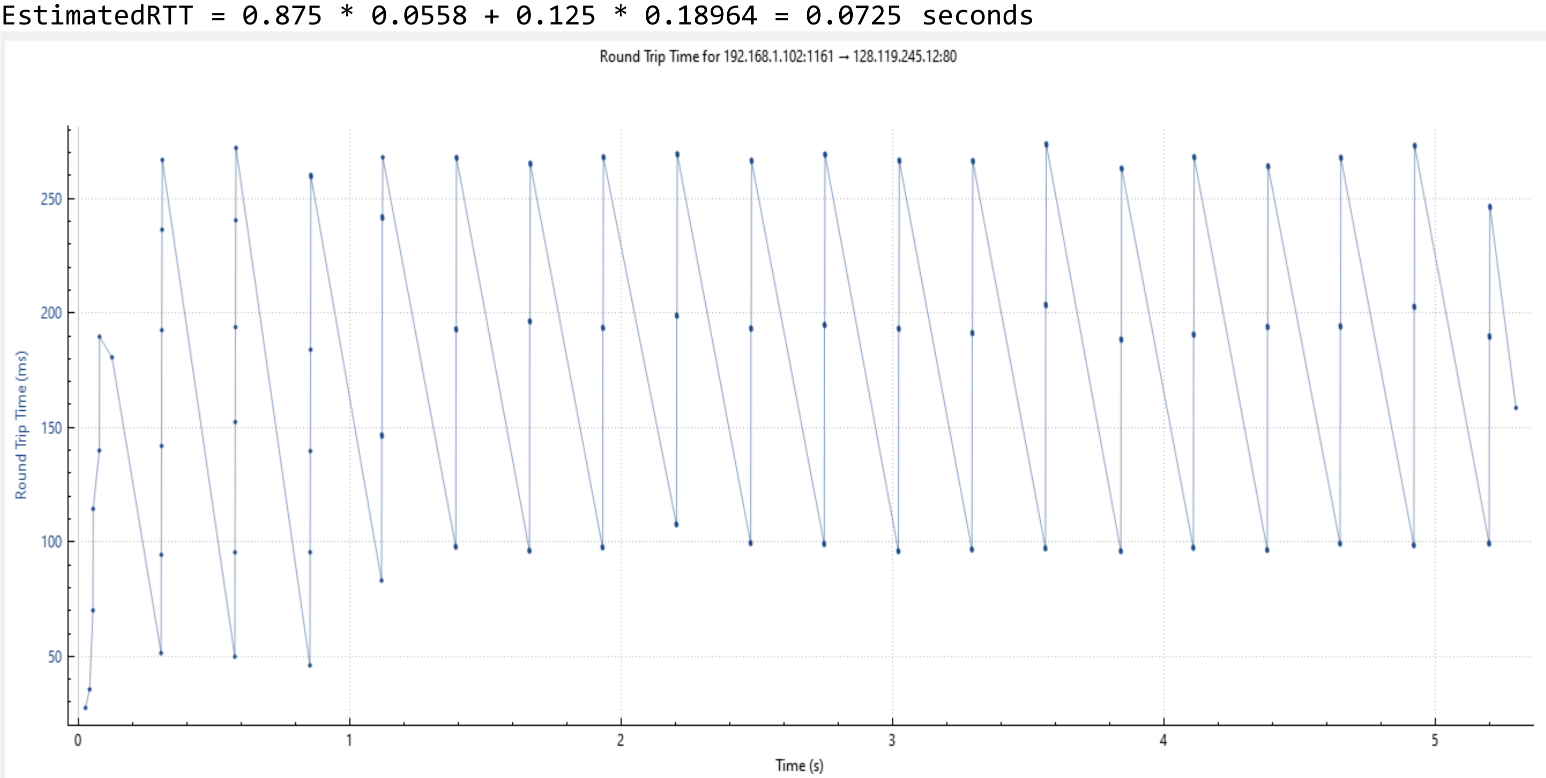
EstimatedRTT after the receipt of the ACK of segment 2:
EstimatedRTT = 0.875 * 0.02746 + 0.125 * 0.035557 = 0.0285 seconds

EstimatedRTT after the receipt of the ACK of segment 3:
EstimatedRTT = 0.875 * 0.0285 + 0.125 * 0.070059 = 0.0337 seconds

EstimatedRTT after the receipt of the ACK of segment 4:
EstimatedRTT = 0.875 * 0.0337+ 0.125 * 0.11443 = 0.0438 seconds

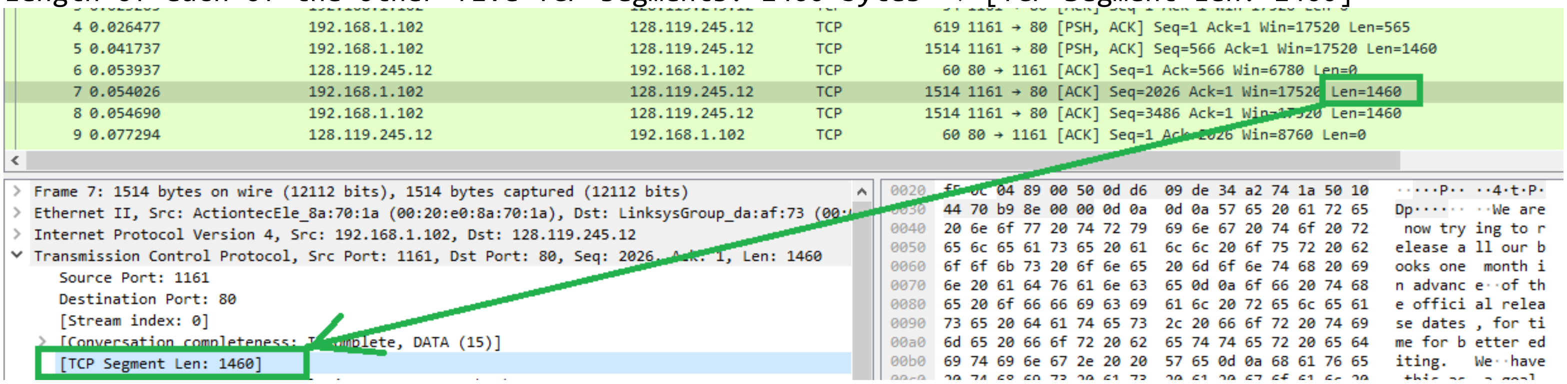
EstimatedRTT after the receipt of the ACK of segment 5:
EstimatedRTT = 0.875 * 0.0438 + 0.125 * 0.13989 = 0.0558 seconds

EstimatedRTT after the receipt of the ACK of segment 6:
EstimatedRTT = 0.875 * 0.0558 + 0.125 * 0.18964 = 0.0725 seconds



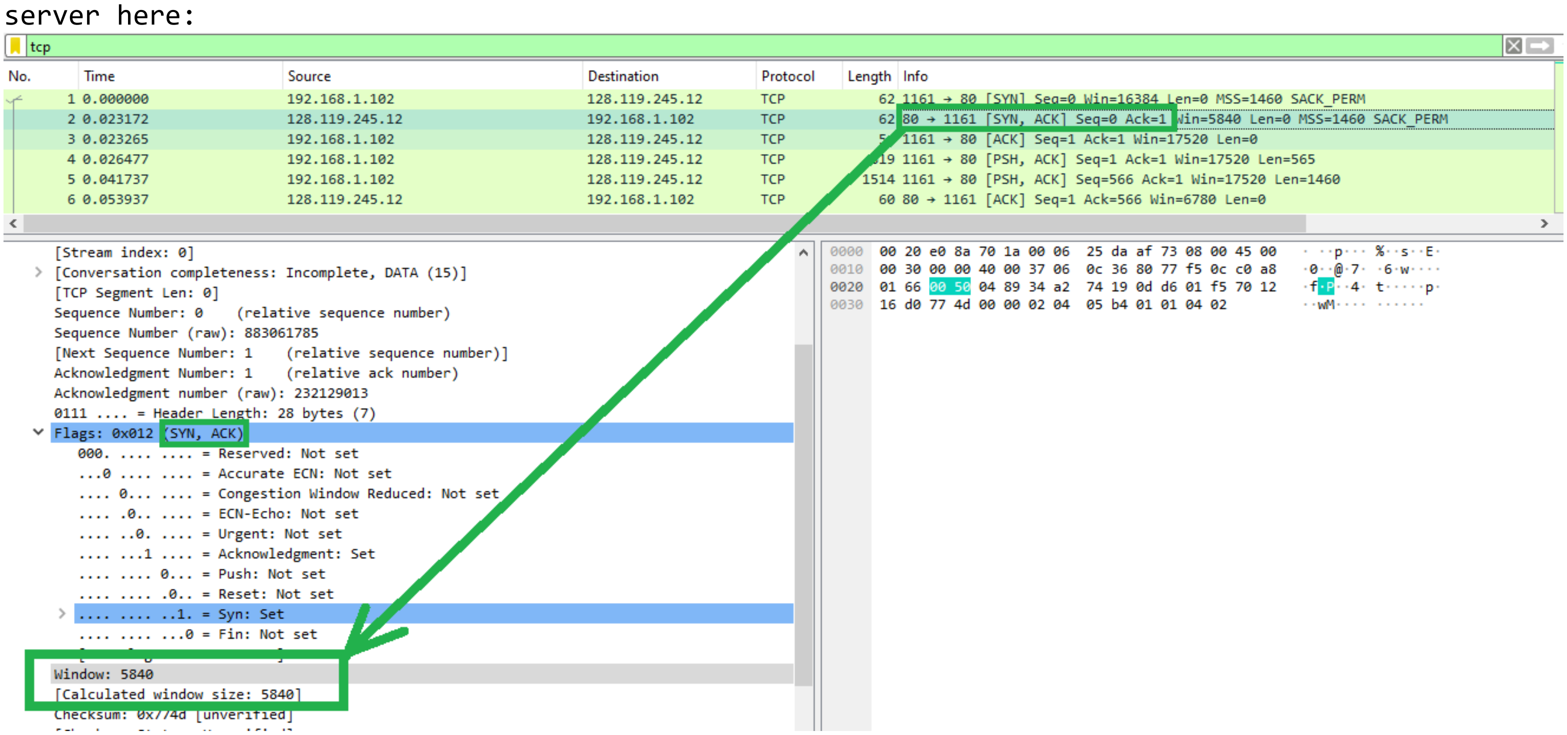
8. What is the length of each of the first six TCP segments?

The first TCP segment length of the HTTP POST: 565 bytes -> [TCP Segment Len: 565]
Length of each of the other five TCP segments: 1460 bytes -> [TCP Segment Len: 1460]



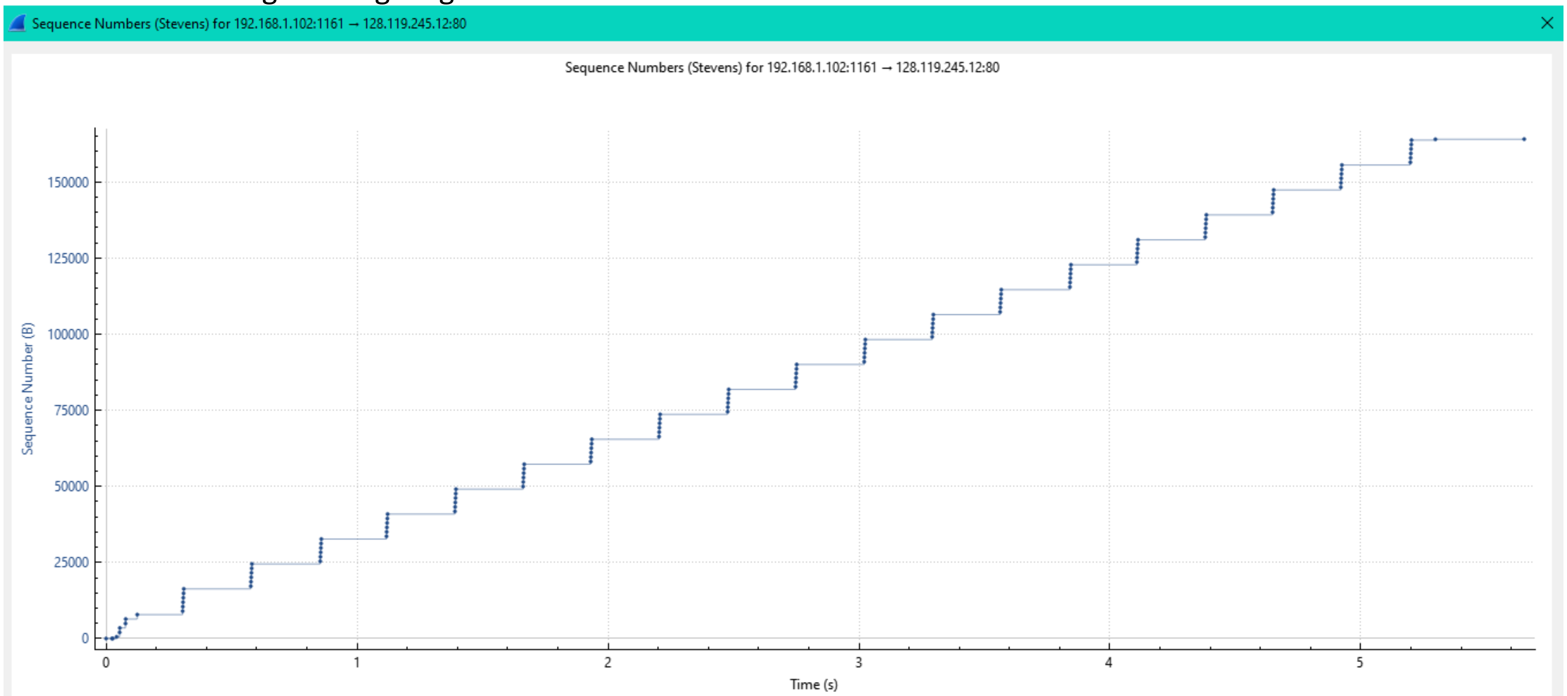
9. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

The minimum amount of buffer space (receiver window) advertised at gaia.cs.umass.edu for the entire trace is 5840 bytes, which shows in the first acknowledgement from the server(in the SYN-ACK).
This receiver window grows steadily until a maximum receiver buffer size of 62780 bytes.
Therefore, the sender will never be throttled due to lacking of receiver buffer space by inspecting this trace.
Because, the size of what the Tx sends is always smaller than the allowed windows size by the server here:



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 is no retransmitted segments in the trace file since in the time sequence graph (stevens), all sequence numbers are monotonically increasing. But, if there is retransmission there will be a drop in the curve, or the Seq number will decrease suddenly then returned to increase again.so to conclude, There are no retransmitted segments in the trace file. We can verify this by checking the sequence numbers of the TCP segments in the trace file or using graph.If there is a retransmitted segment, the sequence number of this retransmitted segment should be smaller than those of its neighboring segments.

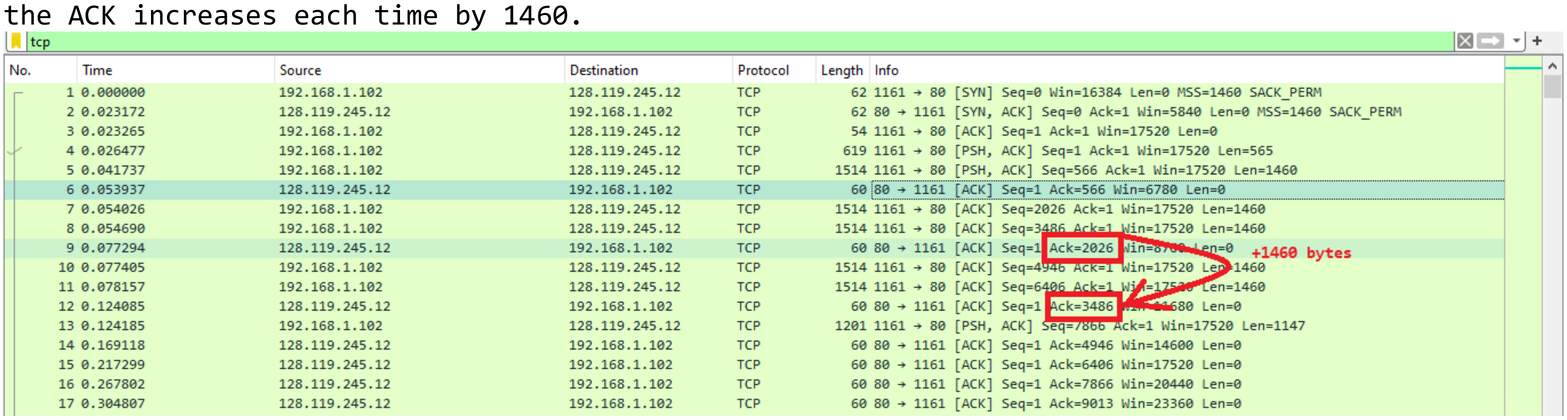


11. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 247 in the text).

In the first acknowledgement, the Rx ACK a segment with size 566 bytes, but all the other sizes were 1460 bytes. So, for example in
The acknowledged sequence numbers of the first 6 ACKs are listed as follows.

	acknowledged sequence number	acknowledged data size
ACK 1	566	566
ACK 2	2026	1460
ACK 3	3486	1460
ACK 4	4946	1460
ACK 5	6406	1460
ACK 6	7866	1460

The difference between the acknowledged sequence numbers of two consecutive ACKs indicates the data received by the server between these two ACKs. By inspecting the amount of acknowledged data size by each ACK, there are a lot of cases where the receiver is ACKing every other segment. For example, after the first TCP ACK segment , We can conclude that the ACK increases each time by 1460.


tcp

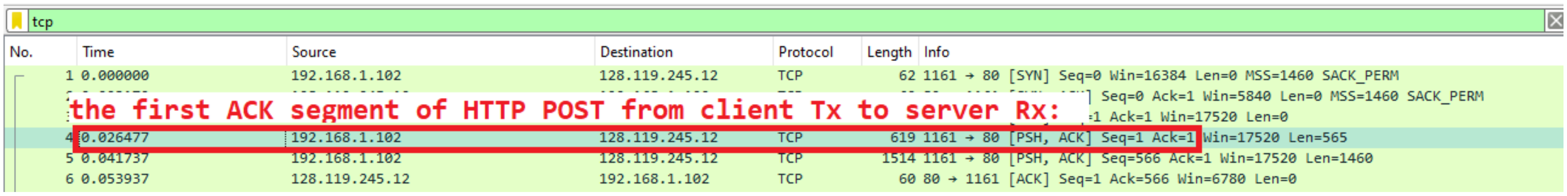
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.102	128.119.245.12	TCP	62	1161 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM
2	0.023172	128.119.245.12	192.168.1.102	TCP	62	80 → 1161 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM
3	0.023265	192.168.1.102	128.119.245.12	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
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
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
8	0.054690	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460
9	0.077294	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 $\boxed{\text{Ack=2026}}$ 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
11	0.078157	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460
12	0.124085	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 $\boxed{\text{Ack=3486}}$ 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
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

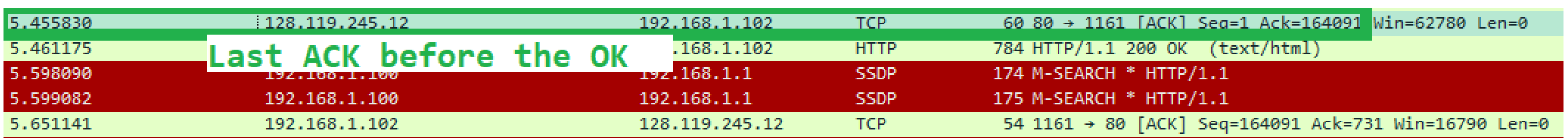
+1460 bytes

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

$$Throughput = \frac{Data}{Time}$$

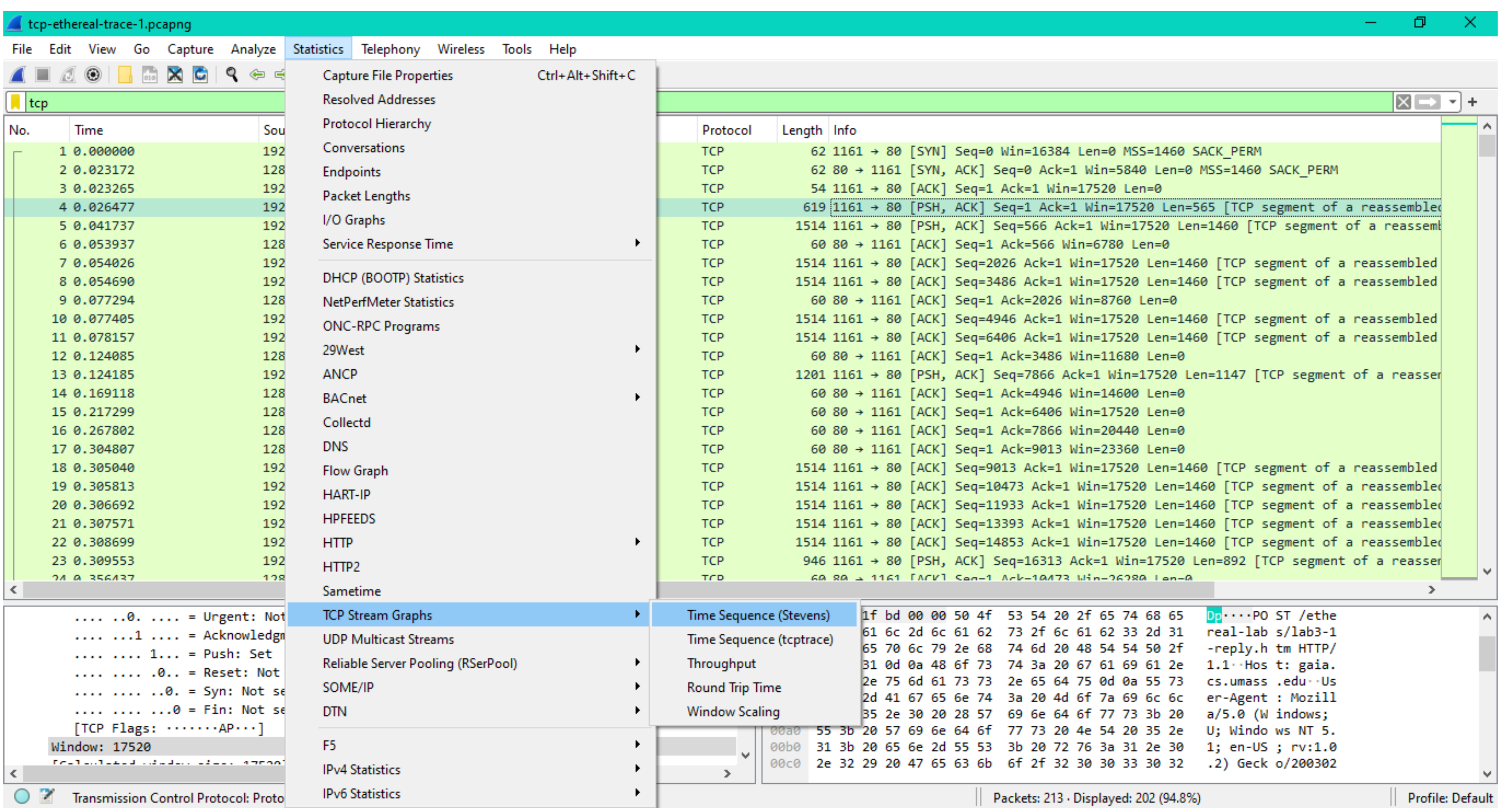
Data = ACK-Seq num of Last ACK (before the HTTP OK)- Seq. of 1st ACK TCP Segment of HTTP POST from client to server= 164091-1 = 164090 bytes
Time = Time of Last ACK (before the HTTP OK)- Time of First ACK TCP Seg of HTTP POST= 5.455830-0.026477= 5.4294 sec
the throughput for the TCP connection = 164090/5.4294 = 30.222 KByte/sec.

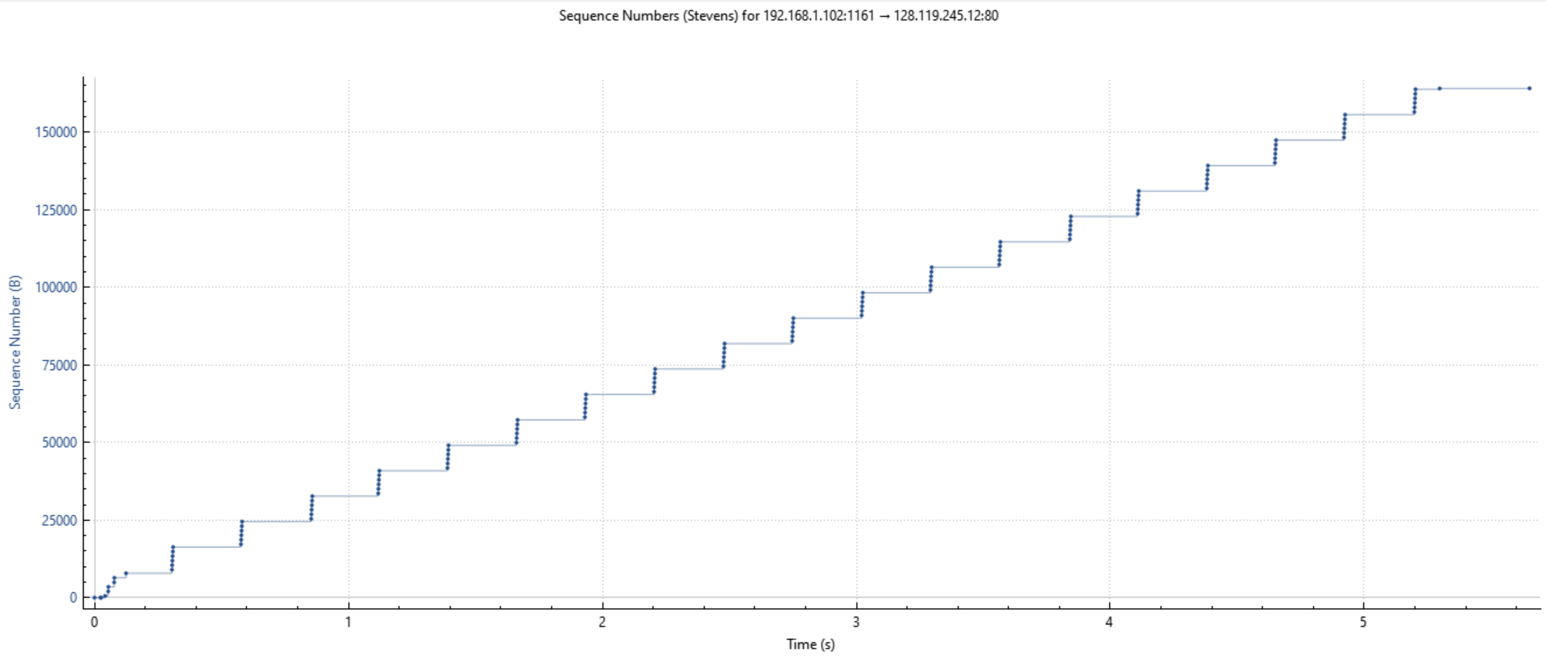




Section 4: TCP congestion control in action

The Task:

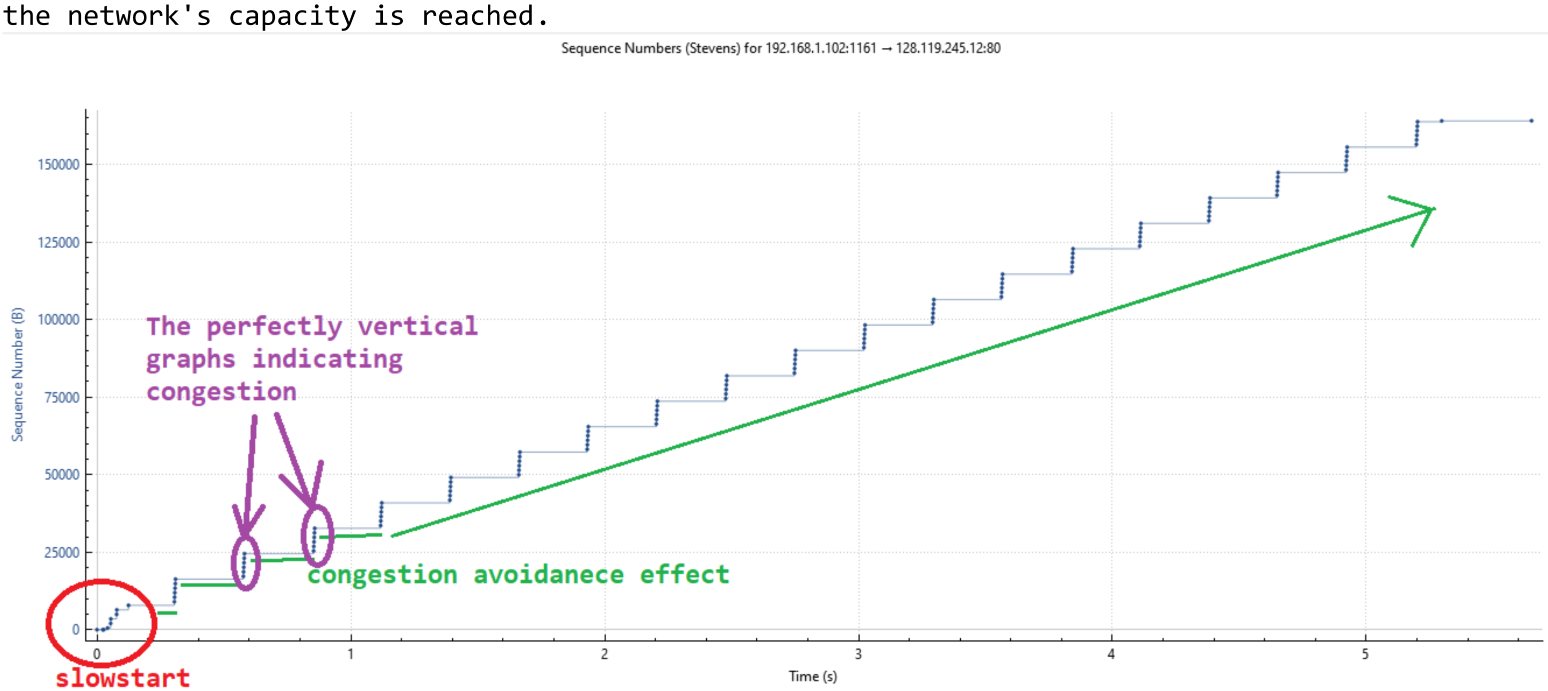




The Questions & Answers:

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 : prevents a network from becoming congested by regulating the amount of data that's sent over it. It negotiates the connection between a sender and receiver by defining the amount of data that can be transmitted with each packet, and slowly increases the amount of data until the network's capacity is reached.



Slow Start starts at the beginning Sequence Number = 566 and ends at Sequence Number = 7866
Congestion Avoidance starts to take place at Sequence Number = 7866.

Comment: This differs from the perfectly exponentially plotted slow start graphs seen earlier in the text, as the plotted graph is a lot more jagged and uneven, as well as the perfectly vertical graphs indicating congestion avoidance compared to the more gradual graphs shown in the text.

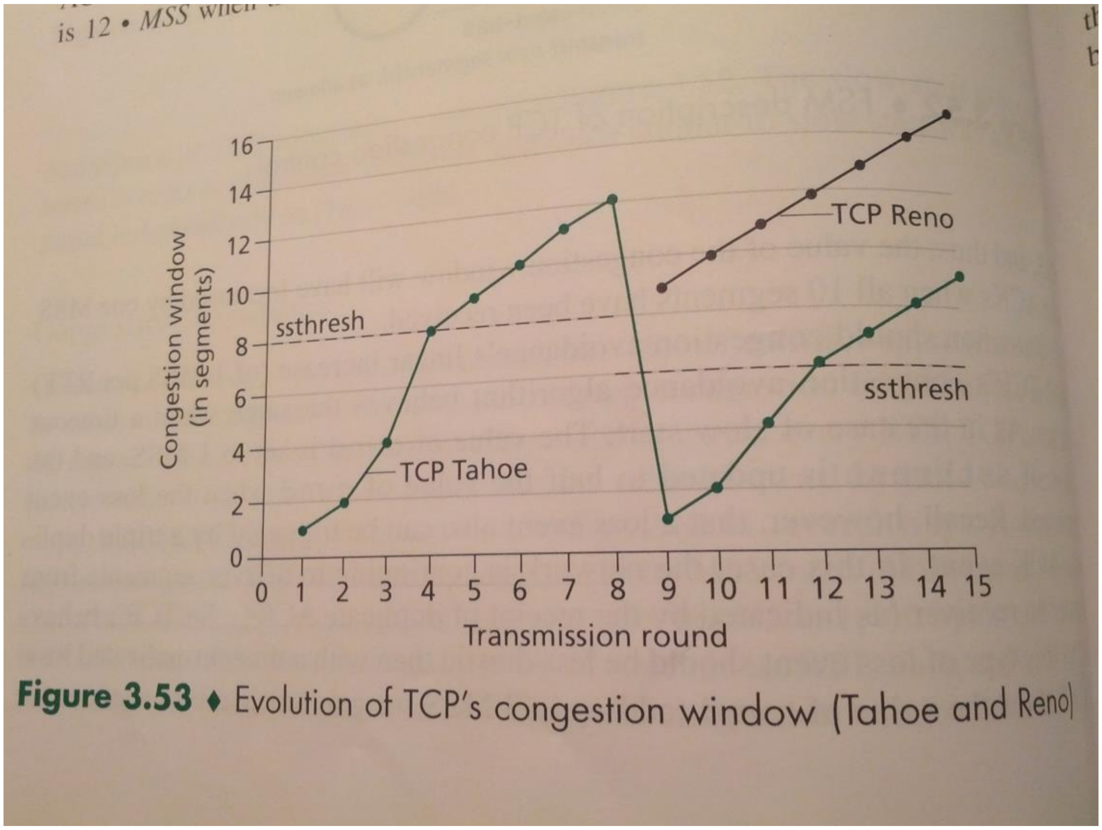


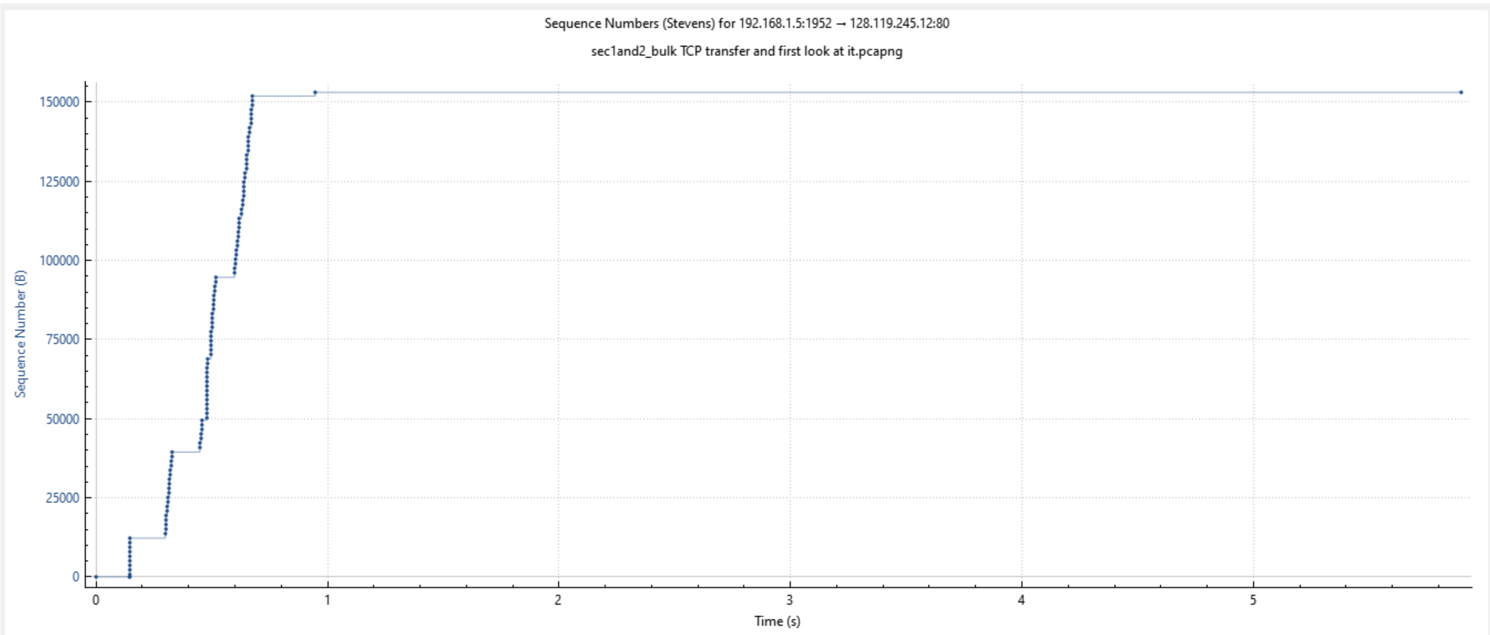
Figure 3.53 ♦ Evolution of TCP's congestion window (Tahoe and Reno)

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?

First solution:

Slow Start starts at Sequence Number = 785 and ends at Sequence Number = 12241

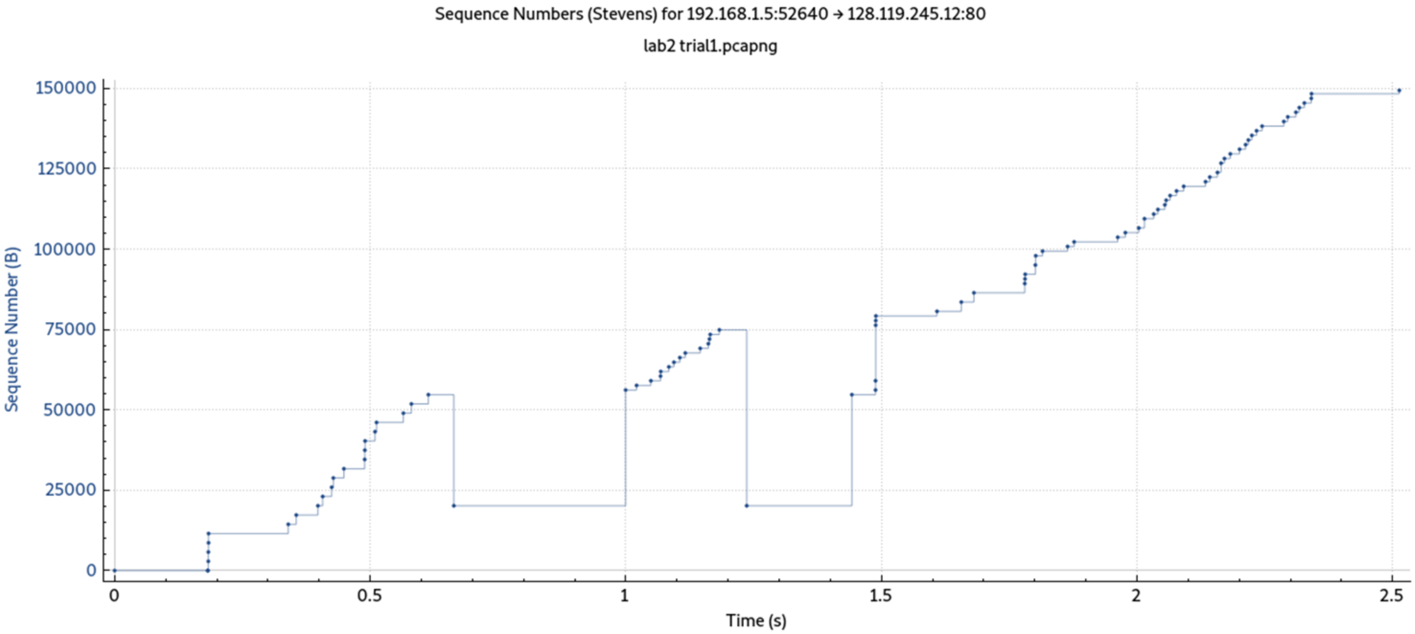
Congestion Avoidance starts at Sequence Number = 12241



Second solution:

Slow Start starts at Sequence Number = 566 and ends at Sequence Number = 7866

Congestion Avoidance starts at Sequence Number = 7866



Comment:

if there is retransmission there will be a drop in the curve, or the Seq number will decrease suddenly then returned to increase again.


thank you


BYE