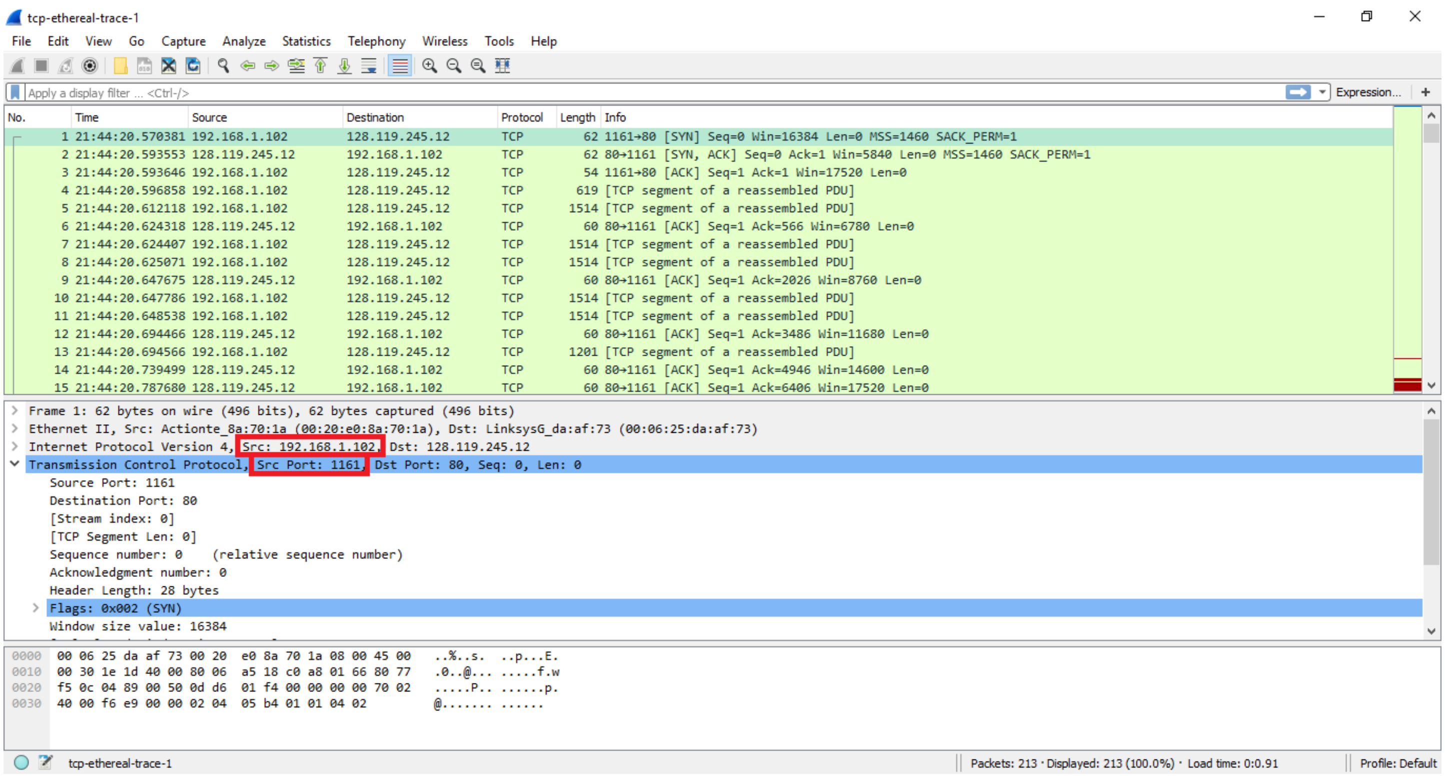
Jessica Spokoyny
Nov 5, 2016
CS 372 – Lab 3

Lab 3

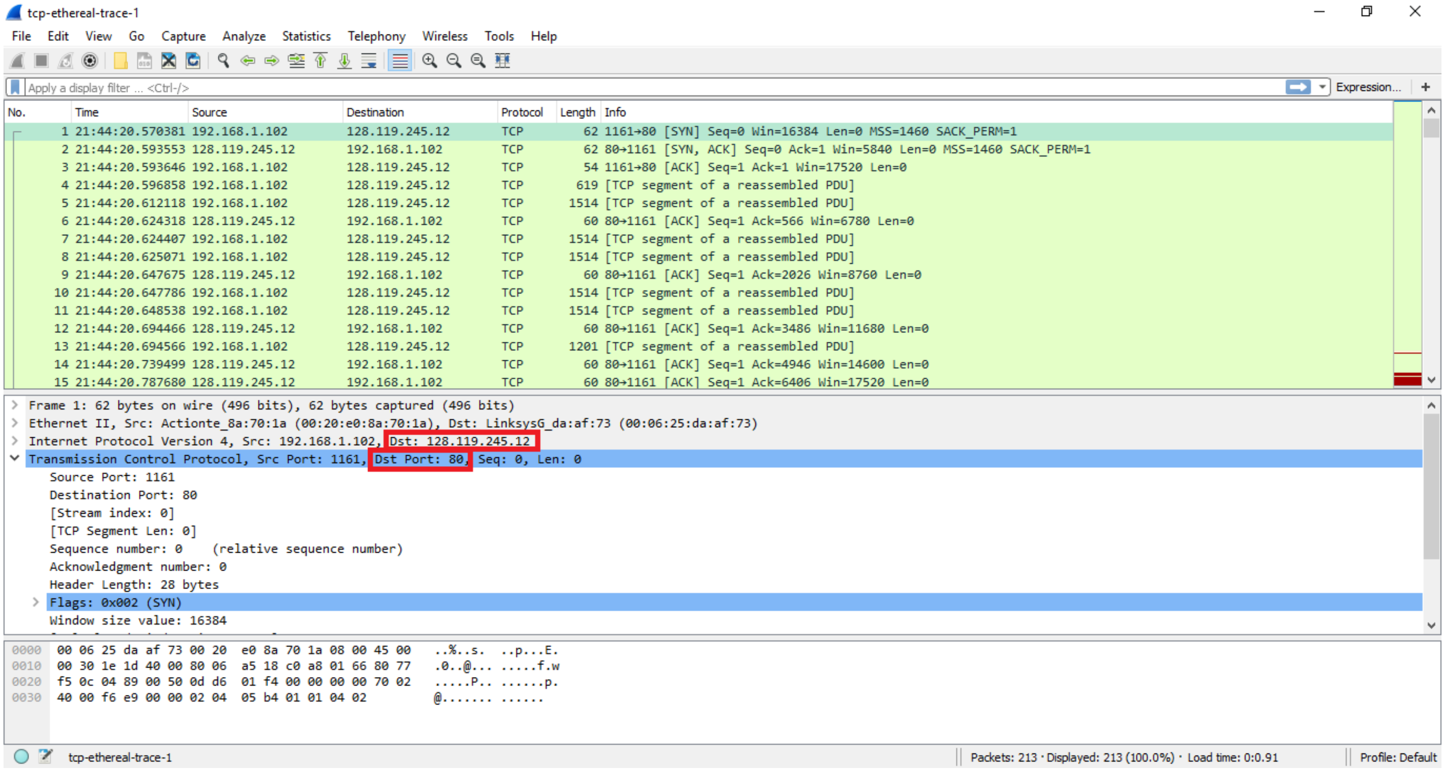
1. **What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? To answer this question, it's probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "details 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.**

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



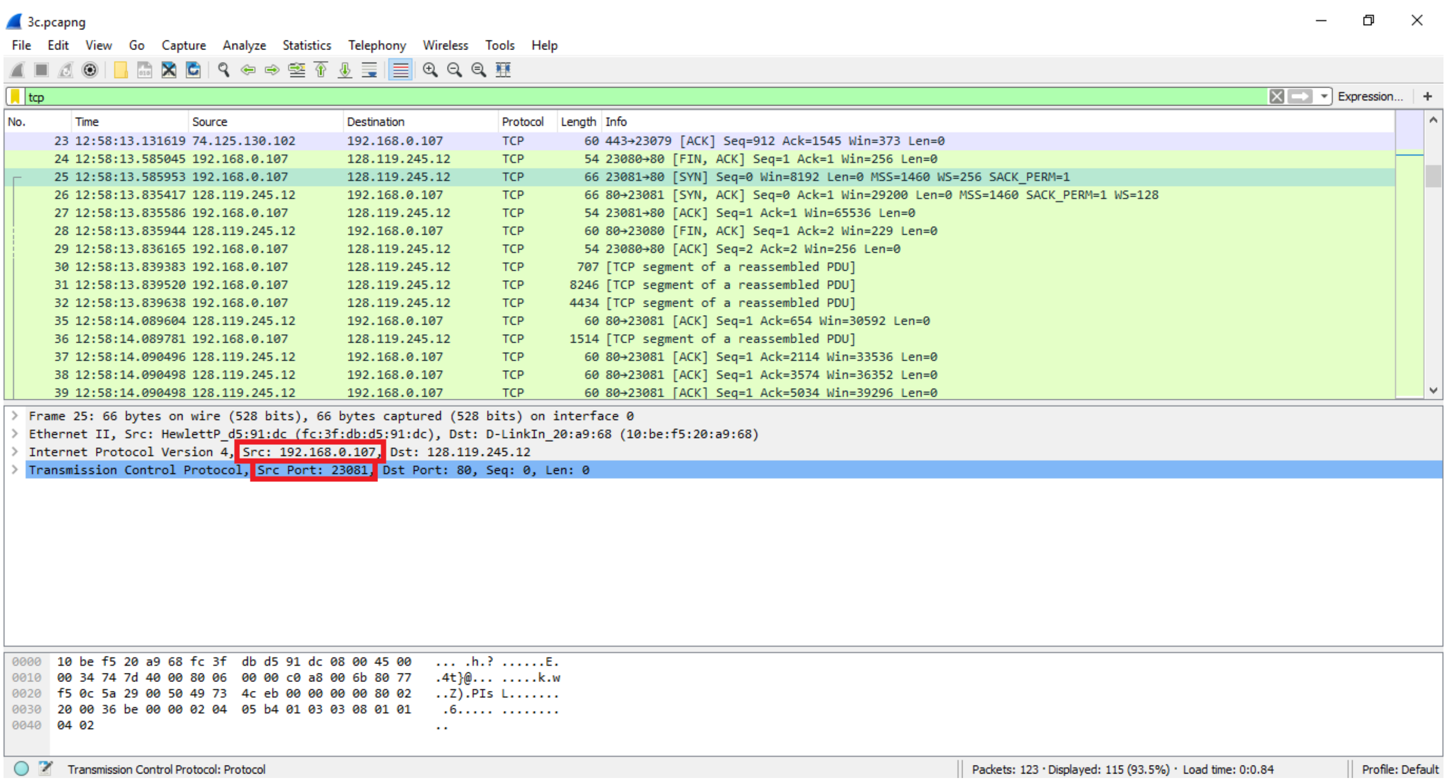
2. **What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?**

The IP address of gaia.cs.umass.edu is: 128.119.245.12 and the TCP port number used is 80



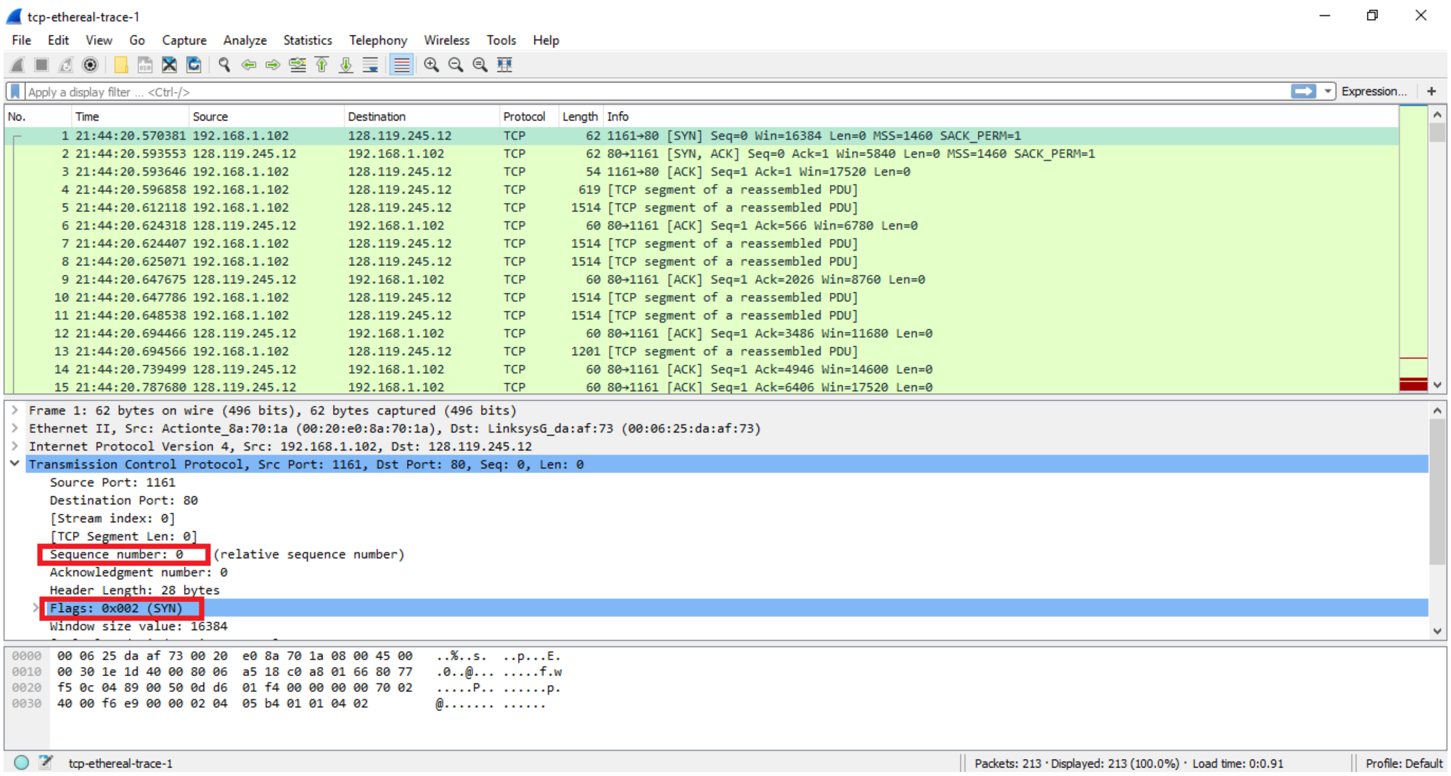
3. What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?

The IP address used by my client computer is: 192.168.0.107 and the TCP port number used by my client computer is 23081



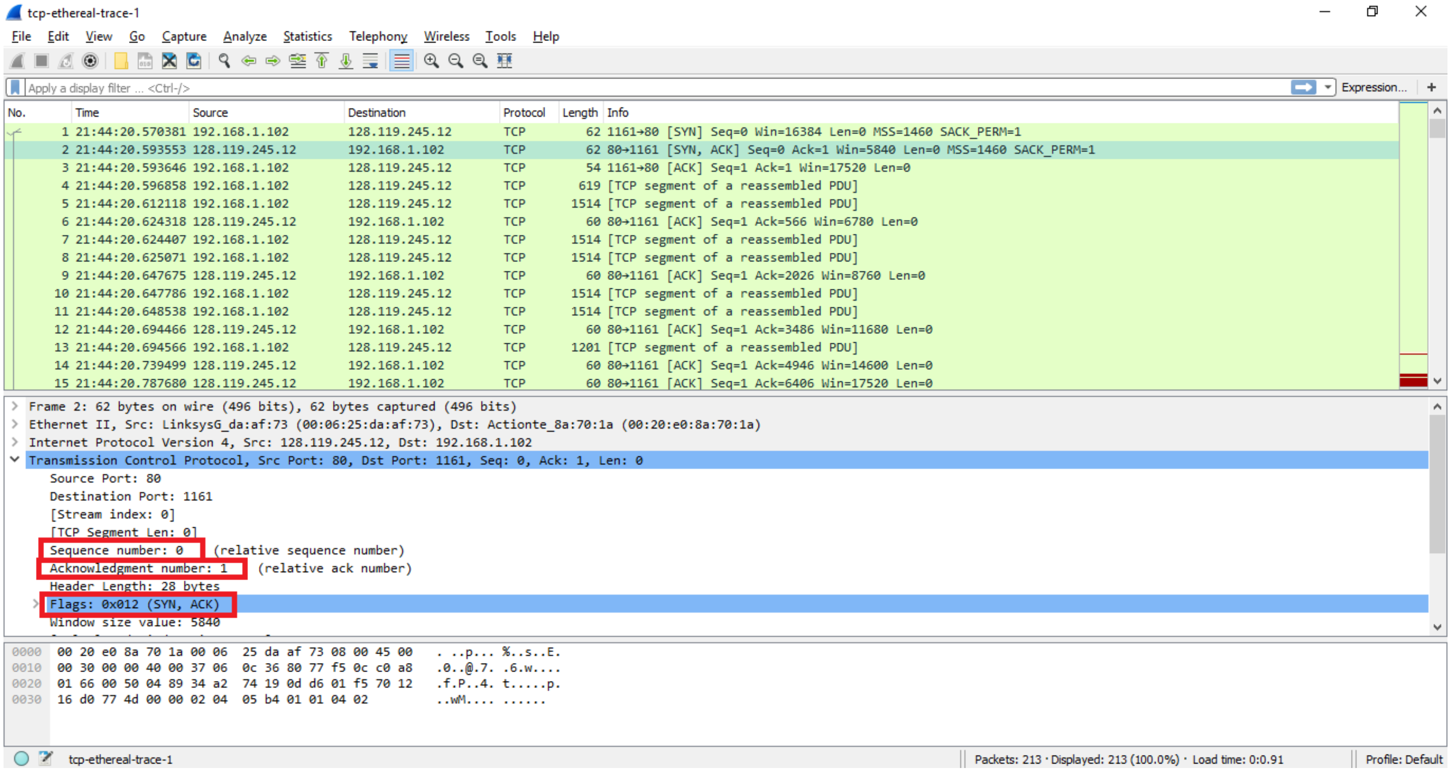
4. **What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?**

The sequence number of the TCP SYN segment that is used to initiate the TCP connection is 0. The segment contains a SYN flag which identifies it as a SYN segment.



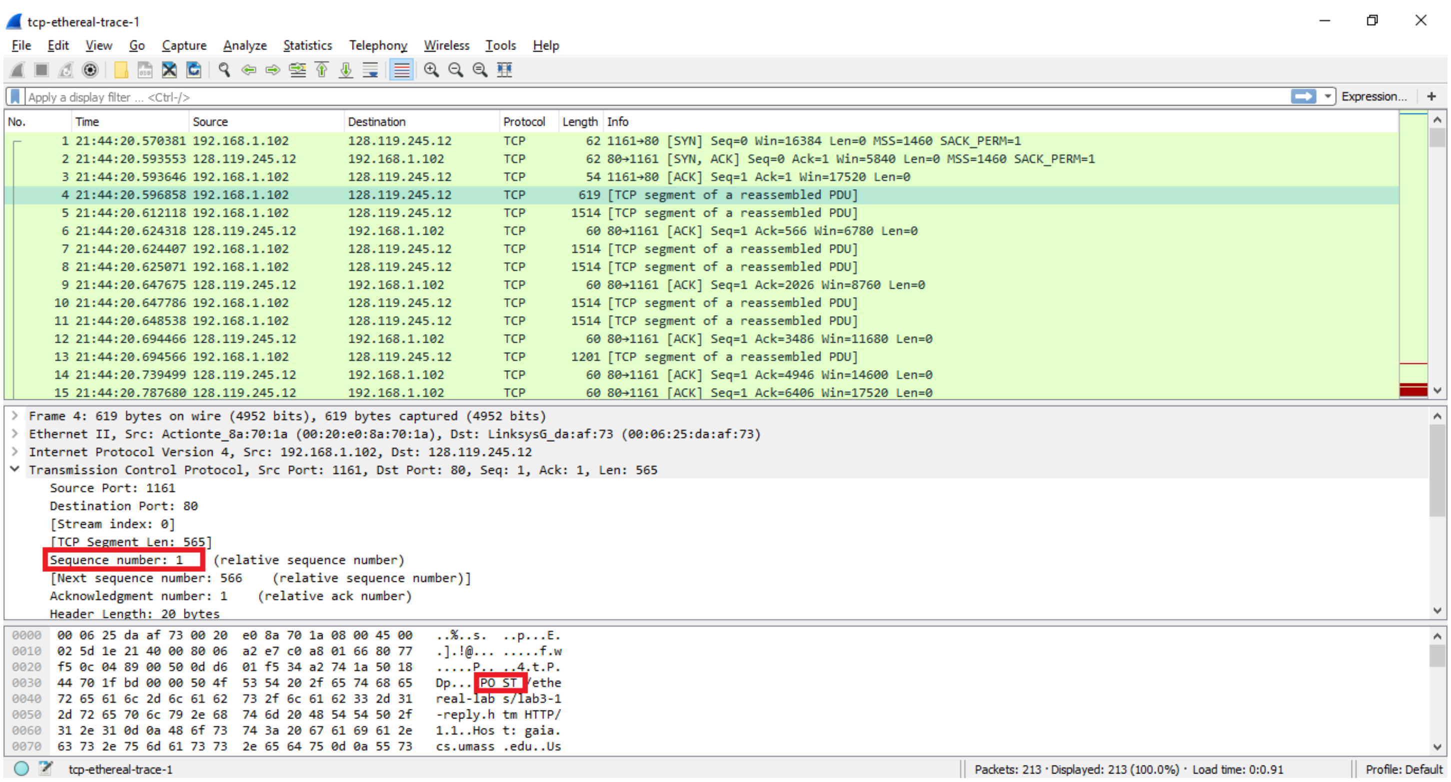
5. **What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a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

The sequence number of the SYNACK segment sent in reply to the SYN is 0. The value of the Acknowledgement field in the SYNACK segment is 1. This value was determined by taking the initial sequence number and adding 1. The segment contains SYN and ACK flags which identify it 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.**

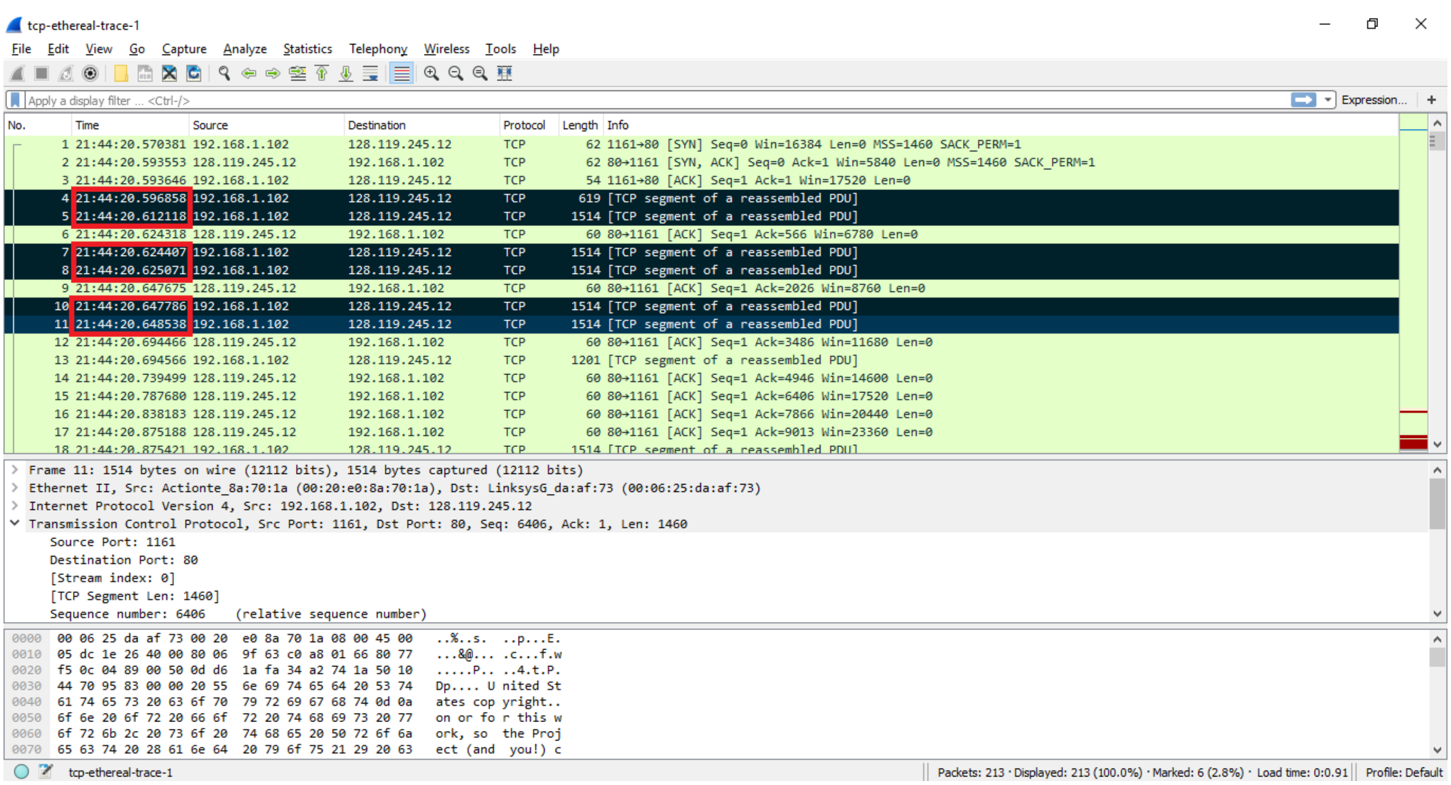
The sequence number of the TCP segment containing the HTTP POST command is 1.

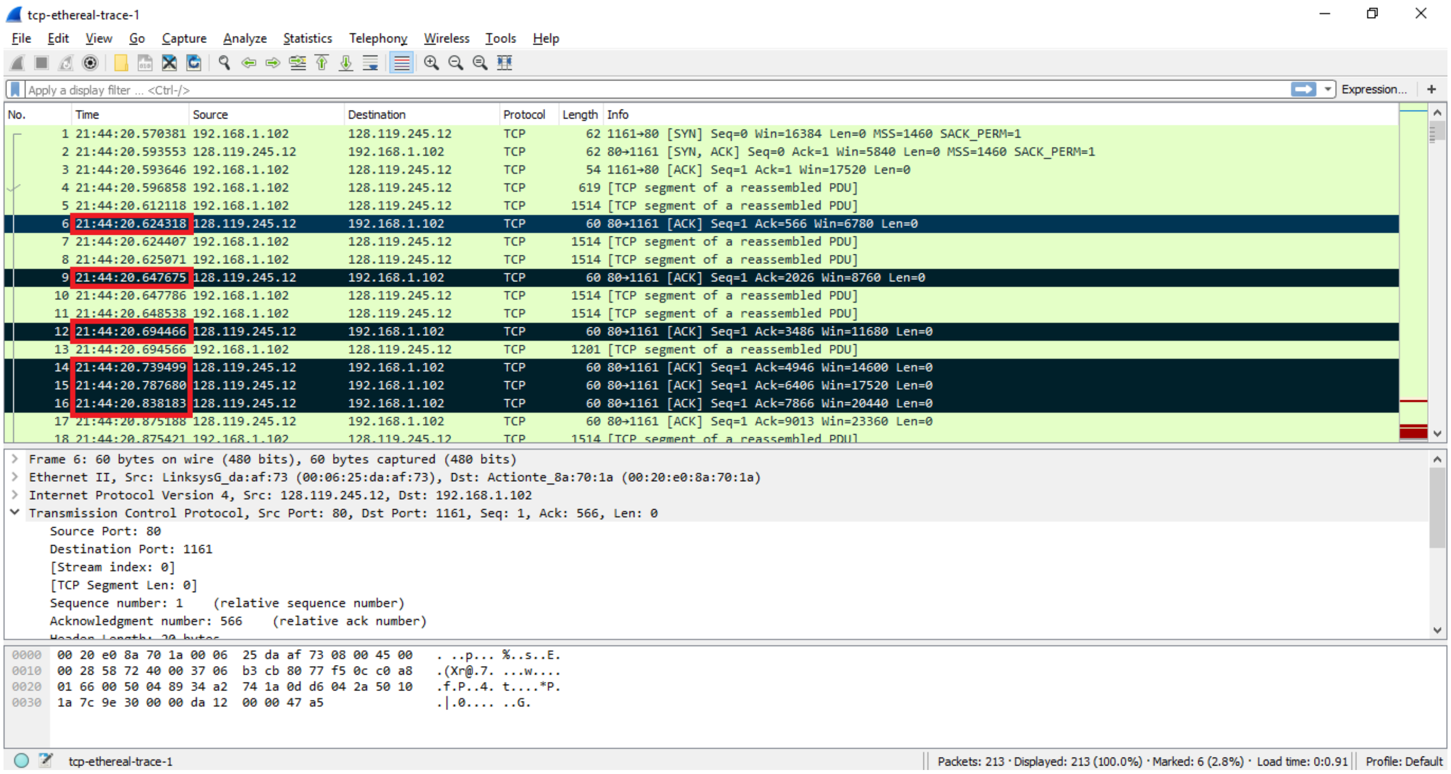


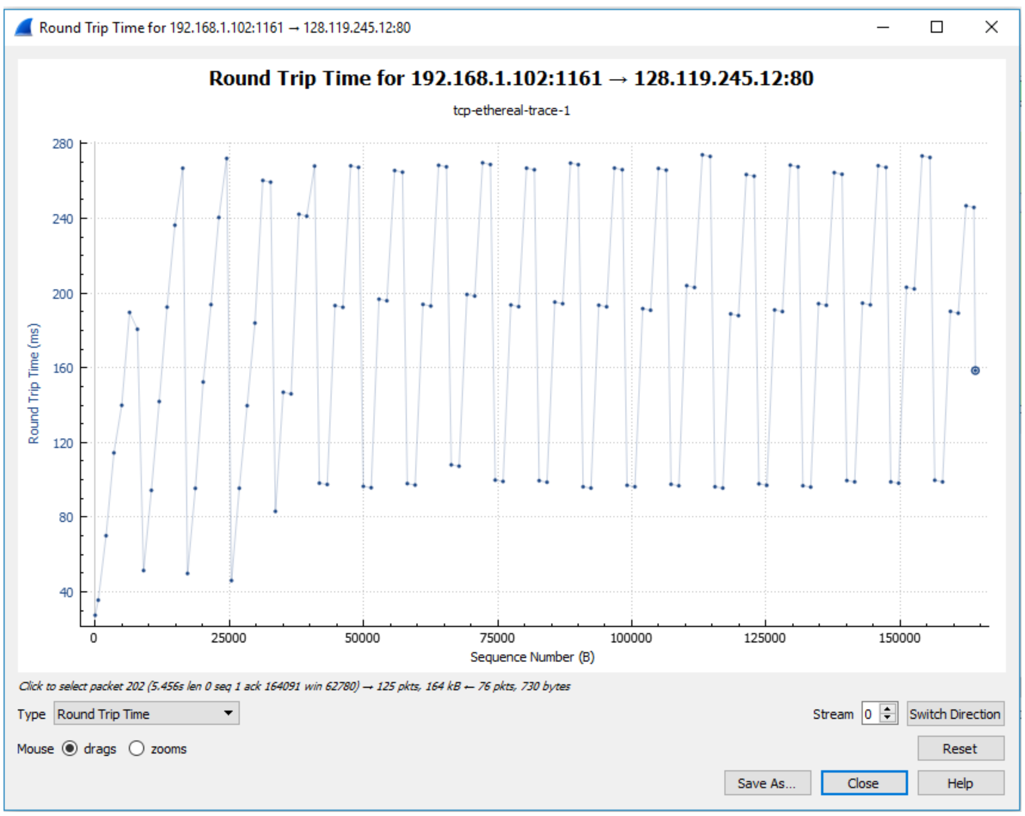
7. **Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received?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

	SEQ number	Sent time	ACK recvd time	RTT (seconds)
Segment 1	1	20.596858	20.624318	.027460
Segment 2	566	20.612118	20.647675	.035557
Segment 3	2026	20.624407	20.694466	.070059
Segment 4	3486	20.625071	20.739499	.114428
Segment 5	4946	20.647786	20.787680	.139894
Segment 6	6406	20.648538	20.838183	.189645

EstimatedRTT after receipt of the ACK of segment 1 = RTT for Segment 1 = .02746 seconds
EstimatedRTT after receipt of the ACK of segment 2 = .875 * .02746 + .125 * .035557 = .0285
EstimatedRTT after receipt of the ACK of segment 3 = .875 * .0285 + .125 * .070059 = .0337
EstimatedRTT after receipt of the ACK of segment 4 = .875 * .0337+ .125 * .114428 = .043791
EstimatedRTT after receipt of the ACK of segment 5 = .875 * .043791 + .125 * .139894 = .055804
EstimatedRTT after the receipt of the ACK of segment 6 = .875 * .055804 + .125 * .189645 = .072534

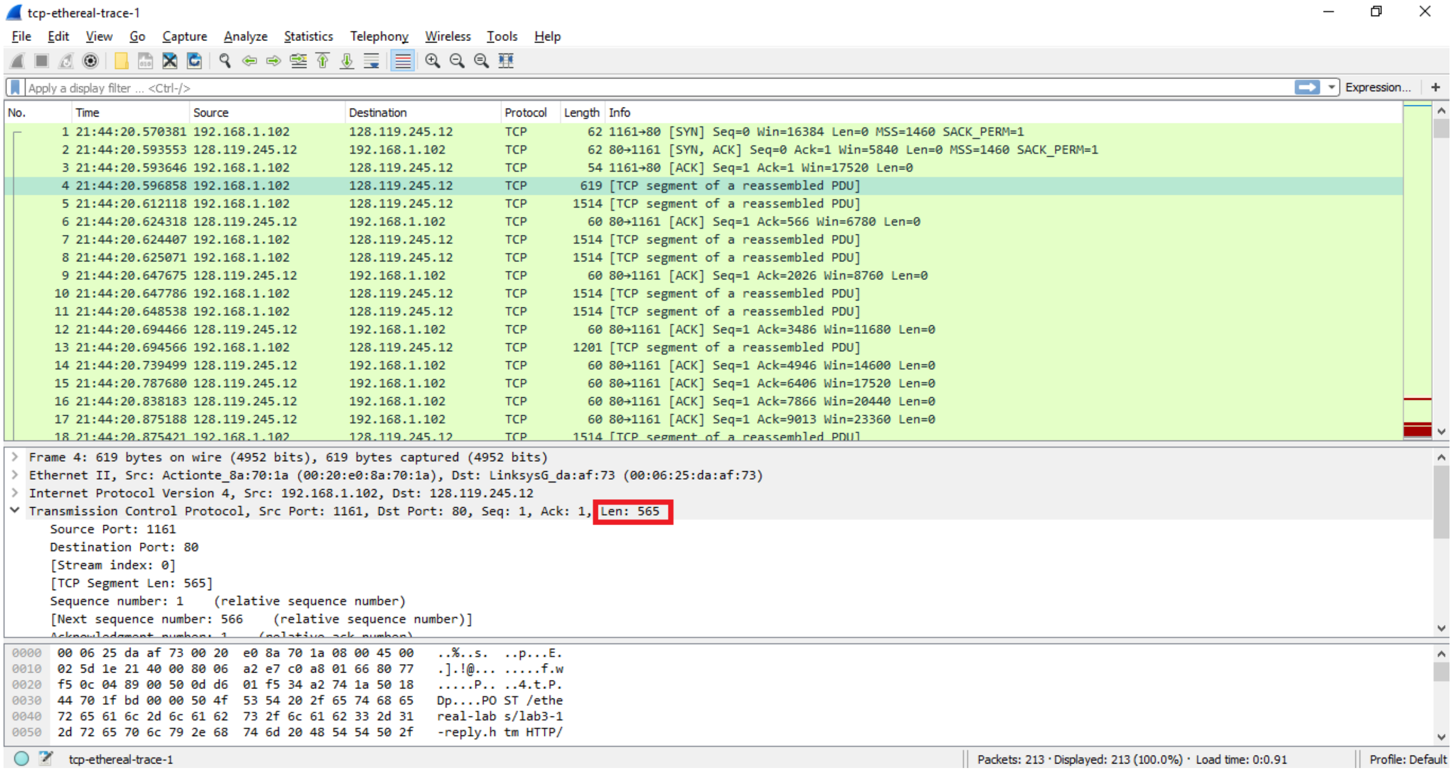


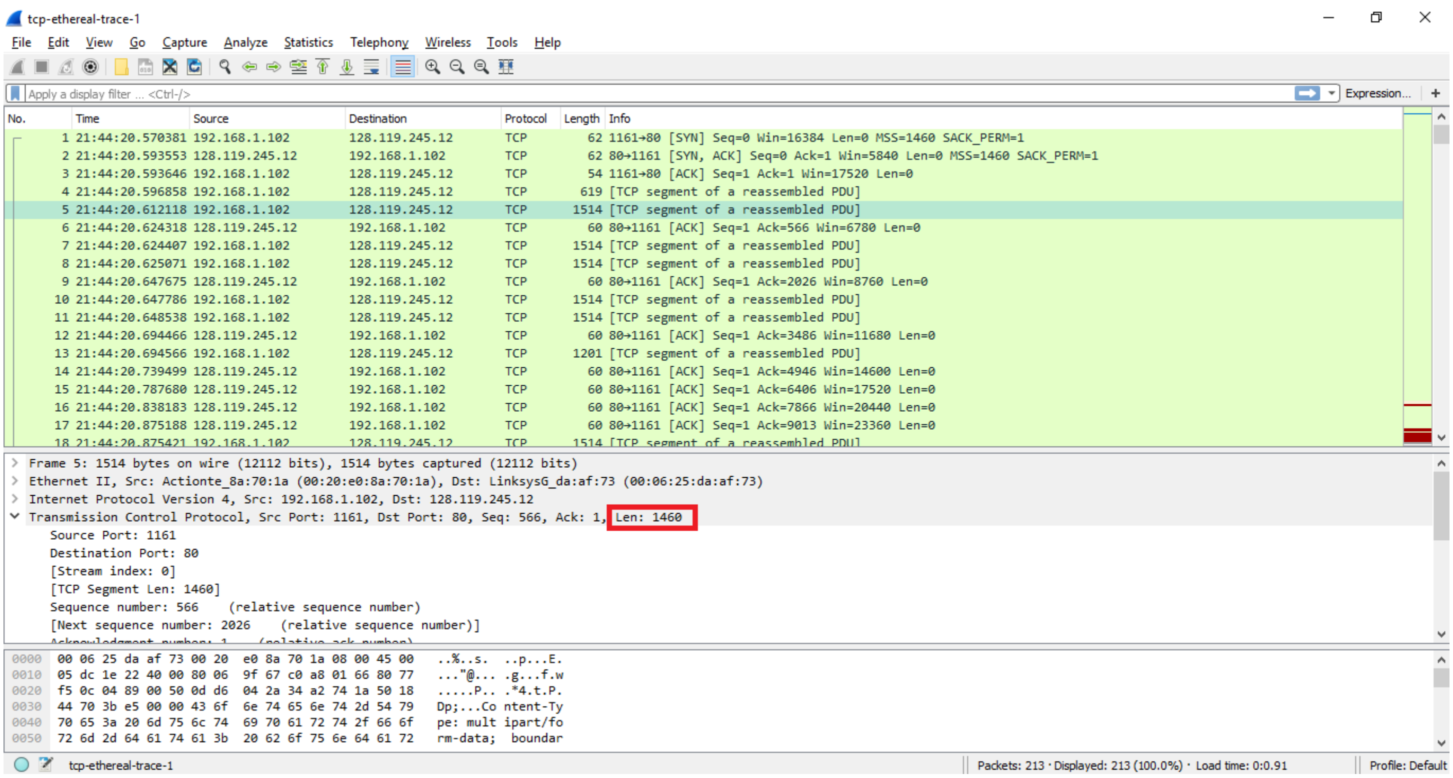




8. What is the length of each of the first six TCP segments?[1]

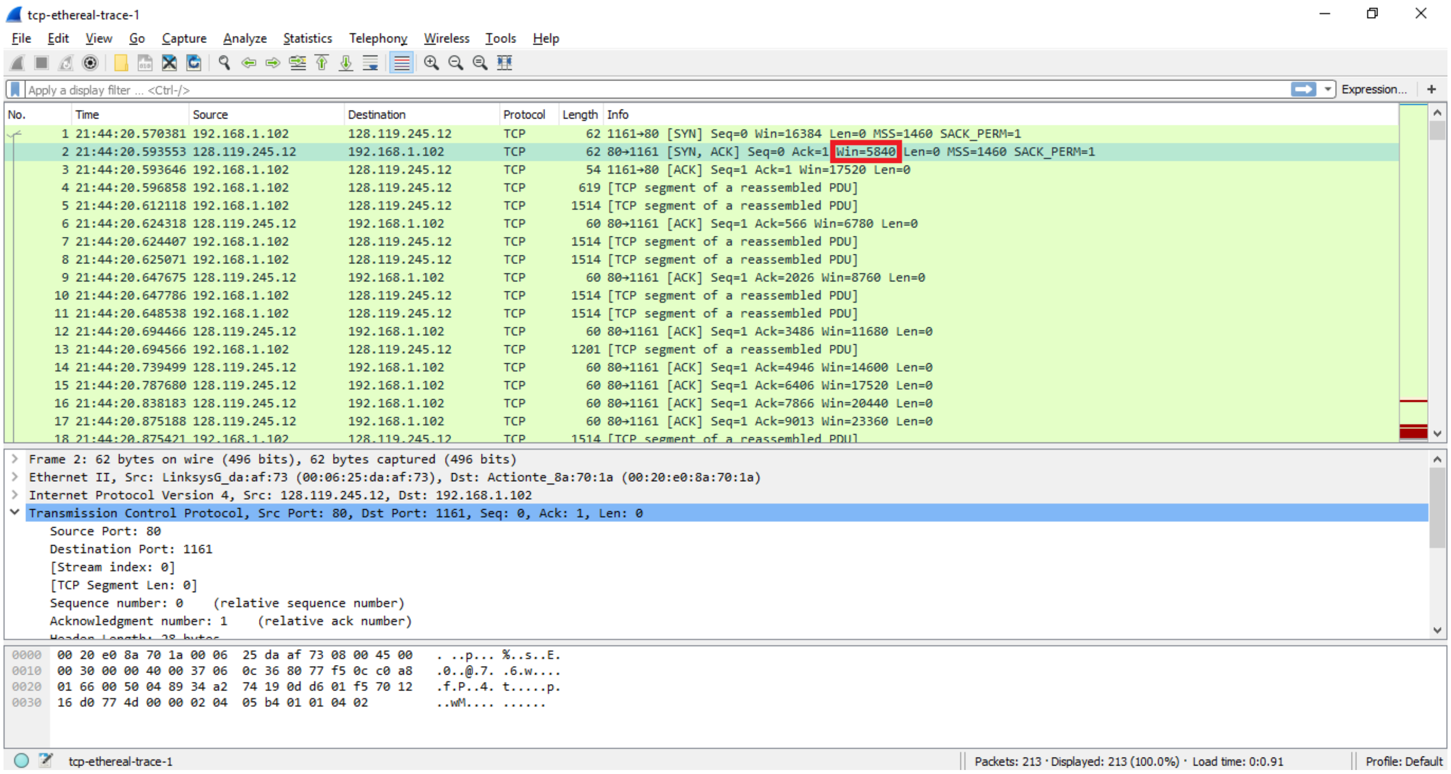
Length of 1st segment: 565 bytes
Length of 2nd, 3rd, 4th, 5th, 6th segments: 1460 bytes





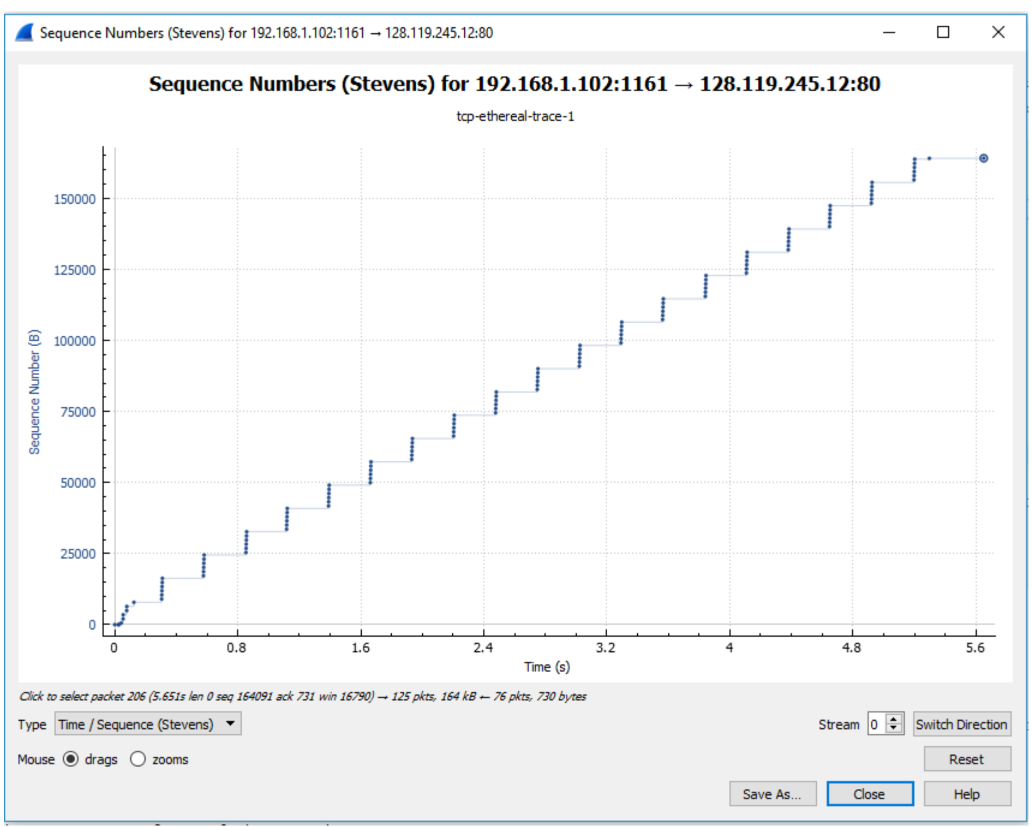
9. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

The minimum amount of available buffer space advertised at the receiver for the entire trace is 5840 bytes. No, the sender is never throttled because we never reach full capacity of the window.



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

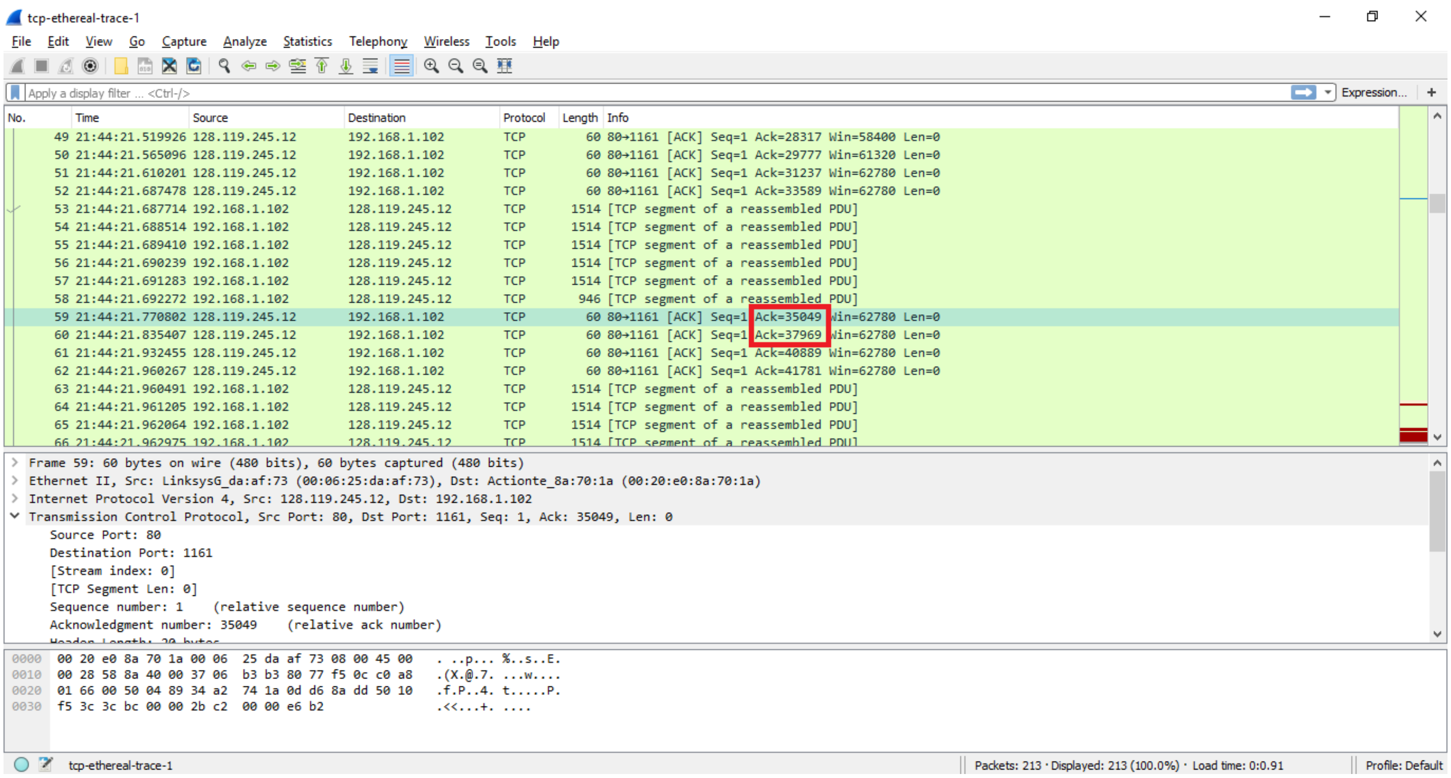
No, there are no retransmitted segments in the trace file. To check, I looked at the Time-Sequence-Graph (Stevens) of this trace and ensured that the sequence numbers and time had a linear relationship. In other words, as time increased, so too did the sequence numbers.



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).**

	ACK number	Acknowledged data
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 receiver typically acknowledges 1460 bytes per ACK. There are some cases where the receiver is ACKing every other received segment as can be seen in segment 60 where the amount of data acknowledged 2920 (1460*2).

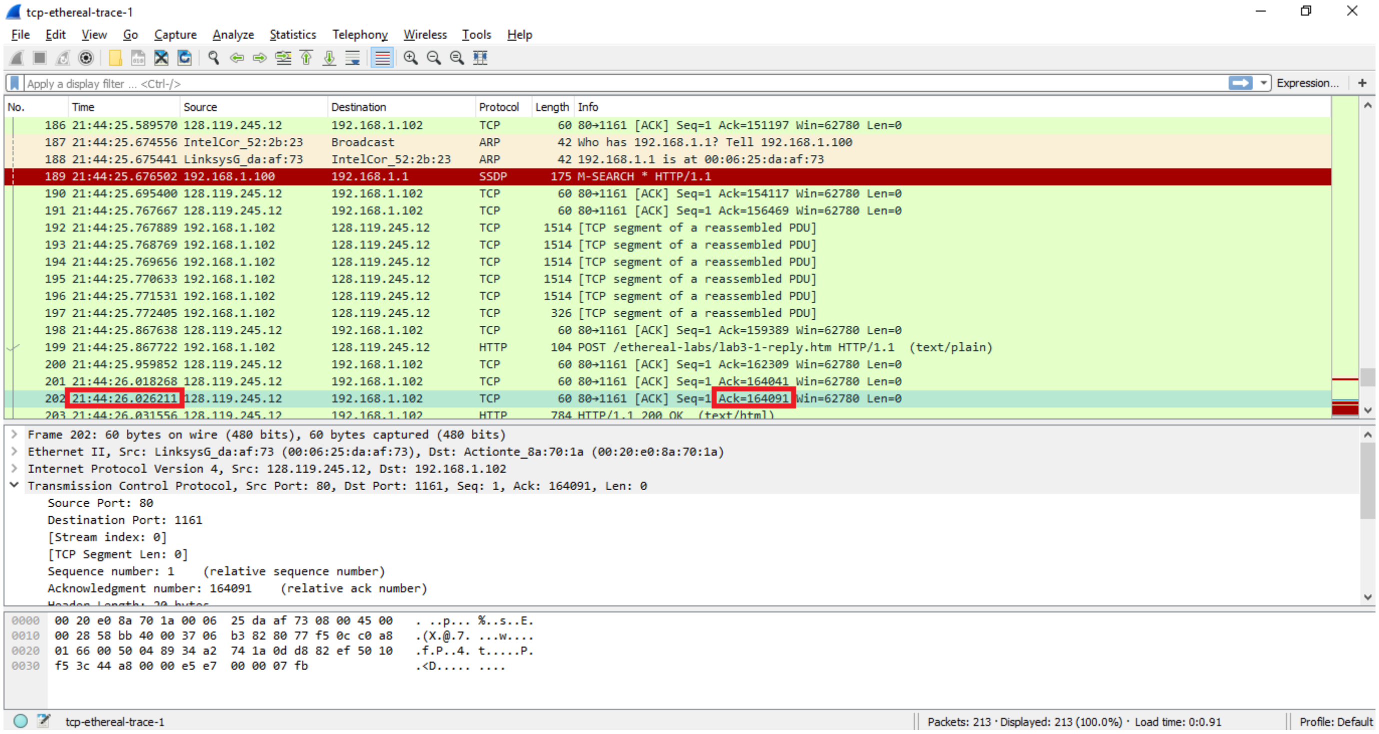


12. **What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.**

Total amount of data transmitted = 164091 (the last ACK received) – 1 (sequence number of first segment) = 164090 bytes

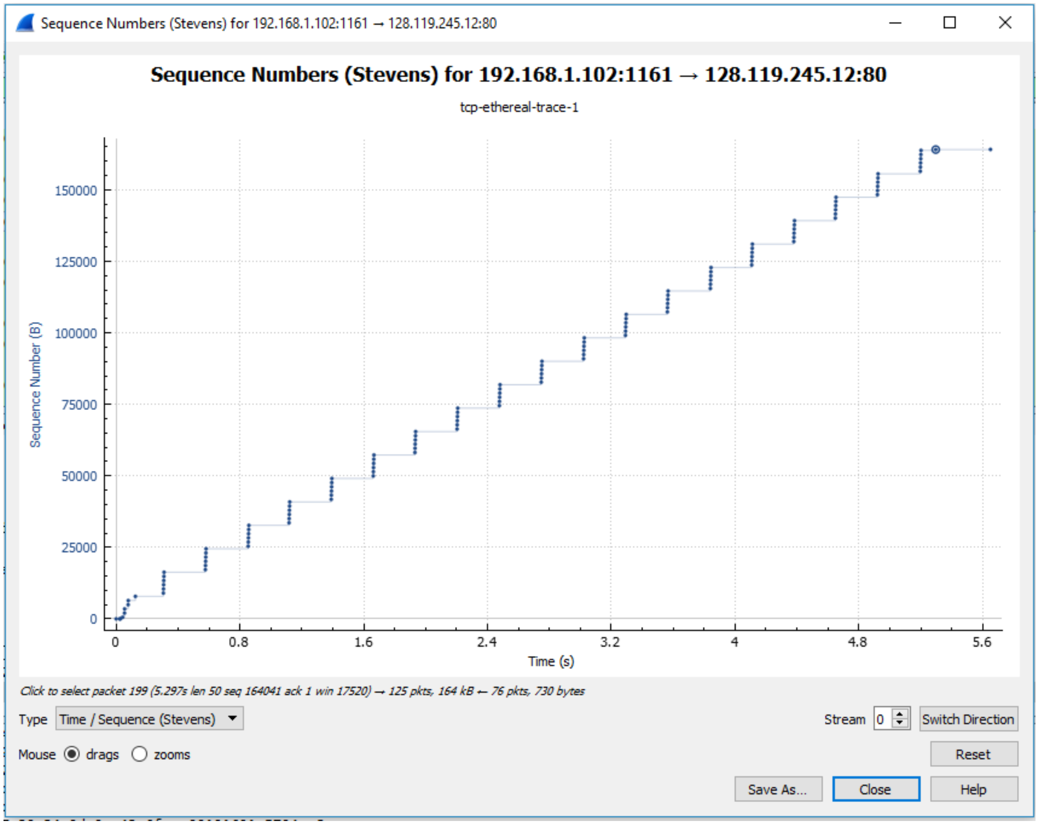
Total transmission time = 26.026211 (time last ACK received) – 20.596858 (time first segment sent) = 5.429353 seconds

Throughput = 164090 / 5.429353 = 30,223 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.**

In order identify the slowstart phase/congestion avoidance phases, we must look at the client's congestion window. However, the size of this window is not advertised. By looking at the amount of data sent out but not yet ACK'd, we can estimate the lower bound of the window. As can be seen in the table below, TCP's slowstart phase begins at the start of the connection and the un-ACK'd data increases rapidly but never grows above 8192B. So, it's not possible to identify the end of the slowstart phase because the client is not sending data fast enough to push the server into congestion avoidance. This behavior is different from the idealized behavior of TCP that we've studied where we normally assume that the clients have an abundance of data that they need to transmit. But in this case, the transmission is already complete before the network ever detects any congestion.

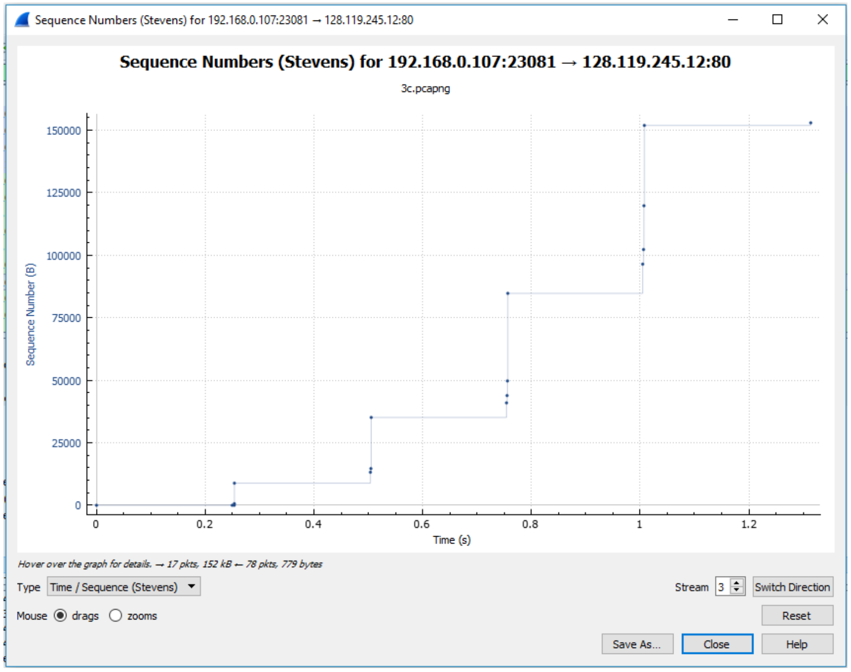


	SEQ number	Size	ACK number	Un-ACK'd data
DATA	1	565		565
DATA	566	1460		2025
ACK			566	1460
DATA	2026	1460		2920
DATA	3486	1460		4380
ACK			2026	2920
DATA	4946	1460		4380
DATA	6406	1460		5840
ACK			3486	4380
DATA	7866	1147		5527
ACK			4946	4067
ACK			6406	2607
ACK			7866	1147
ACK			9013	0
DATA	9013	1460		1460
DATA	10473	1460		2920
DATA	11933	1460		4380
DATA	13393	1460		5840
DATA	14853	1460		7300
DATA	16313	892		8192
ACK			10473	6732
ACK			11933	5272
ACK			13393	3812
ACK			14853	2352
ACK			16313	892
ACK			17205	0
DATA	17205	1460		1460
DATA	18665	1460		2920
DATA	20125	1460		4380
DATA	21585	1460		5840
DATA	23045	1460		7300

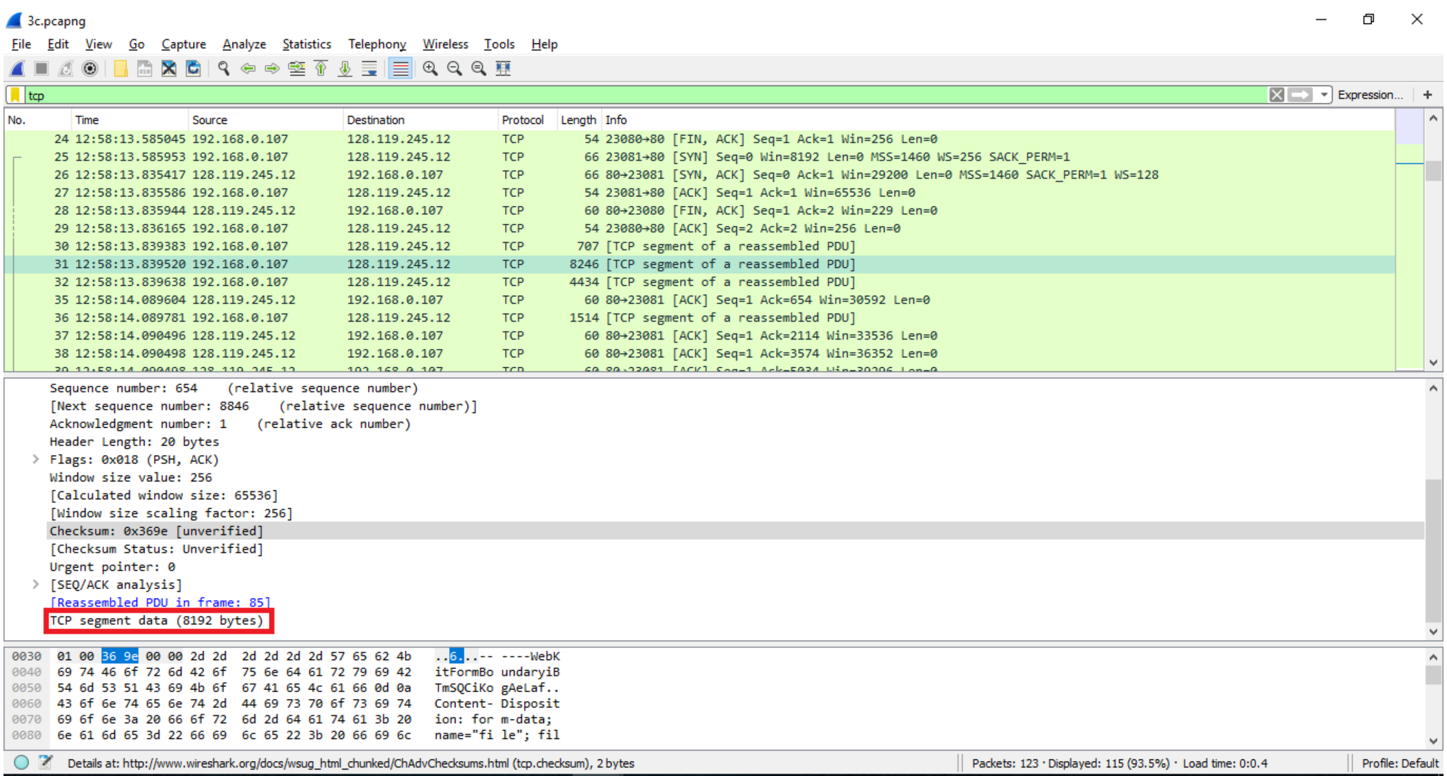
DATA	24505	892		8192
ACK			18665	6732
ACK			20125	5272
ACK			21585	3812
ACK			23045	2352
ACK			24505	892
ACK			25397	0
DATA	25397	1460		1460
DATA	26857	1460		2920
DATA	28317	1460		4380
DATA	29777	1460		5840
DATA	31237	1460		7300
DATA	32697	892		8192
ACK			26857	6732
ACK			28317	5272
ACK			29777	3812
ACK			31237	2352
ACK			33589	0
DATA	33589	1460		1460
DATA	35049	1460		2920
DATA	36509	1460		4380
DATA	37969	1460		5840
DATA	39429	1460		7300
DATA	40889	892		8192
ACK			35049	6732
ACK			37969	3812
ACK			40889	892
ACK			41781	0
DATA	41781	1460		1460
DATA	43241	1460		2920
DATA	44701	1460		4380
DATA	46161	1460		5840
DATA	47621	1460		7300
DATA	49081	892		8192
ACK			44701	5272
ACK			47621	2352
ACK			49973	0
DATA	49973	1460		1460
DATA	51433	1460		2920
DATA	52893	1460		4380
DATA	54353	1460		5840
DATA	55813	1460		7300
DATA	57273	892		8192
ACK			52893	5272
ACK			55813	2352
ACK			58165	0

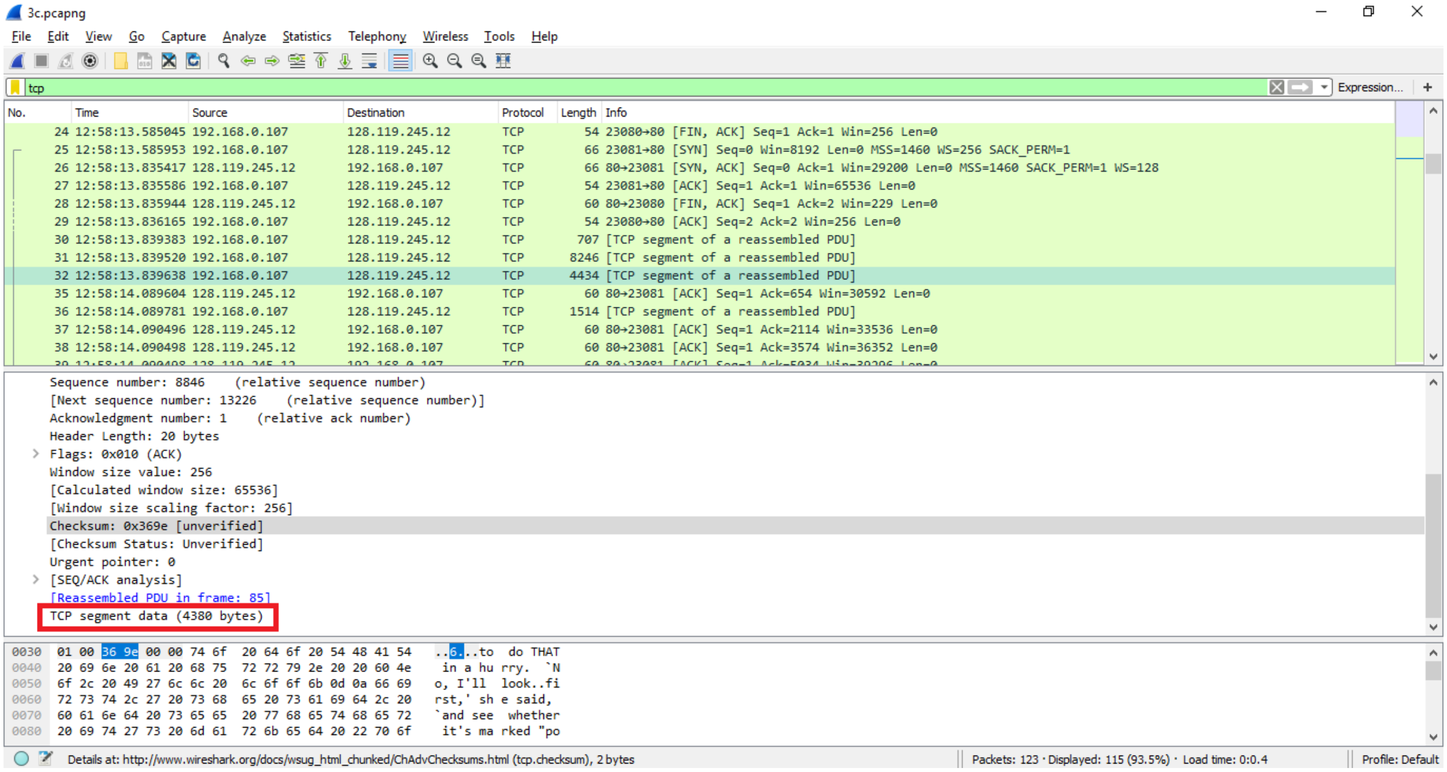
14. Answer Question 13 for the trace that you captured when you transferred a file from your *own* computer to gaia.cs.umass.edu

The behavior I observed when transferring a file from my own computer to gaia.cs.umass.edu was quite similar to that of the TCP trace provided. However, as previously noted, my trace indicates a TCP length greater than 1500 bytes, and Wireshark was reporting the wrong TCP segment length. Because of this inconsistency, calculating the un-ACK'd data was inefficient.



** As noted in footnote 3 of the lab instructions, I used the tcp-ethereal-trace-1 trace file for all of the calculations and analysis required as my trace indicated a TCP length greater than 1500 bytes. Wireshark was reporting the wrong TCP segment length and showed only one large TCP segment rather than multiple smaller segments. I have provided some screen shots below of my trace.




3c.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp
Expression...
No. Time Source Destination Protocol Length Info
24 12:58:13.585045 192.168.0.107 128.119.245.12 TCP 54 23080→80 [FIN, ACK] Seq=1 Ack=1 Win=256 Len=0
25 12:58:13.585953 192.168.0.107 128.119.245.12 TCP 66 23081→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
26 12:58:13.835417 128.119.245.12 192.168.0.107 TCP 66 80→23081 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
27 12:58:13.835586 192.168.0.107 128.119.245.12 TCP 54 23081→80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
28 12:58:13.835944 128.119.245.12 192.168.0.107 TCP 60 80→23080 [FIN, ACK] Seq=1 Ack=2 Win=229 Len=0
29 12:58:13.836165 192.168.0.107 128.119.245.12 TCP 54 23080→80 [ACK] Seq=2 Ack=2 Win=256 Len=0
30 12:58:13.839383 192.168.0.107 128.119.245.12 TCP 707 [TCP segment of a reassembled PDU]
31 12:58:13.839520 192.168.0.107 128.119.245.12 TCP 8246 [TCP segment of a reassembled PDU]
32 12:58:13.839638 192.168.0.107 128.119.245.12 TCP 4434 [TCP segment of a reassembled PDU]
35 12:58:14.089604 128.119.245.12 192.168.0.107 TCP 60 80→23081 [ACK] Seq=1 Ack=654 Win=30592 Len=0
36 12:58:14.089781 192.168.0.107 128.119.245.12 TCP 1514 [TCP segment of a reassembled PDU]
37 12:58:14.090496 128.119.245.12 192.168.0.107 TCP 60 80→23081 [ACK] Seq=1 Ack=2114 Win=33536 Len=0
38 12:58:14.090498 128.119.245.12 192.168.0.107 TCP 60 80→23081 [ACK] Seq=1 Ack=3574 Win=36352 Len=0
Sequence number: 8846 (relative sequence number)
[Next sequence number: 13226 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Header Length: 20 bytes
Flags: 0x010 (ACK)
Window size value: 256
[Calculated window size: 65536]
[Window size scaling factor: 256]
Checksum: 0x369e [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
[SEQ/ACK analysis]
[Reassembled PDU in frame: 85]
TCP segment data (4380 bytes)
0030 01 00 36 9e 00 00 74 6f 20 64 6f 20 54 48 41 54 ..6...to do THAT
0040 20 69 6e 20 61 20 68 75 72 72 79 2e 20 20 60 4e in a hu rry. `N
0050 6f 2c 20 49 27 6c 6c 20 6c 6f 6f 6b 0d 0a 66 69 o, I'll look..fi
0060 72 73 74 2c 27 20 73 68 65 20 73 61 69 64 2c 20 rst,' sh e said,
0070 60 61 6e 64 20 73 65 65 20 77 68 65 74 68 65 72 `and see whether
0080 20 69 74 27 73 20 6d 61 72 6b 65 64 20 22 70 6f it's ma rked "po
Details at: http://www.wireshark.org/docs/wsug_html_chunked/ChAdvChecksums.html (tcp.checksum), 2 bytes
Packets: 123 · Displayed: 115 (93.5%) · Load time: 0:0.4
Profile: Default