LAB REPORT 3

Subject: TCP/UDP PROTOCOL

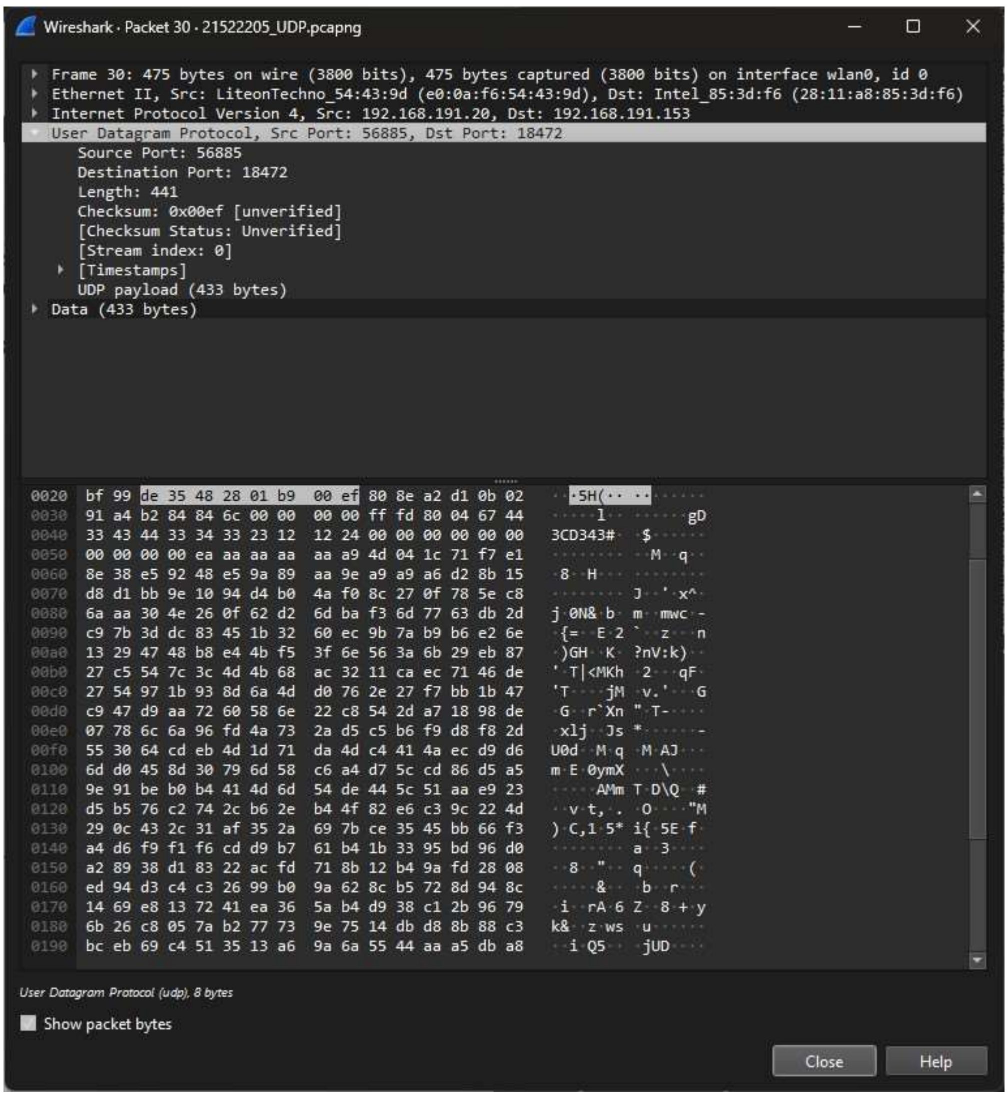
Student information	**Student 1** ID: 22520836 Name: Ngo Thi Hong Ly **Student 2** ID: 22521099 Name: Le Hoang Thien Phu **Student 3** ID: 22521237 Name: Pham Quang Dai Phuc **Student 4** ID: 22521240 Name: Le Minh Sang
Class	**CS4283.O21.CTTT.1**
Work division:	**[Student 1]**: Do question 9,10,11,12 task 3 and support the other **[Student 2]**: Do question 4,5,6,7,8 task 3 and support the other **[Student 3]**: Do the task 2, 4 and support the other **[Student 4]**: Do the task 1 and support the other
Video link of implementation *(if required)*	
Opinions (if any) + Difficulties encountered + Suggestions, comments...	

Task 1

1. **Select one UDP packet from your trace. From this packet, determine how many fields there are in the UDP header.**

→ From the selected UDP packet, observe the UDP header fields. Common fields in the UDP header include:

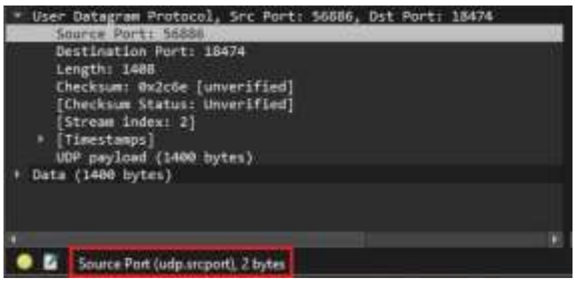
- Source Port: 56886
- Destination Port: 18474
- Length: 1408 bytes
- Checksum: 0x2c6e

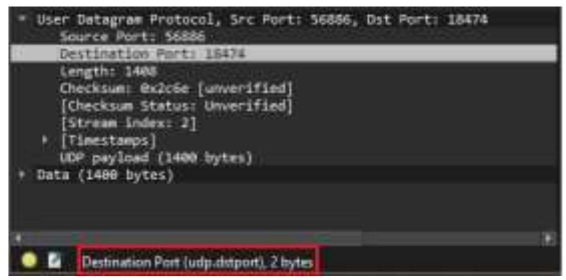


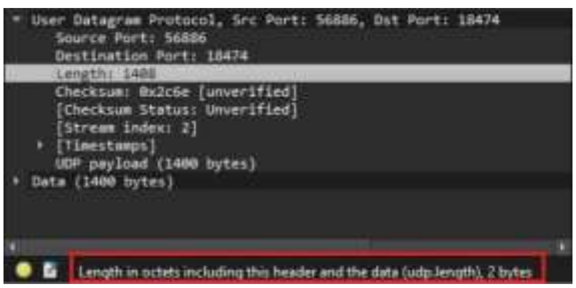
2. By consulting the displayed information in Wireshark's packet content field for this packet, determine the length (in bytes) of each of the UDP header fields.

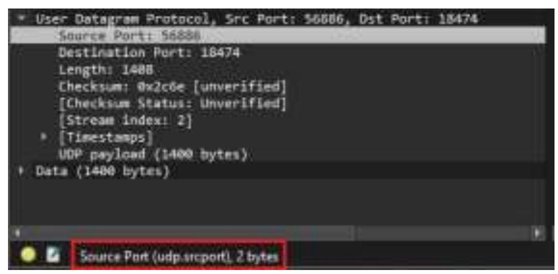
→ There are four UDP header fields:

- Source Port: 2 bytes
- Destination Port: 2 bytes
- Length: 2 bytes
- Checksum: 2 bytes









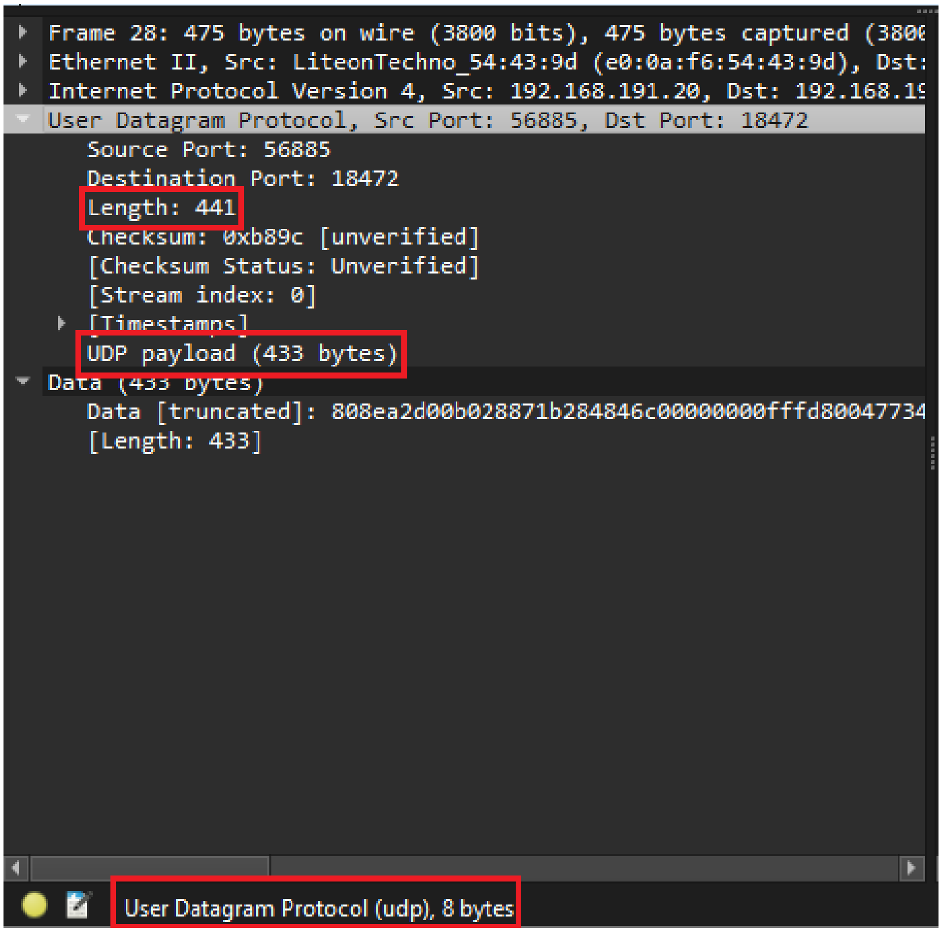
3. The value in the Length field is the length of what?

→ **Length Field in UDP Header**: is the length of the entire datagram

Length = UPD header + UDP payload

In UDP header field, each field is 2 bytes long. Therefore the total length of the UDP header is 8 bytes. Beside that, the UDP payload is 443 bytes long. Therefore, the Length = UDP header + UDP payload = 441

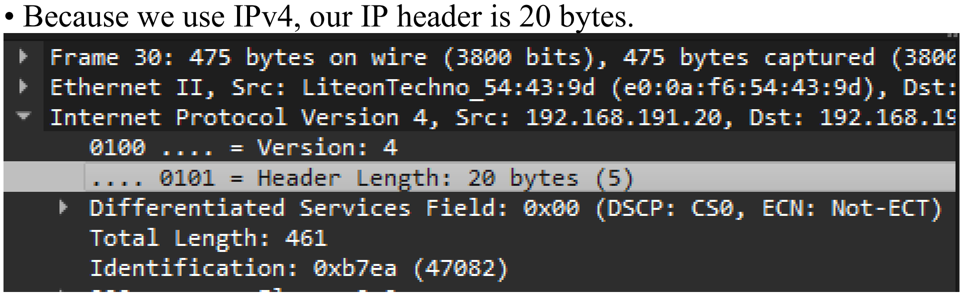
- UDP header: 8 bytes
- UDP payload: 443 bytes
- Length: 441



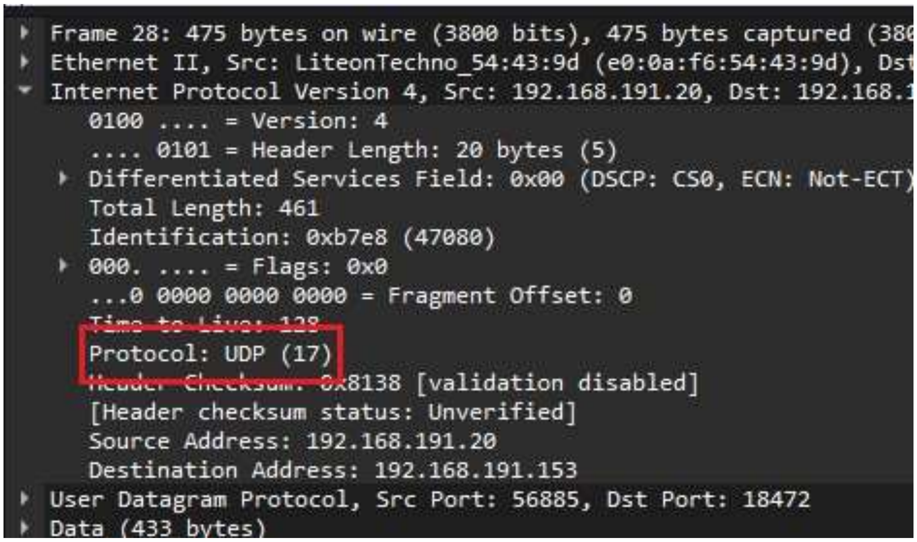
4. What is the maximum number of bytes that can be included in a UDP payload?

→ The maximum number of bytes that a UDP payload can contain (including UDP header and IP header) is 216-1 = 65535 (bytes). (Length)

- Which: Length = Data Length + UDP header Length + IP header Length.
- In there:
  - Our UDP header is fixed at 8 bytes (as described above)
  - Because we use IPv4, our IP header is 20 bytes.



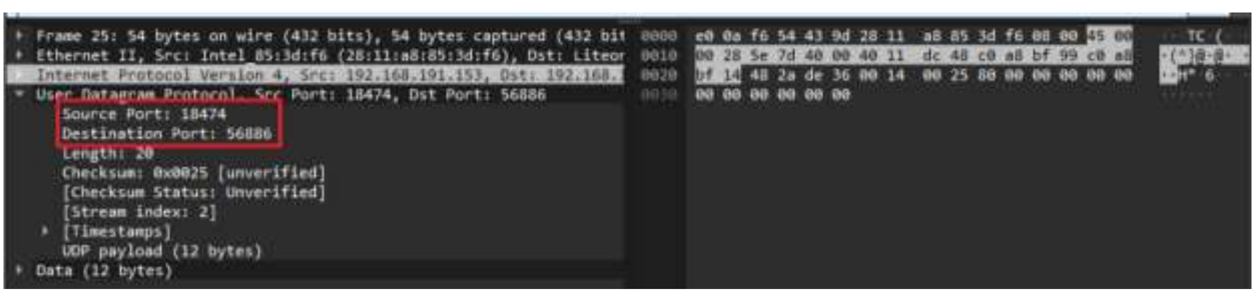
5. **What is the largest possible source port number?**
   → The largest possible source port number is determined by the size of the field allocated for port numbers in the UDP header. Typically, it's 16 bits, allowing for a maximum value of $2^{16} - 1 = 65535$.
6. **What is the protocol number for UDP?**
   →The protocol number for UDP (User Datagram Protocol) is 17 in decimal notation and 0x11 in hexadecimal notation.



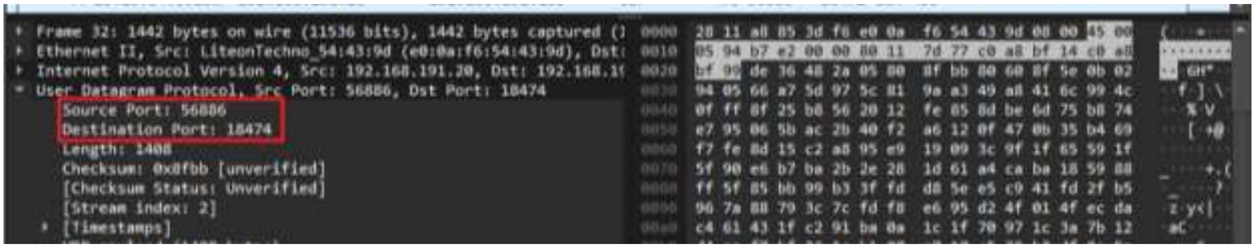
7. **Examine a pair of UDP packets in which your host sends the first UDP packet and the second UDP packet is a reply to this first UDP packet.**

→ Relationship between Port Numbers:

- **First UDP Packet (Frame 25)**:
  - Source Port: 18474
  - Destination Port: 56886



- **Second UDP Packet (Frame 32)** (Reply to the first packet):
  - Source Port: 56886
  - Destination Port: 18474


Frame 32: 1442 bytes on wire (11536 bits), 1442 bytes captured
Ethernet II, Src: LiteonTechno_54:43:9d (e0:0a:f6:54:43:9d), Dst:
Internet Protocol Version 4, Src: 192.168.191.20, Dst: 192.168.1
User Datagram Protocol, Src Port: 56886, Dst Port: 18474
Source Port: 56886
Destination Port: 18474
Length: 1408
Checksum: 0x8fbb [unverified]
[Checksum Status: Unverified]
[Stream index: 2]
[Timestamps]

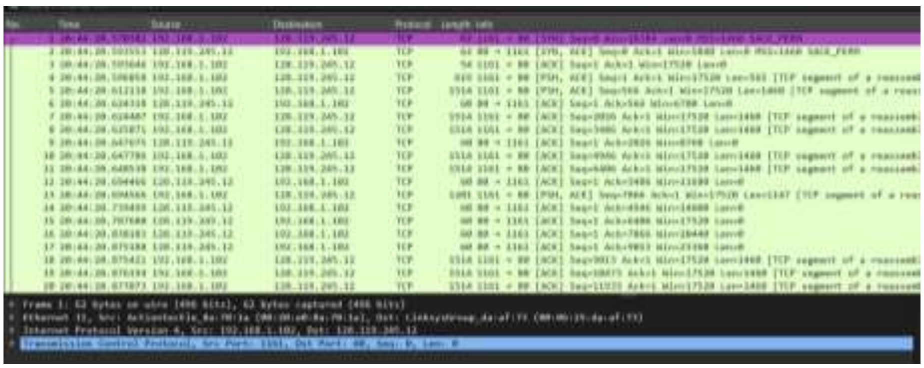
Task 2

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**

→ **Client computer**

IP address: 192.168.1.102

Port number: 1161

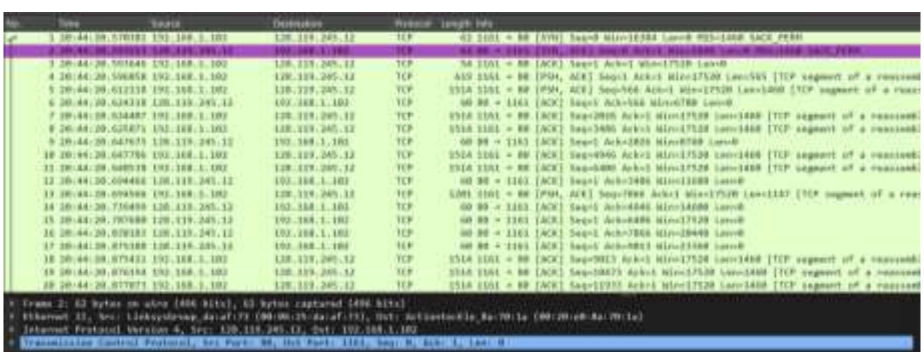


2. **What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?**

→ gaia.cs.umass.edu

IP address: 128.119.245.12

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?**

→ Client computer

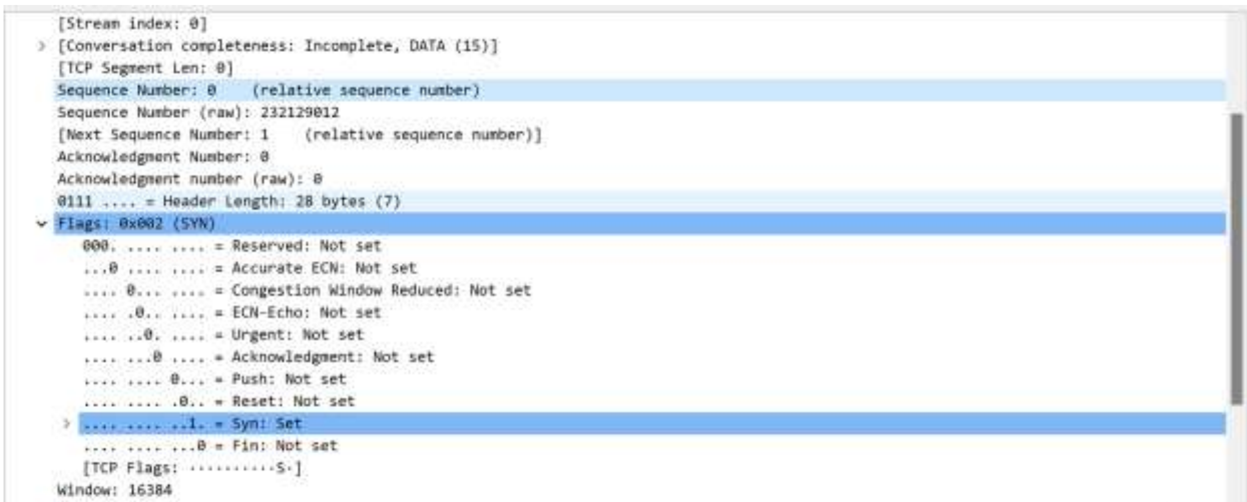
IP address: 192.168.191.20

Port number: 65022

Task 3

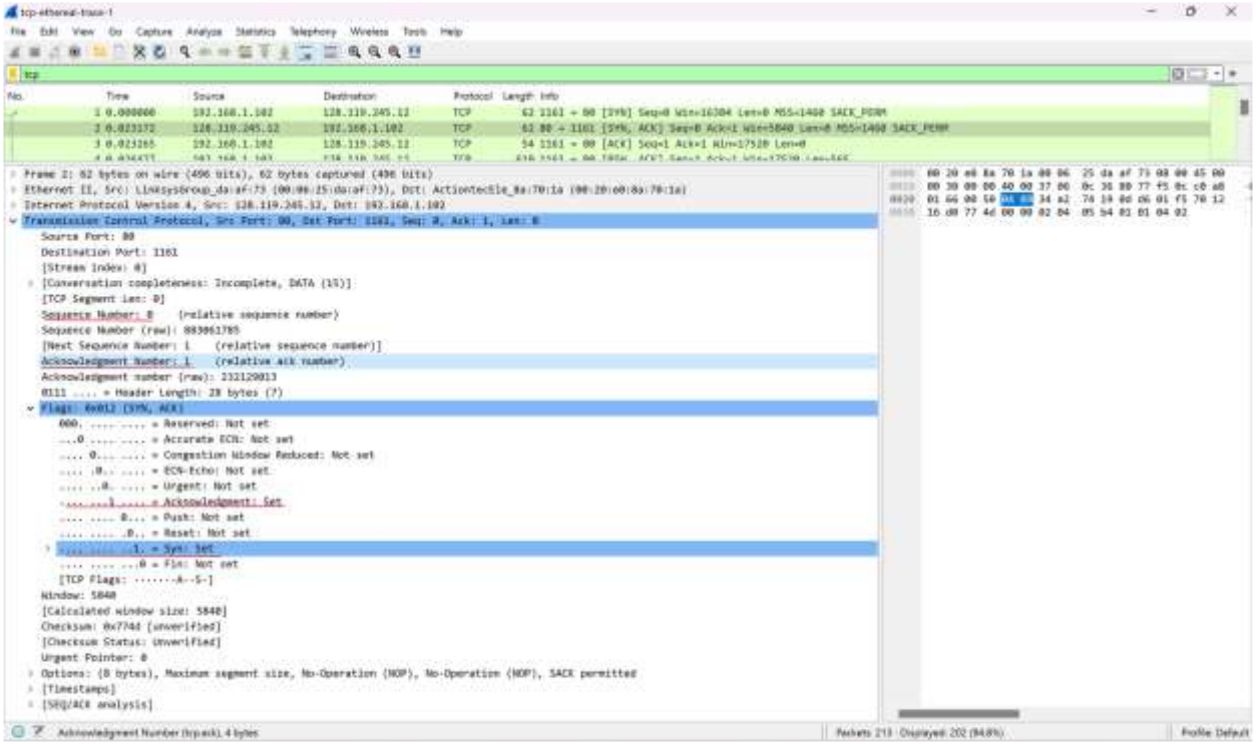
4. **What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?**

  - 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: 0
  - In a Flags field, the SYN flag is set = 1 that identifies the segment as a SYN segment.



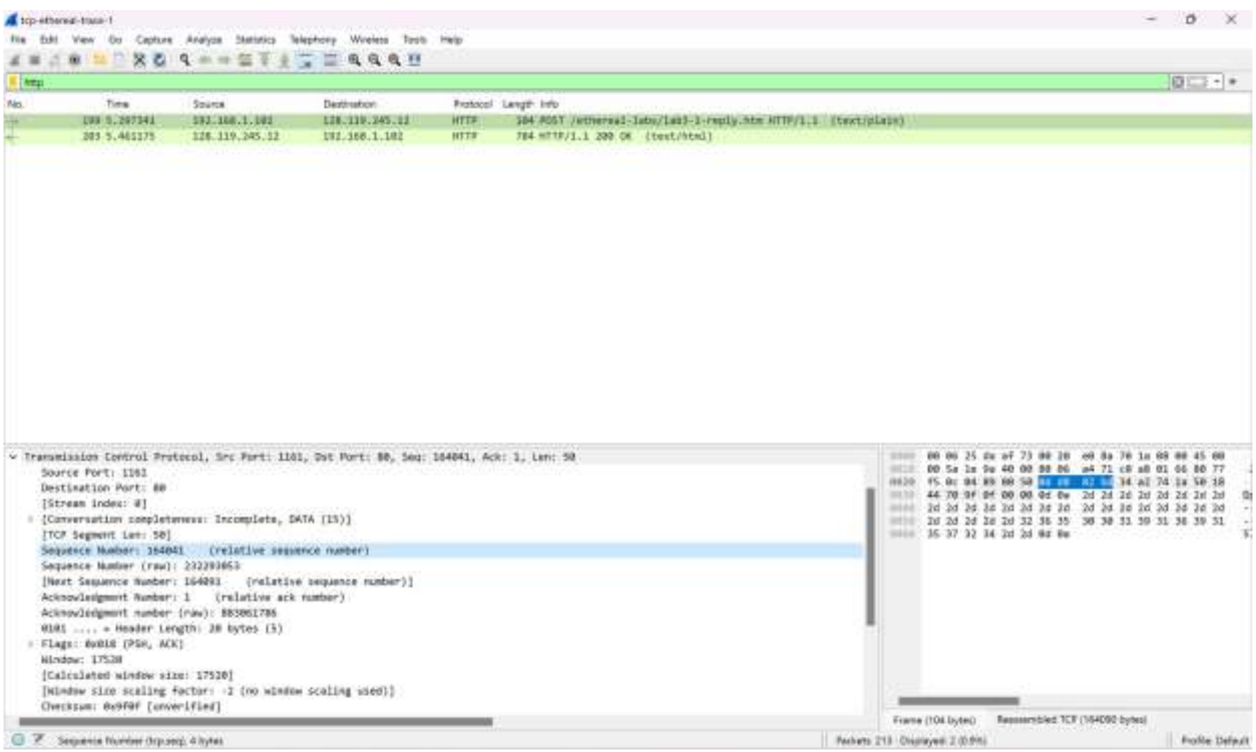
5. **What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a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

  - The sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN: 0
  - The value of the Acknowledgement field in the SYNACK segment: 1
  - In a Flags field, the (SYN, ACK) flag and the Acknowledgement flag is set = 1 that identifies the segment as a SYNACK segment.

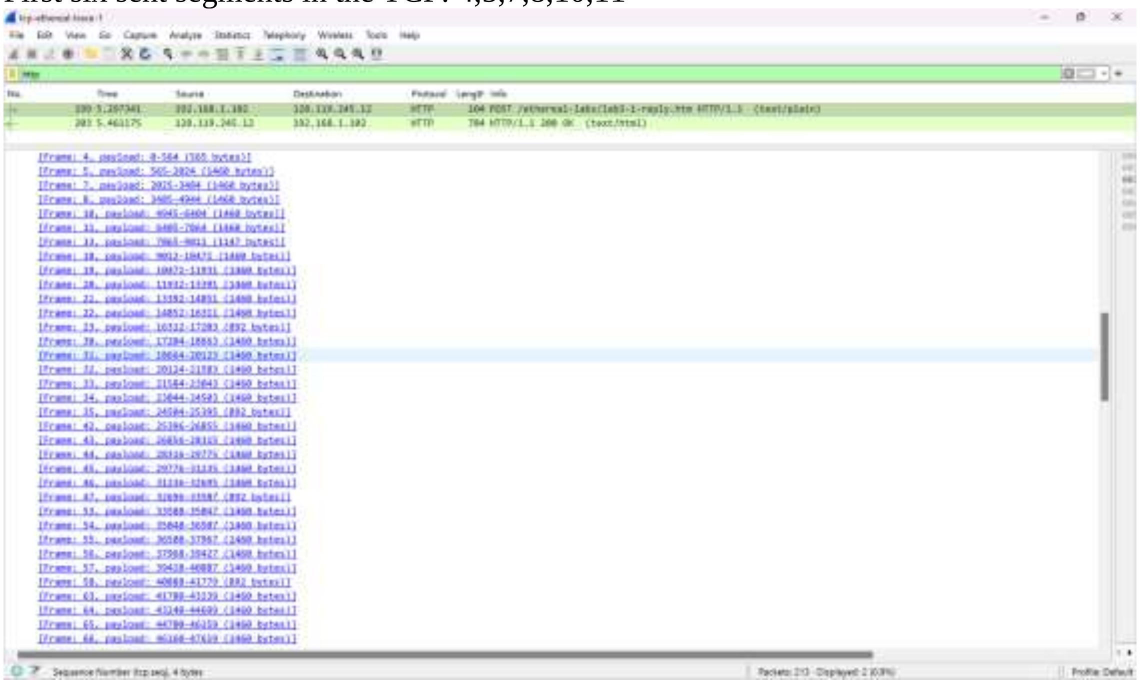


6. **What is the sequence number of the TCP segment containing the HTTP POST command?**

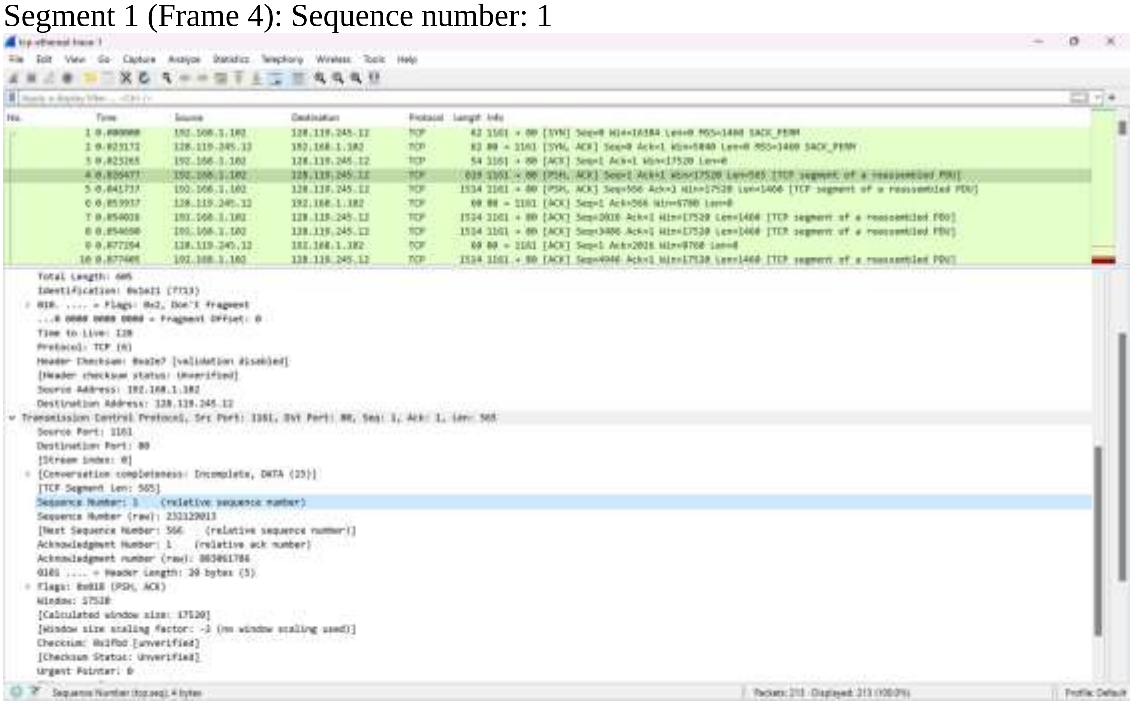
→ The sequence number of the TCP segment containing the HTTP POST command: 164041



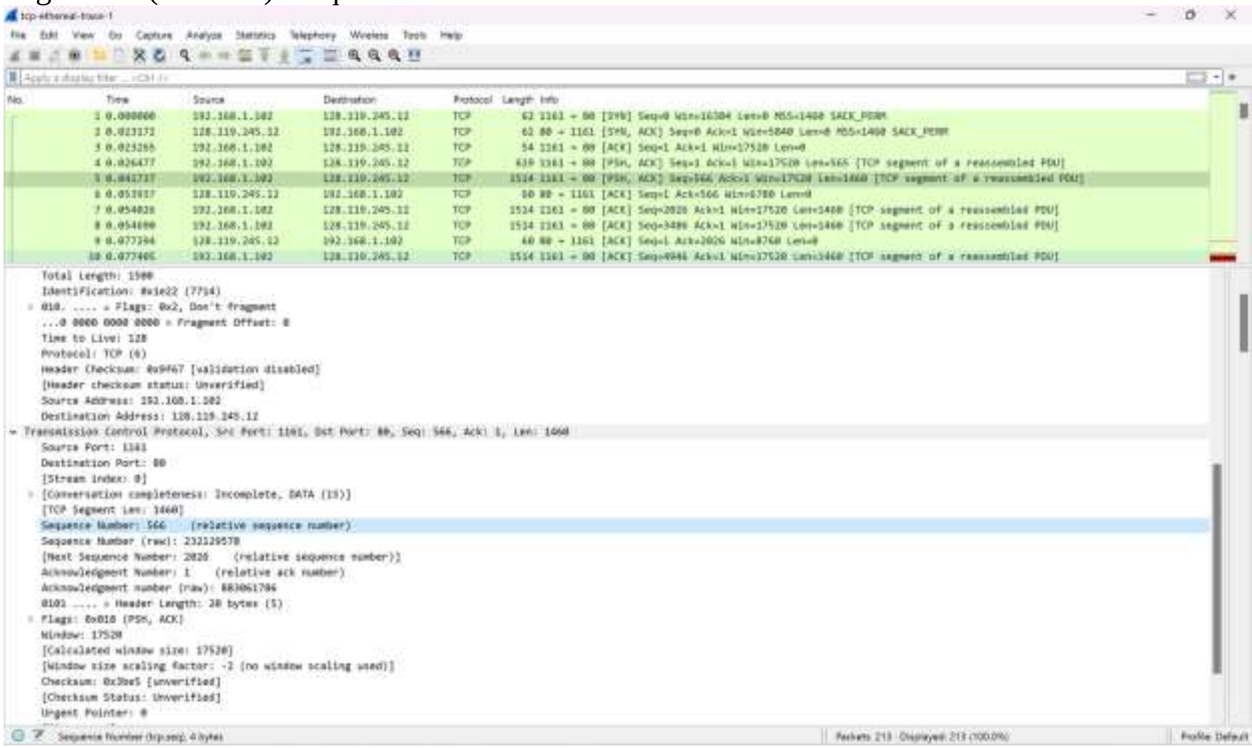
7. **Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? 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 after the receipt of each ACK?**
   - First six sent segments in the TCP: 4,5,7,8,10,11



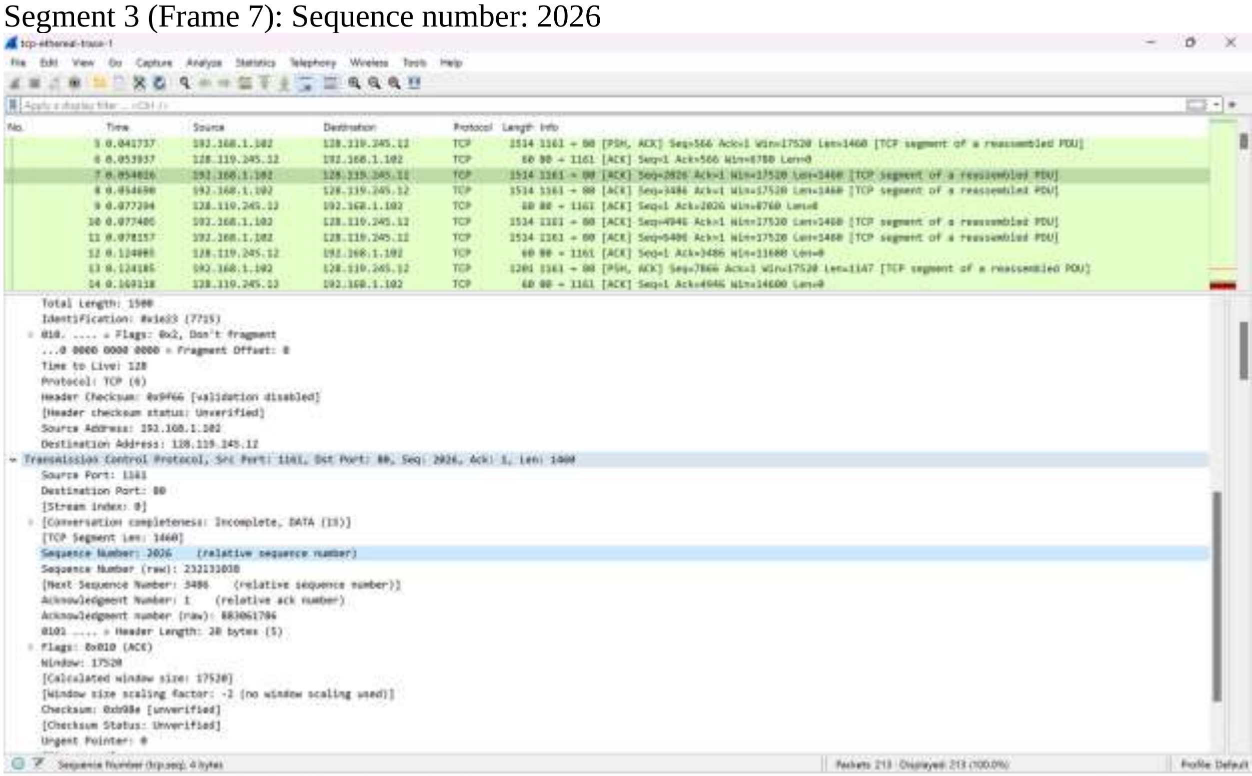
   - Segment 1 (Frame 4): Sequence number: 1



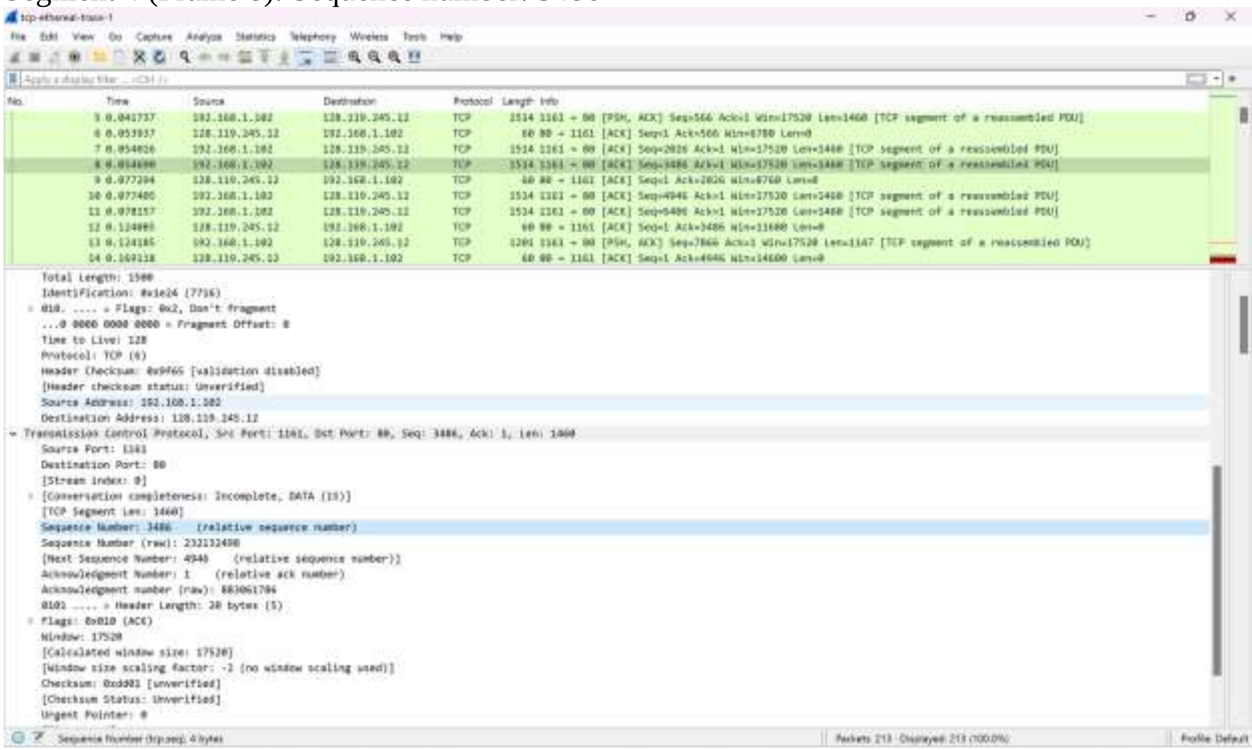
- Segment 2 (Frame 5): Sequence number: 566



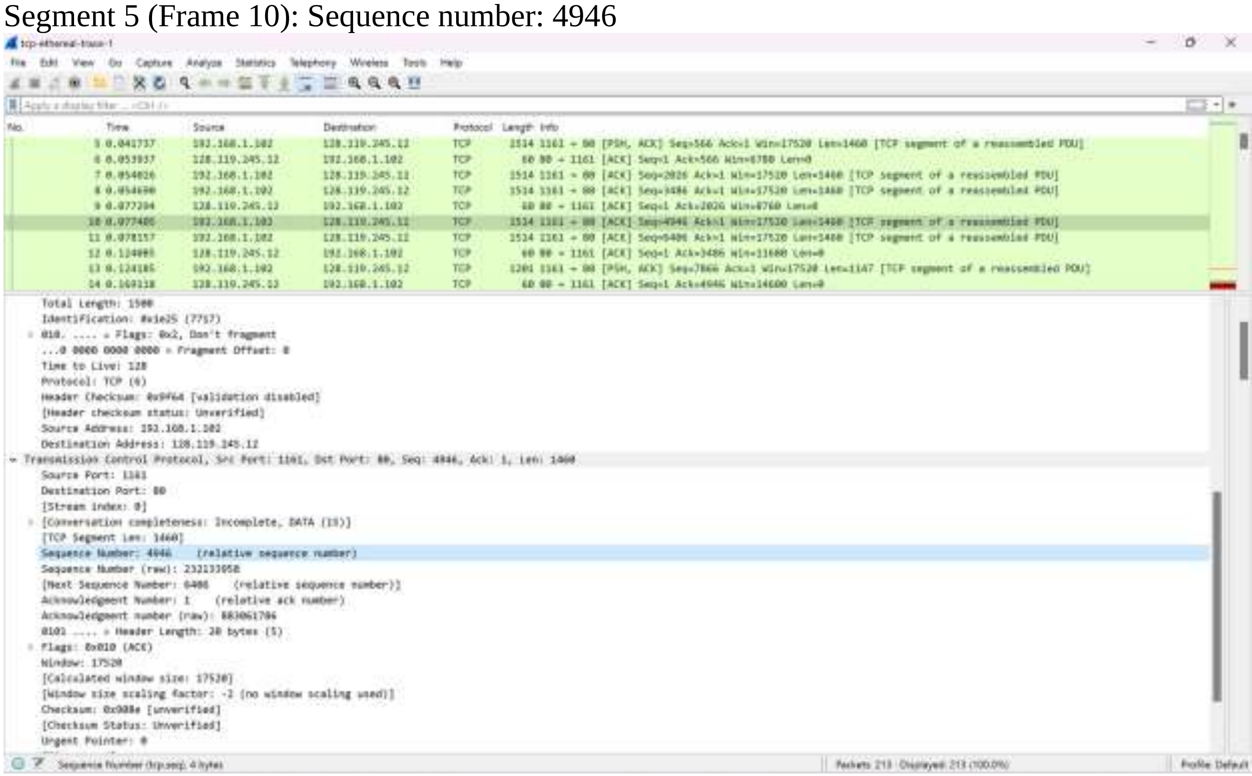
- Segment 3 (Frame 7): Sequence number: 2026



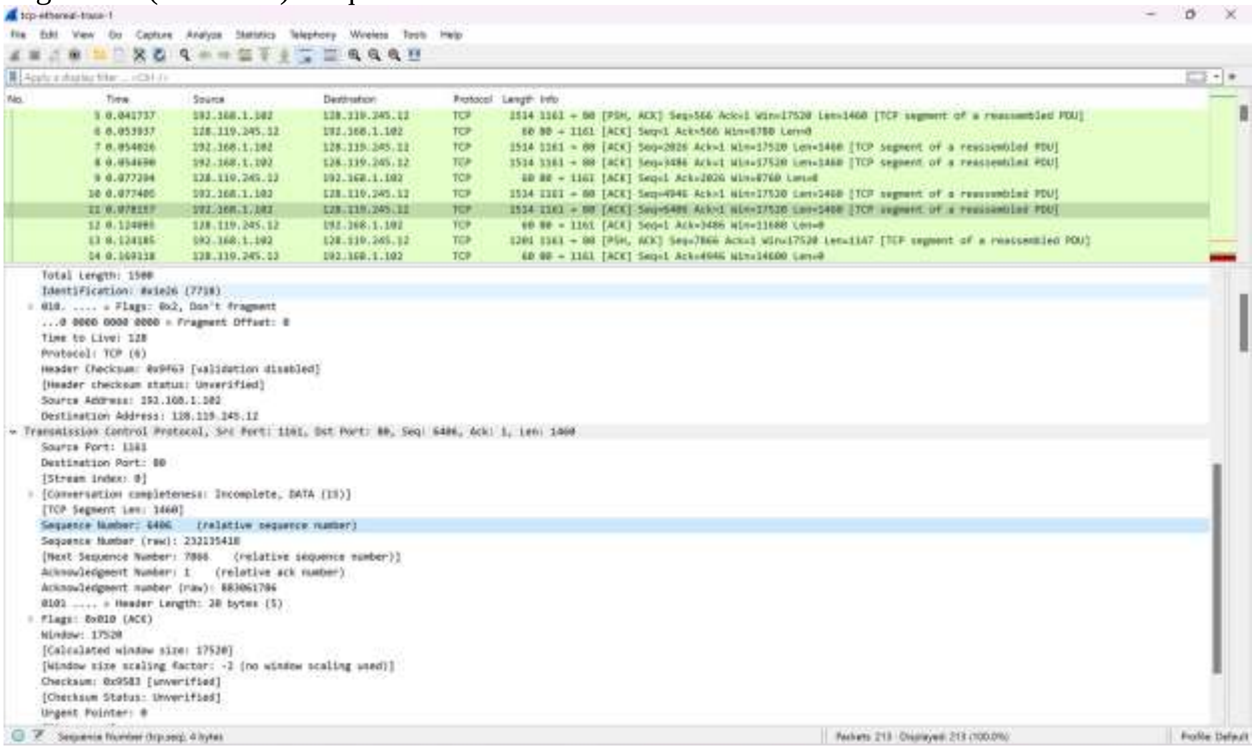
- Segment 4 (Frame 8): Sequence number: 3486



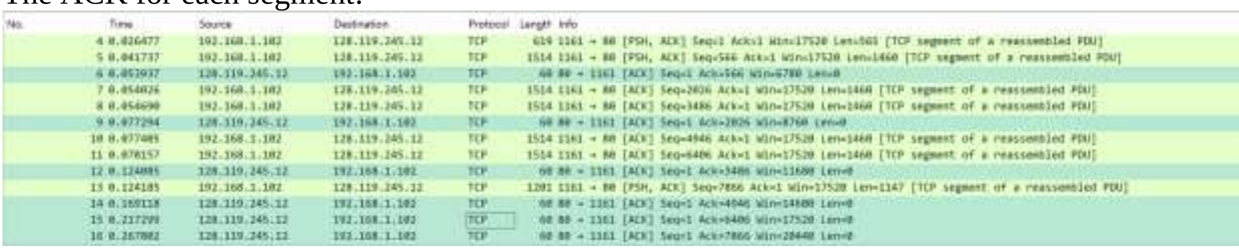
- Segment 5 (Frame 10): Sequence number: 4946



- Segment 6 (Frame 11): Sequence number: 6406



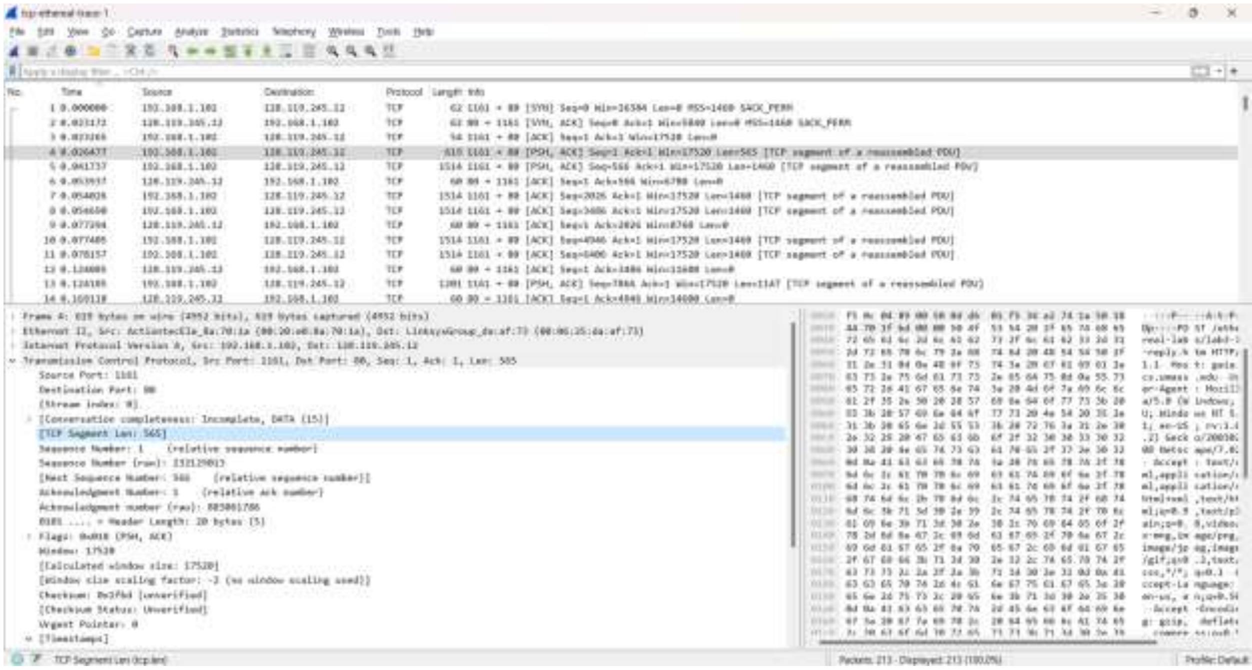
- The ACK for each segment:



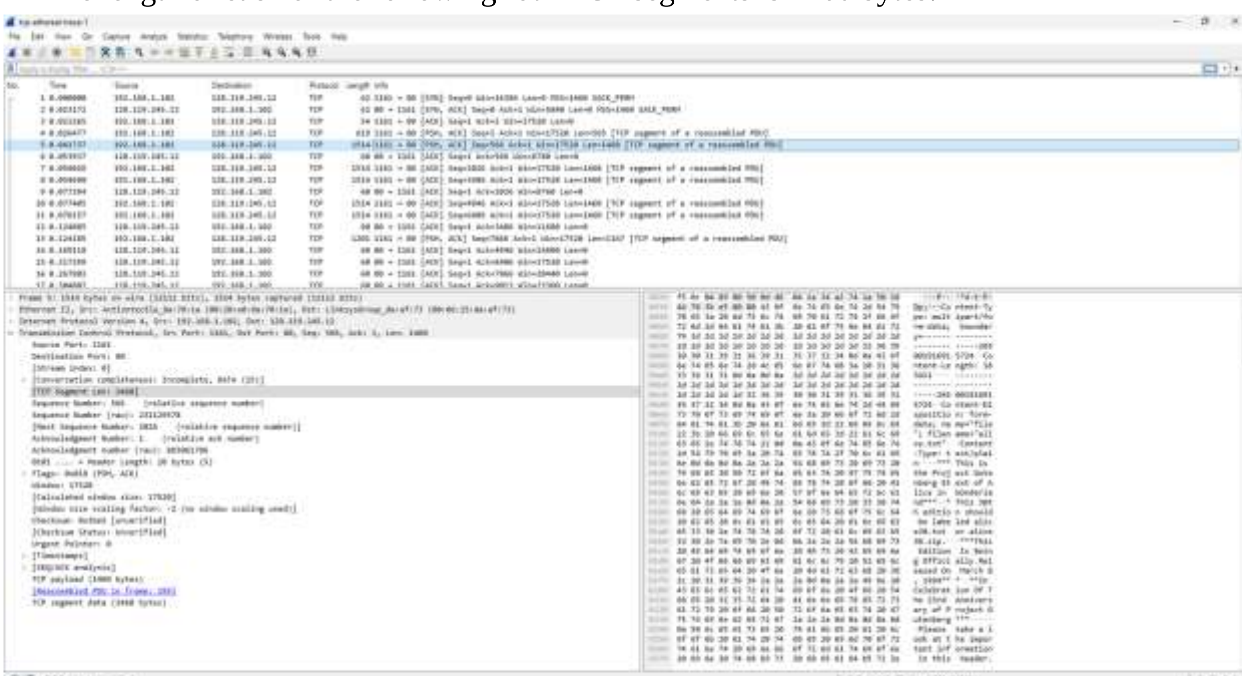
Segment	Sent time	ACK received time	RTT (seconds)
1	0.026477	0.053937	0.02746
2	0.041737	0.077294	0.035557
3	0.054026	0.124085	0.070059
4	0.054690	0.169118	0.114428
5	0.077405	0.217299	0.139894
6	0.078157	0.267802	0.189645

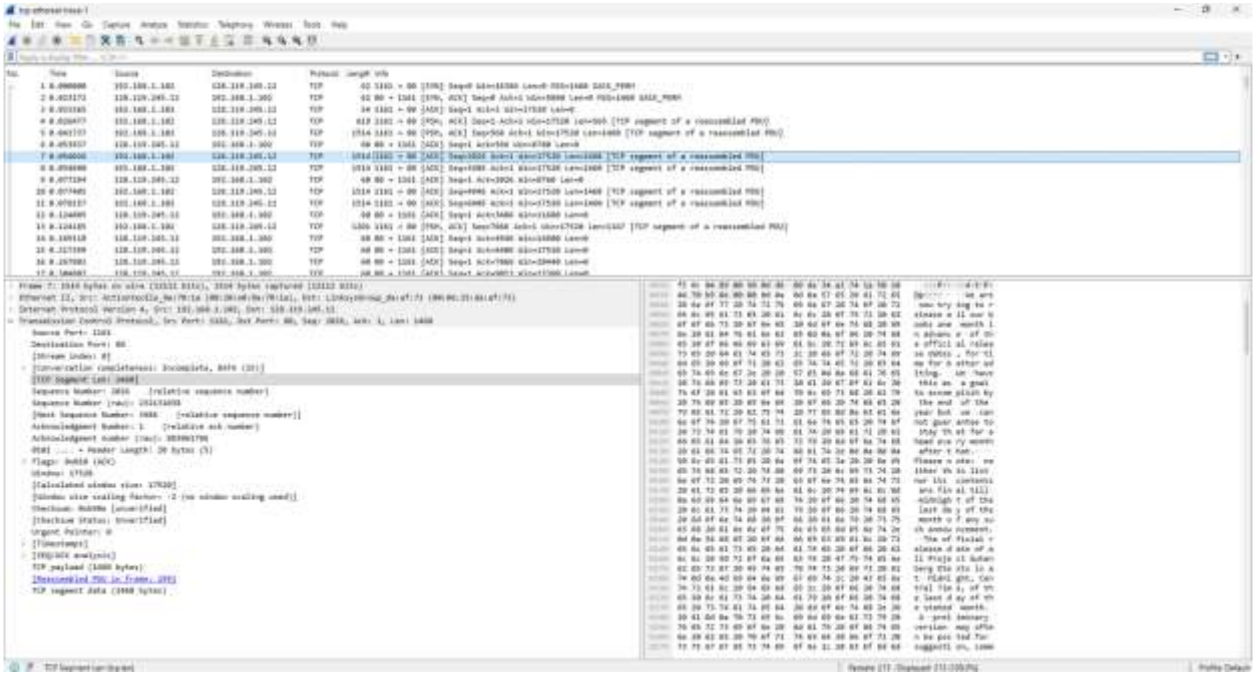
8. What is the length of each of the first six TCP segments?

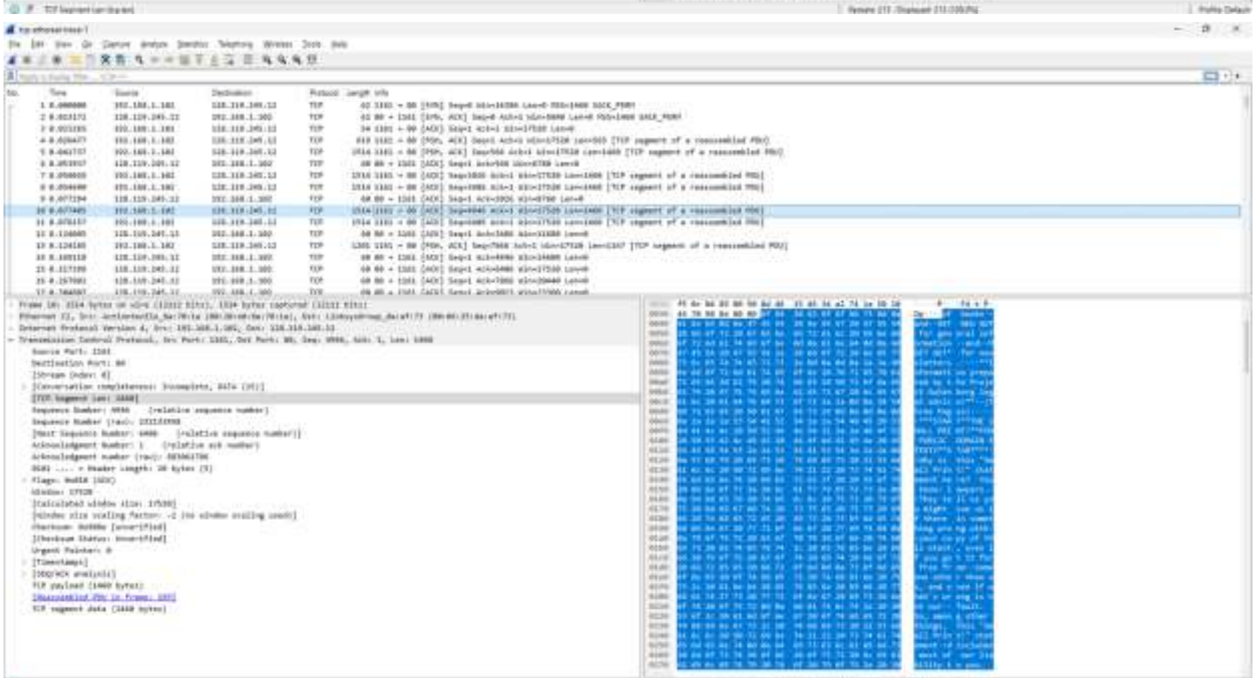
→ The length of the first TCP segment is 565 bytes:

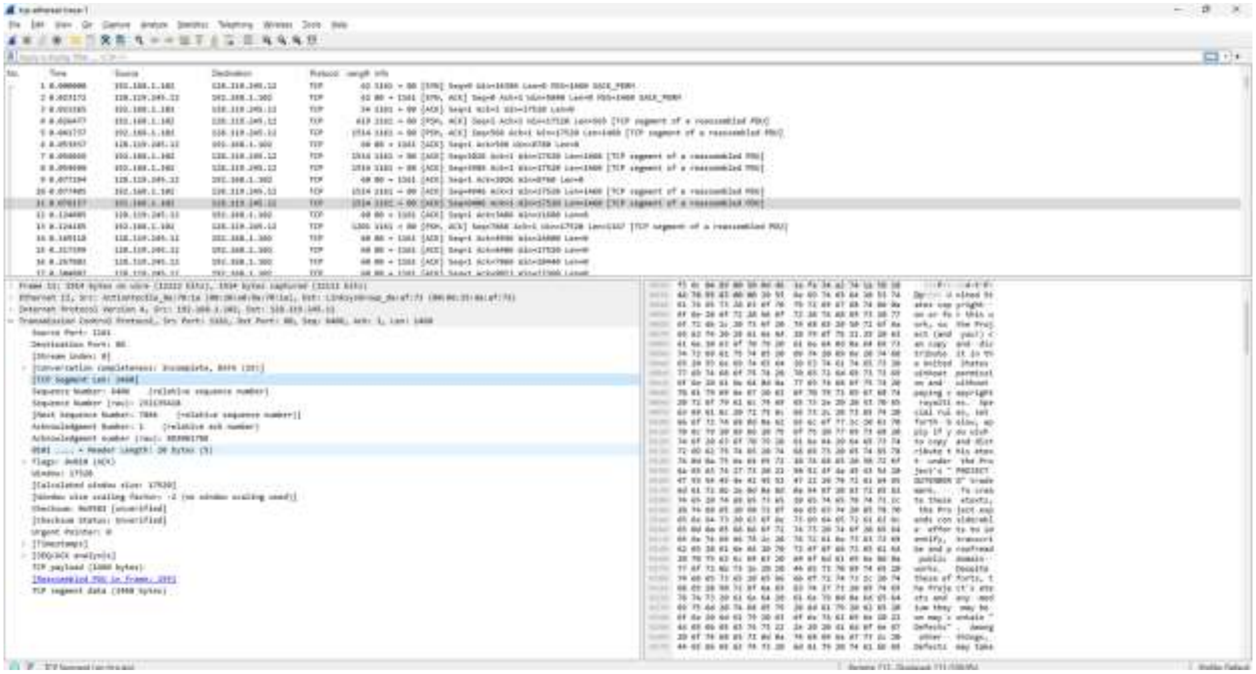


→ The length of each of the following four TCP segments is 1460 bytes:

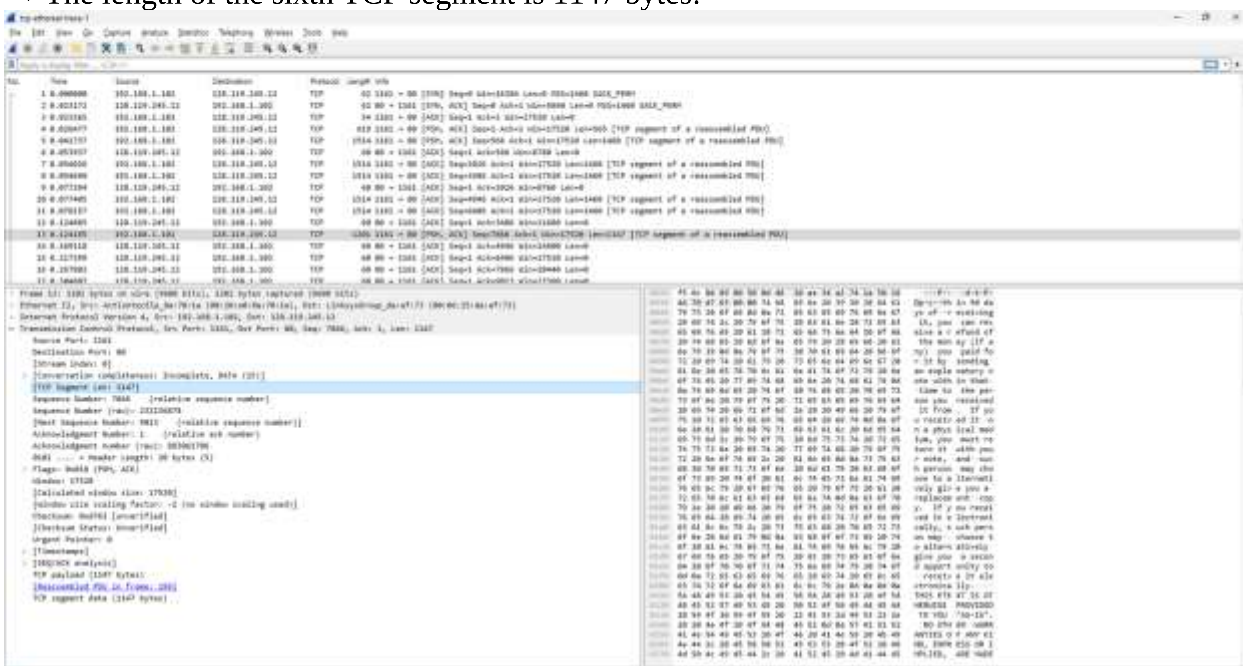






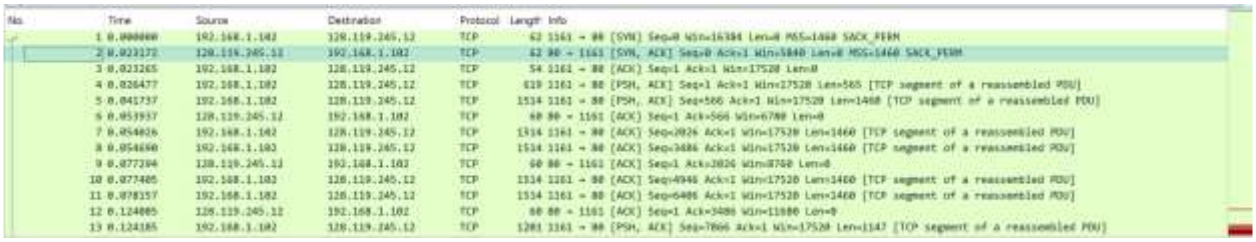


→ The length of the sixth TCP segment is 1147 bytes:



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 received for the entire trace: 5840
- The lack of receiver buffer space never ever throttle the sender

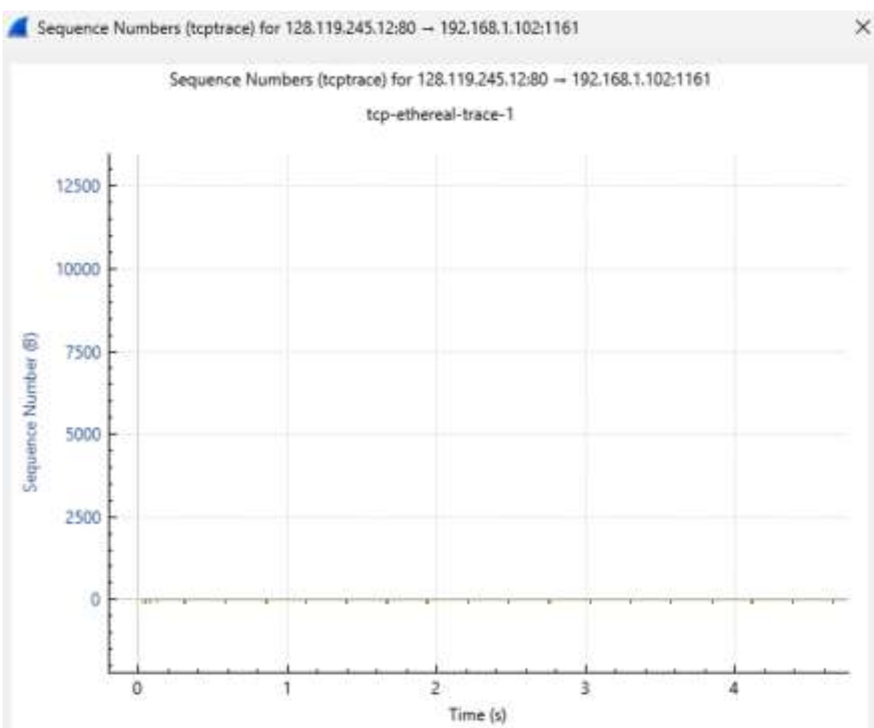


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.102	128.119.245.12	TCP	62	1161 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM
2	0.023172	128.119.245.12	192.168.1.102	TCP	62	80 → 1161 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM
3	0.023265	192.168.1.102	128.119.245.12	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 [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]

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.

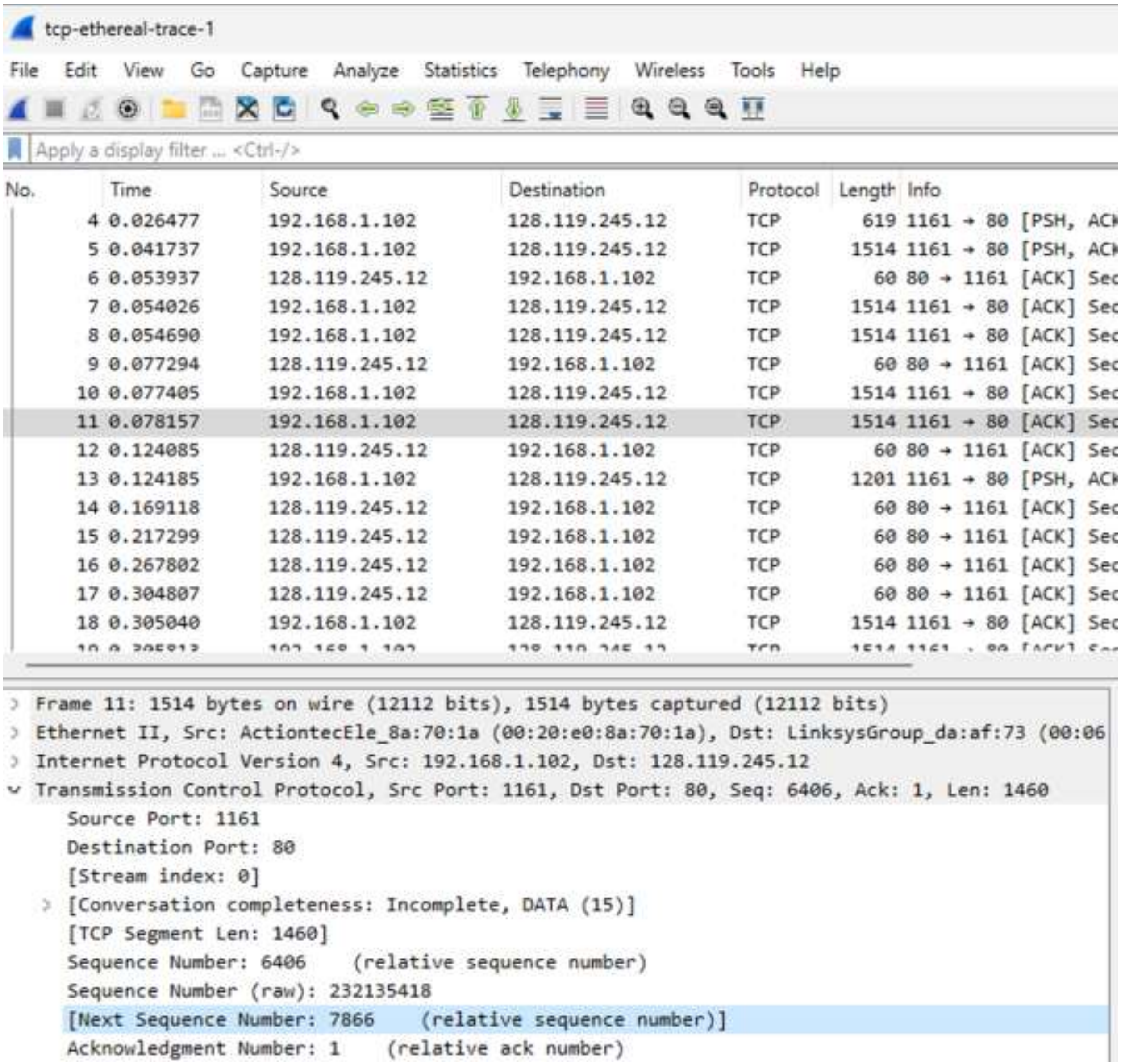
We can verify this by checking the sequence numbers of the TCP segments in the trace file. In the Time Sequence-Graph (Stevens) of this trace, all sequence numbers from the source (192.168.1.102) to the destination (128.119.245.12) are increasing monotonically with respect to time. If there is a retransmitted 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 250 in the text).

	acknowledged sequence number	acknowledged data
ACK 1	566	565
ACK 2	2026	1460
ACK 3	3486	1460
ACK 4	4946	1460
ACK 5	6406	1460
ACK 6	7866	1460
ACK 7	9013	1147

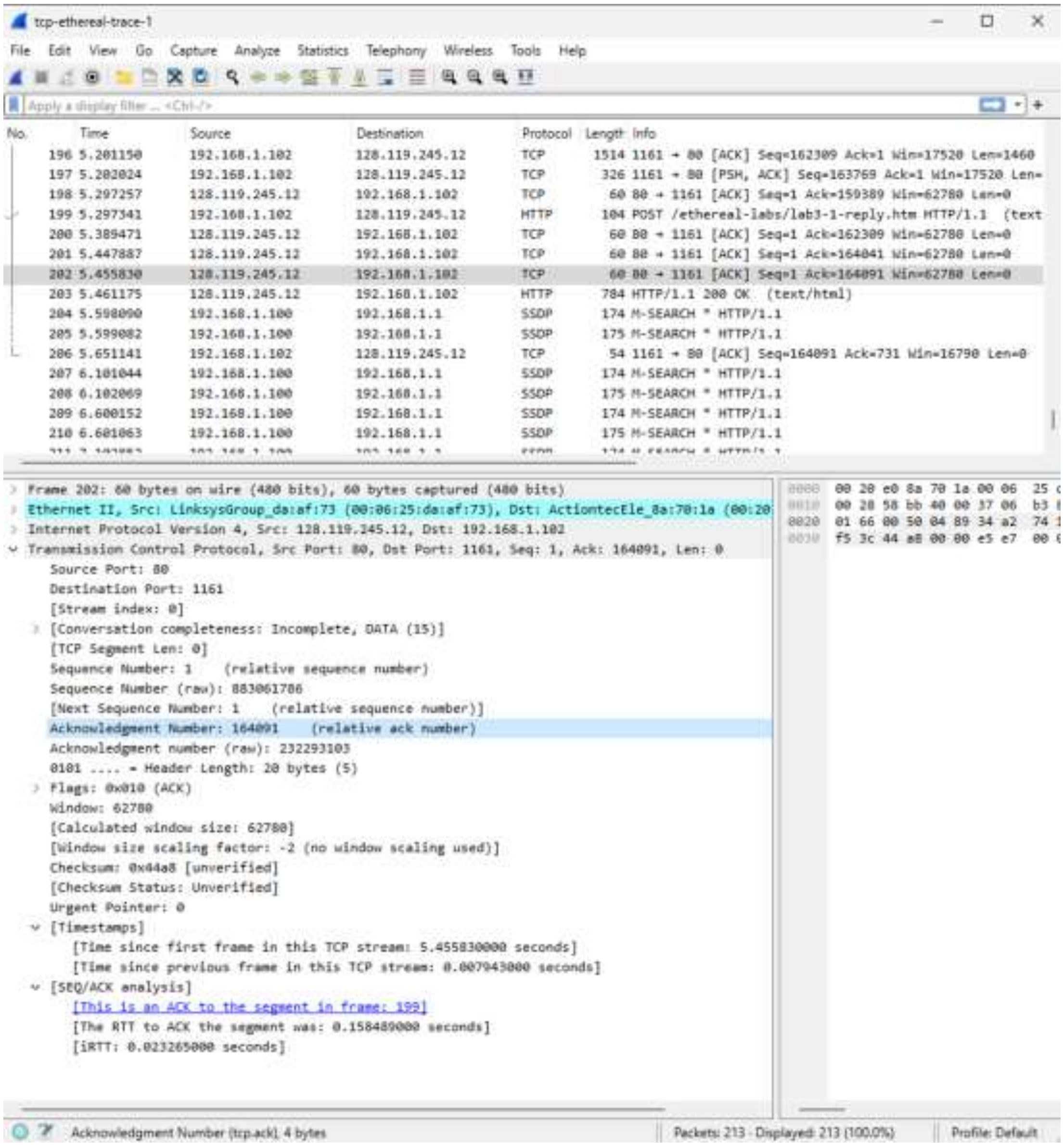
Example for the ACK 6:



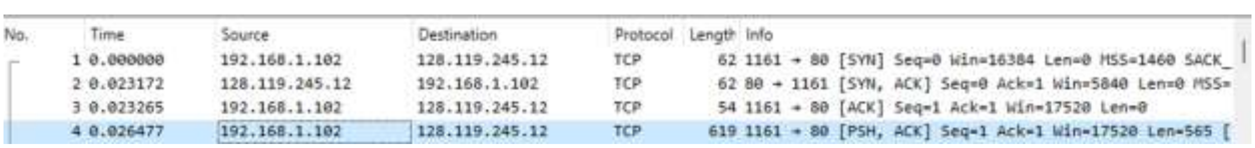
The difference between the acknowledged sequence numbers of two consecutive ACKs indicates the data received by the server between these two ACKs. By inspecting the amount of acknowledged data by each ACK, there are cases where the receiver is ACKing every other segment. For example, segment of No. 80 acknowledged data with 2920 bytes = 1460*2 bytes.

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

The computation of TCP throughput largely depends on the selection of averaging time period. As a common throughput computation, in this question, we select the average time period as the whole connection time. Then, the average throughput for this TCP connection is computed as the ratio between the total amount data and the total transmission time. The total amount data transmitted can be computed by the difference between the sequence number of the first TCP segment (i.e. 1 byte for No. 4 segment) and the acknowledged sequence number of the last ACK (164091 bytes for No. 202 segment). Therefore, the total data are 164091 - 1 = 164090 bytes. The whole transmission time is the difference of the time instant of the first TCP segment (i.e., 0.026477 second for No.4 segment) and the time instant of the last ACK (i.e., 5.455830 second for No. 202 segment). Therefore, the total transmission time is 5.455830 - 0.026477 = 5.4294 seconds. Hence, the throughput for the TCP connection is computed as 164090/5.4294 = 30.222 KByte/sec.



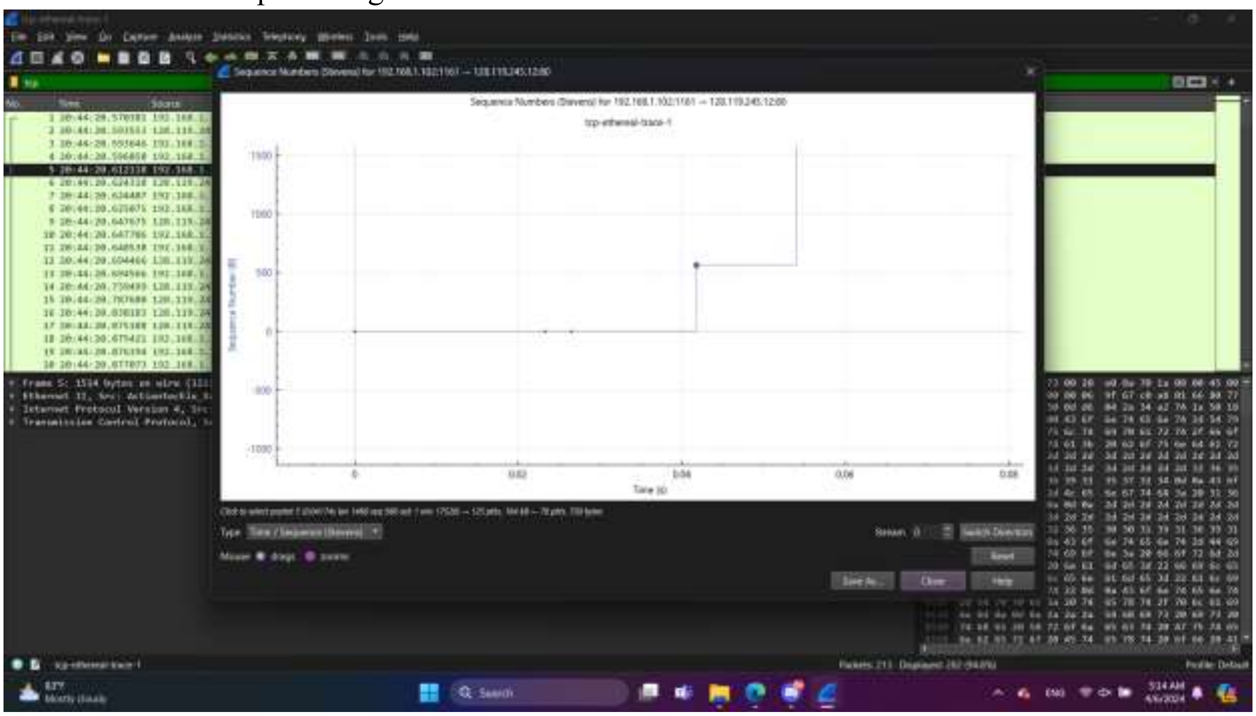
No.4 segment:



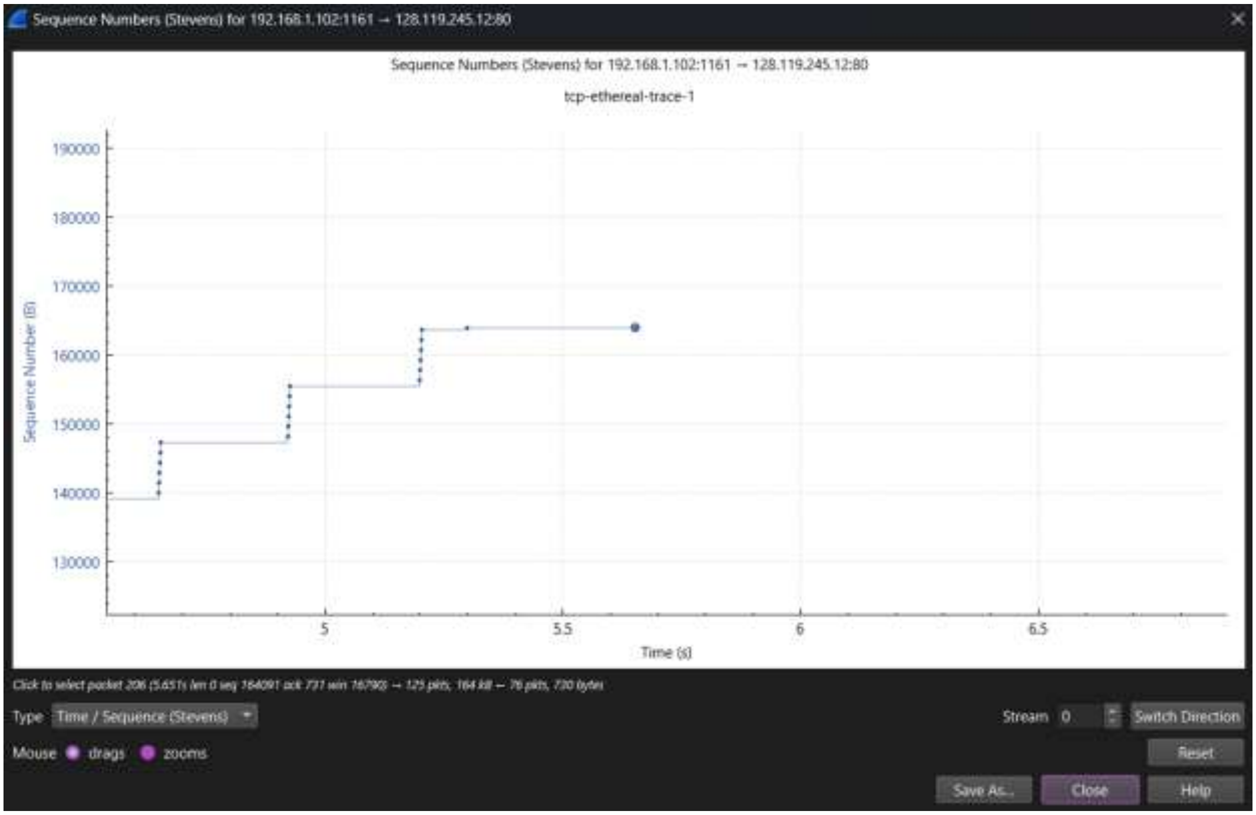
Task 4

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.

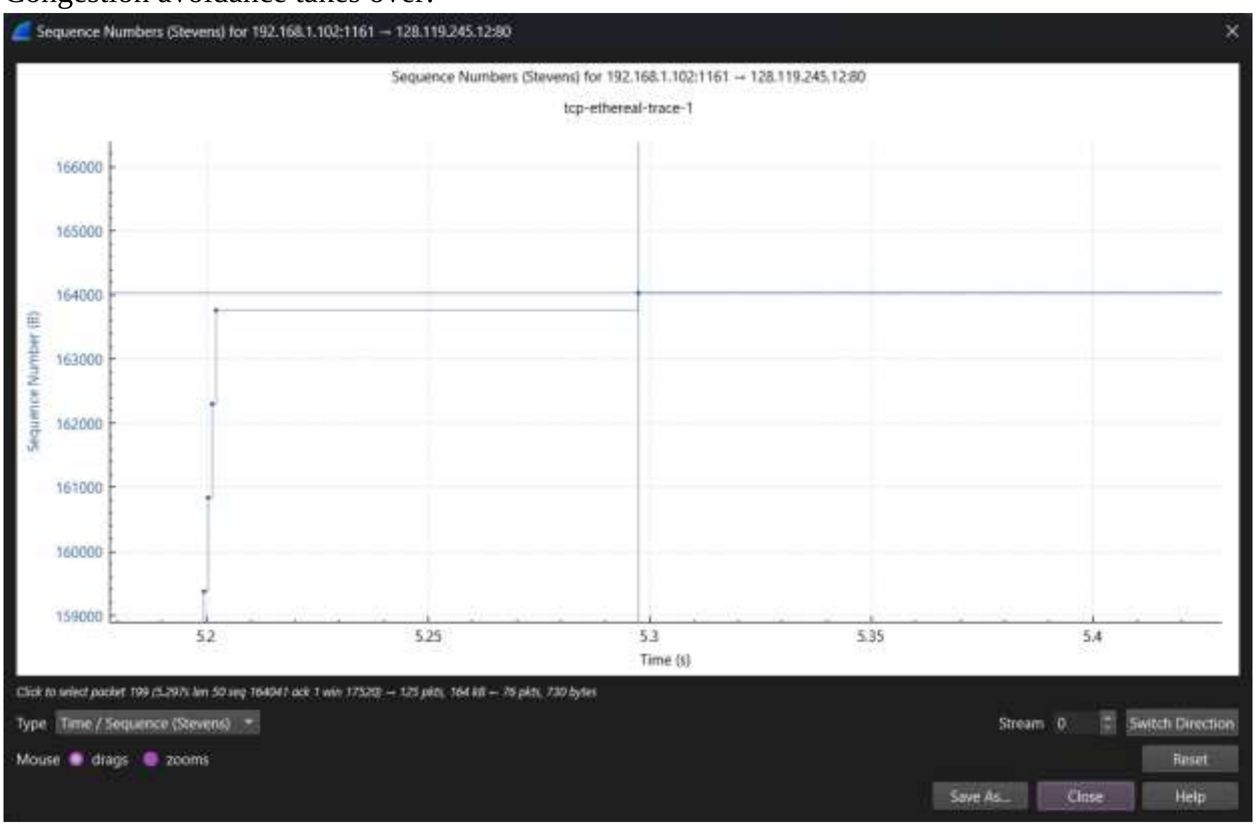
→ TCP's slowstart phase begins:



TCP's slowstart phase ends:



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:

By observing the plot, we can see that the slow-start phase only lasts for first 1-1.5 second. Afterwards, it seems that the TCP session is always in congestion avoidance state. In this case, we do not observe the expected linear increase behaviour, i.e. the TCP transmit window does not grow linearly during this phase. In fact, it appears that the sender transmits packets in batches of 6. This does not seem to be caused by flow control since the receiver advertised window is significantly larger than 5 packets. The reason for this behaviour might be due to the fact that the HTTP server has enforced a rate-limit of some sort.