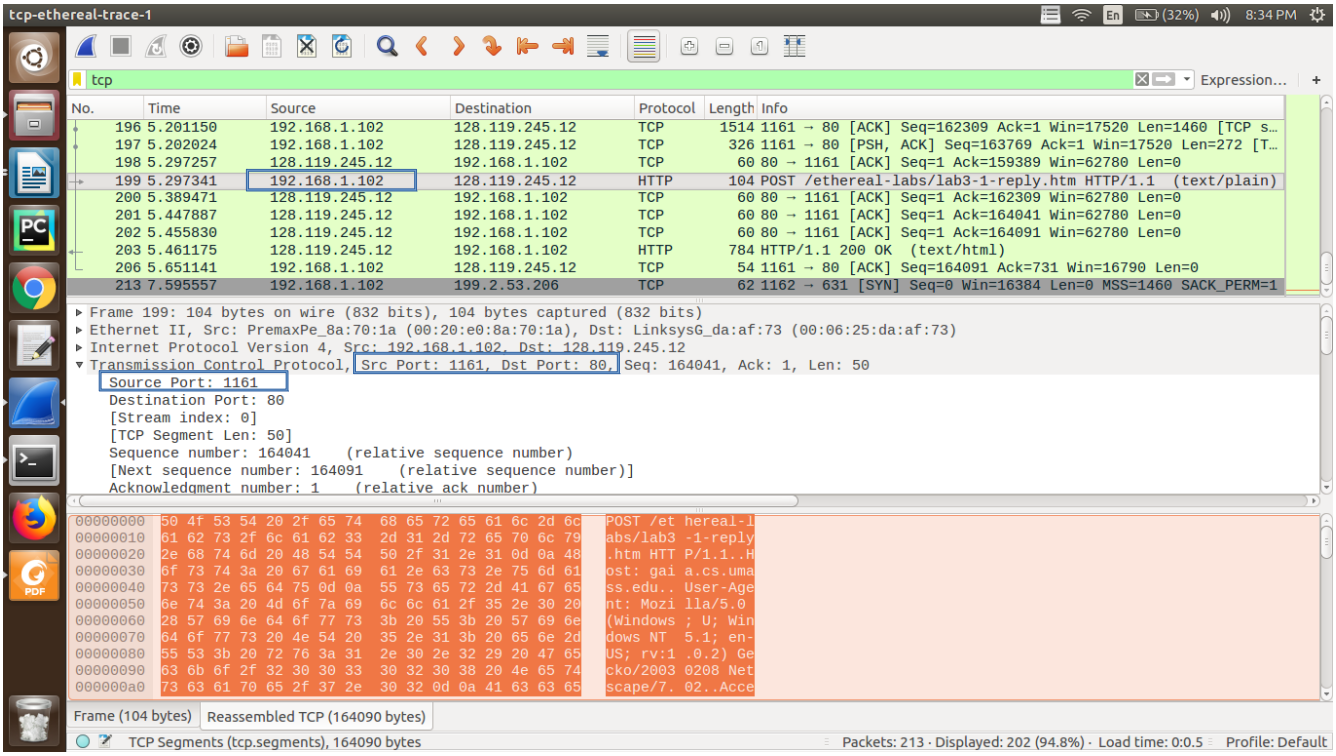
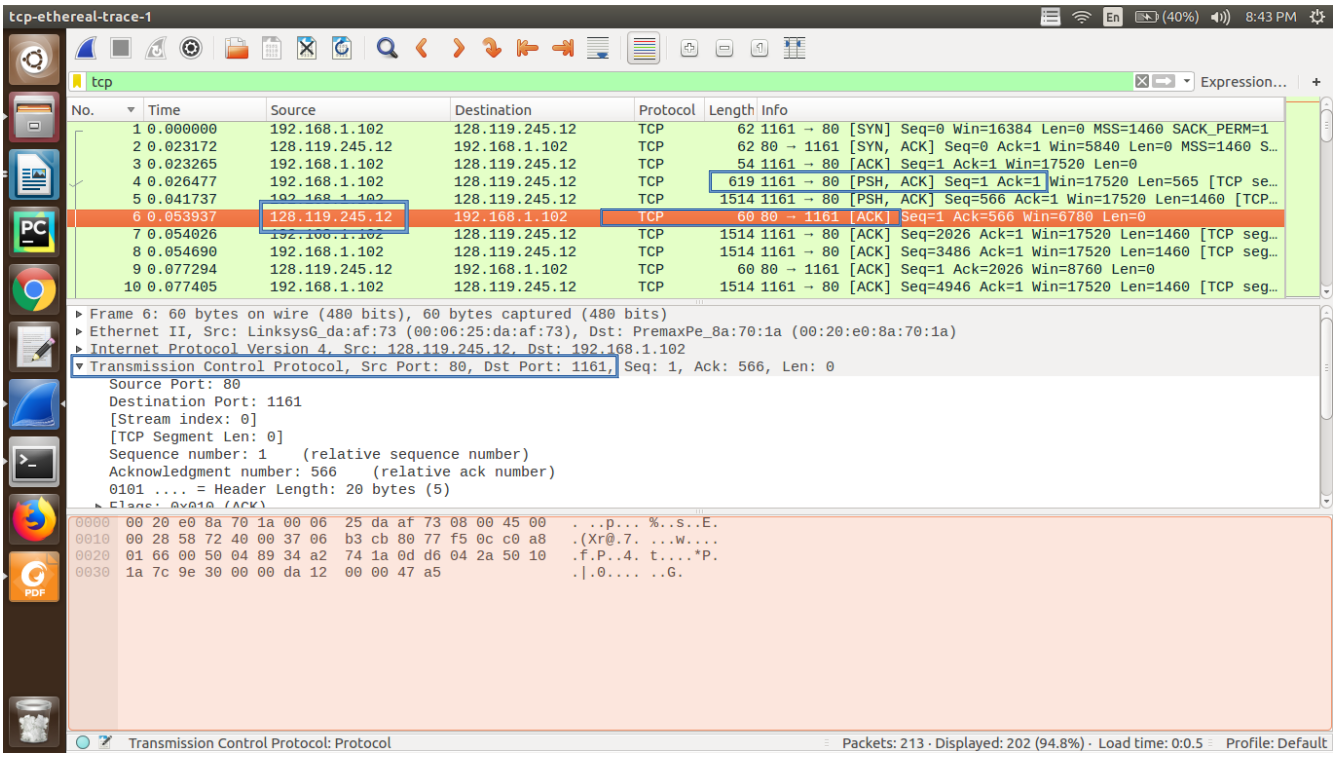
Used Trace files for all questions except question no 3.

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".
Ans: Source IP Address :- 192.168.1.102
TCP Port number used by the client : 1161

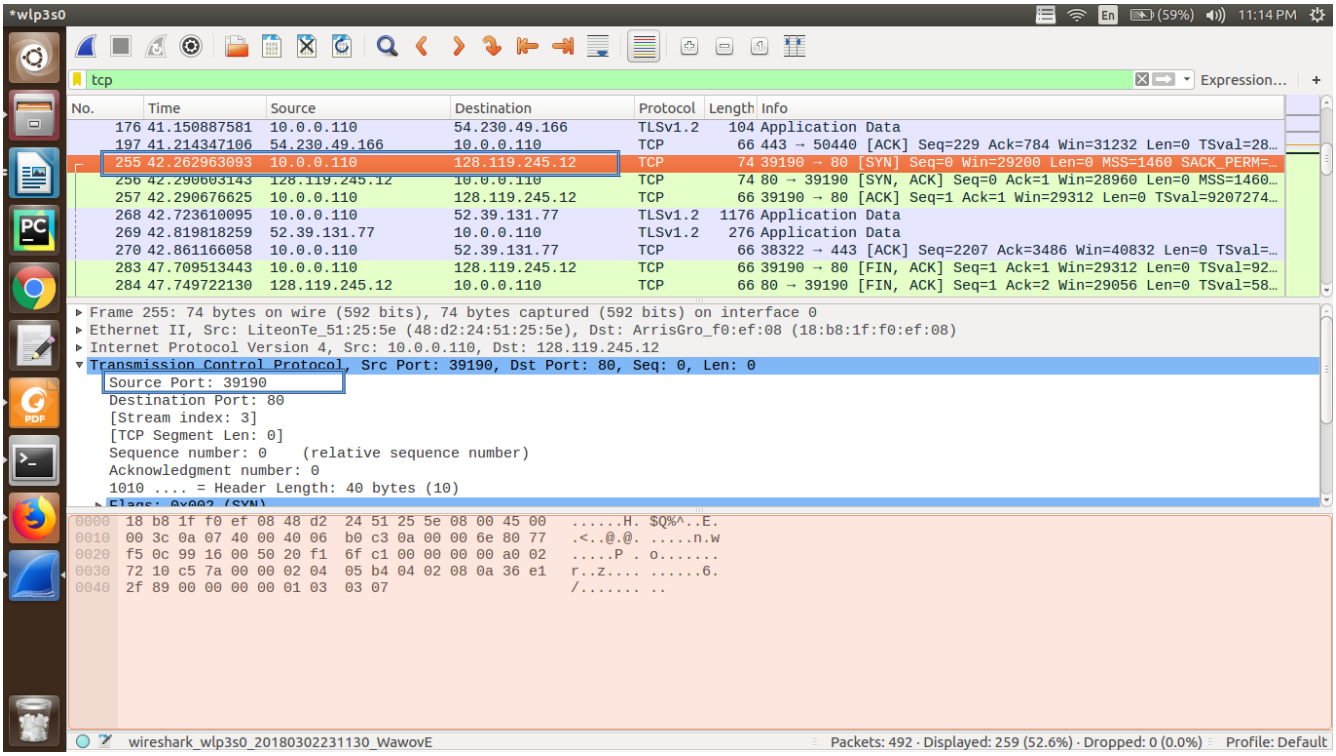


2. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

Ans:- 1) IP Address of gaia.cs.umass.edu = **128.119.245.12**
2) gaia.cs.umass.edu is receiving TCP connections on Port 80 and sending response to this to client on port 1161.

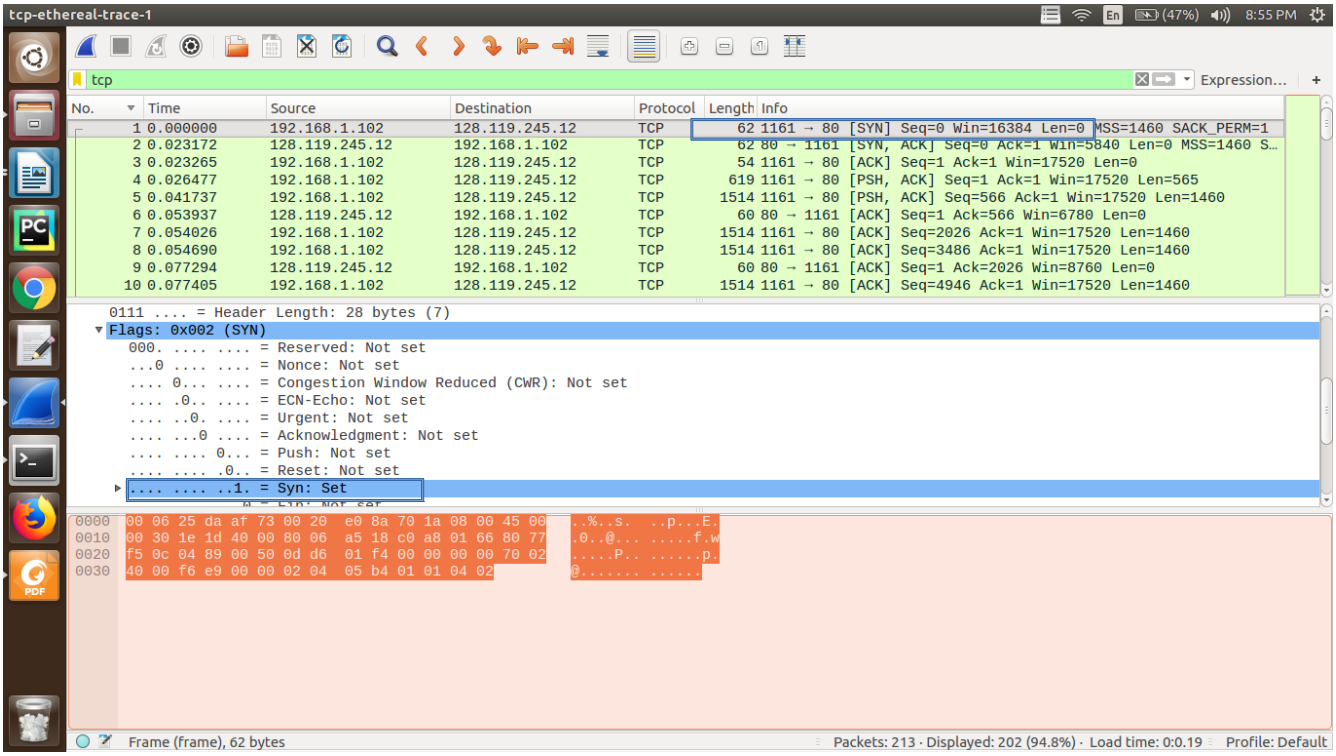


3. What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?
Ans: IP address : 10.0.0.110
Port used by client : 39190



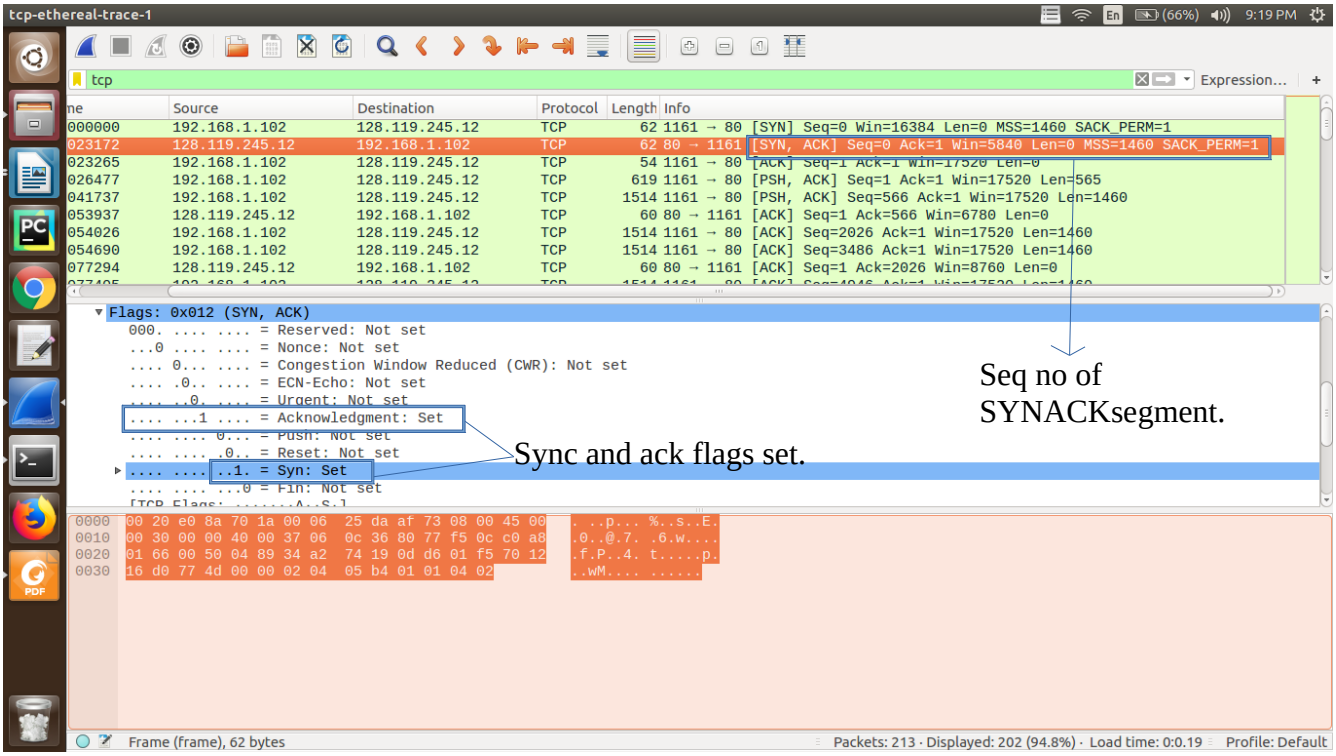
4. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?

Ans:- Sequence number of the packet in which TCP SYN segment used to indicate the TCP connection is **'seq=0'.** The SYN Flag being set to **'1'** in the flags section identifies the segment as a SYN segment. Below Screen shot highlights the sequence number used by the client to initiate TCP connection and highlights the SYN flag being set to 1, this shows the handshake initiation.



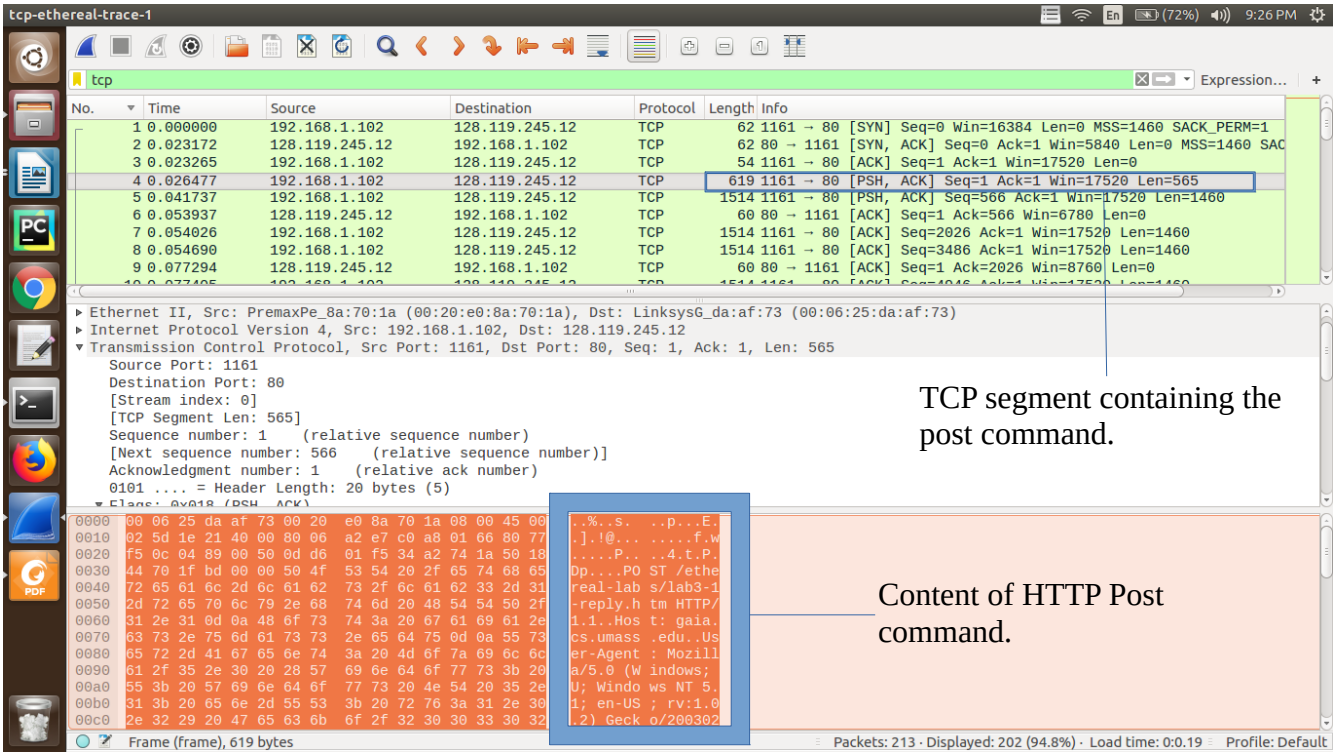
5. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK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

Ans :- 1) Sequence number of the SYNACK segment sent by gaia.cs.umass.edu to client in reply to SYN is **‘0’**.
2) Value of the acknowledgement field in the SYNACK segment is set to ‘1’.
3) When gaia.cs.umass.edu received the segment with seq=0 and SYN flag set to one. It understood that a new TCP connection initiation request has arrived with the flag set and to let the remote computer know and go further in the handshake process it sets the SYN and ACK flag to 1 determining the client that is ready to accept the connection and go ahead with the setup.
4) Both the SYN and ACK flag are set in the segment determines that it is a SYNACK segment.



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

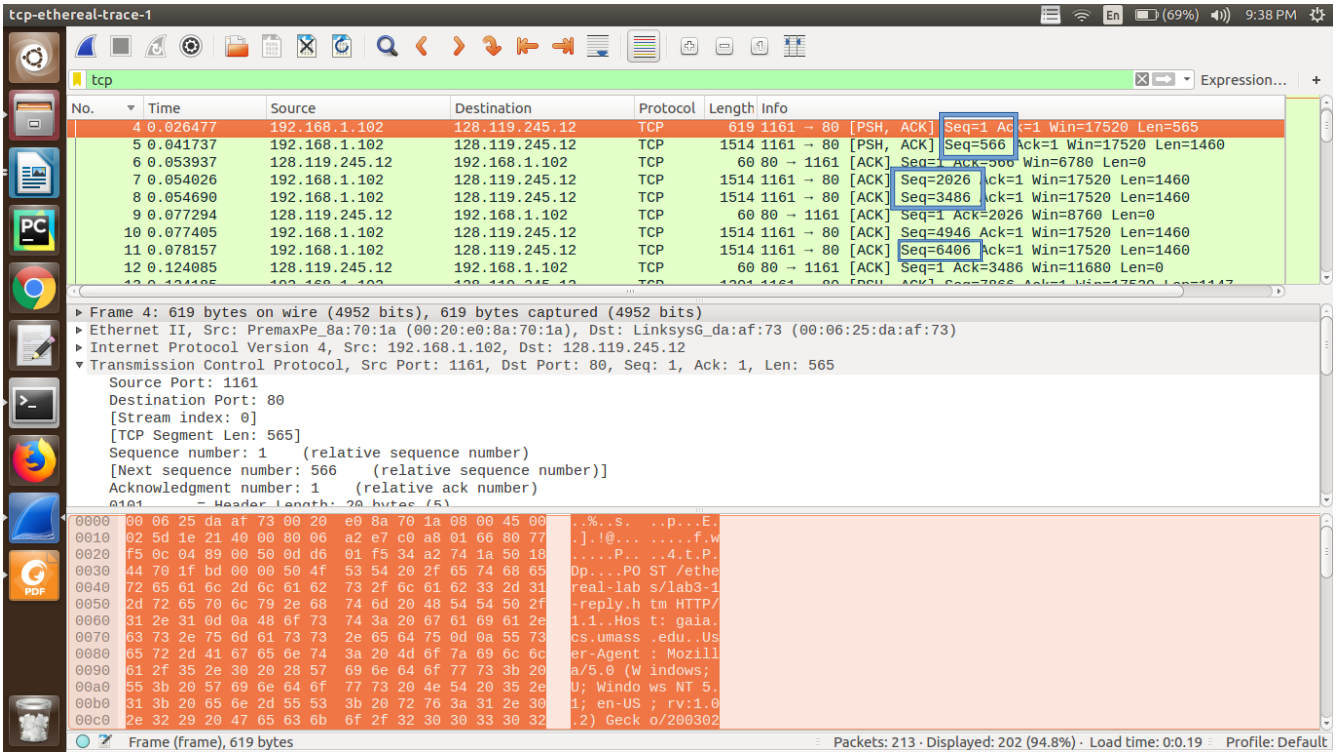
Ans:- The sequence number of the TCP segment containing the HTTP POST command is 1.



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 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.

Ans:-

1) The sequence number of the first six TCP segments are **1, 566, 2026, 3486, 4946, 6406.**



2) Time at which each segment was sent

1) Segment 1: Seq num = 1
   Time at which it was sent = **0.026477** seconds.
   Time at which ACK was recvd = **0.053937** seconds.
2) Segment 1: Seq num = 566
   Time at which it was sent = **0.041737** seconds.
   Time at which ACK was recvd = **0.077294** seconds.
3) Segment 1: Seq num = 2026
   Time at which it was sent = **0.054026** seconds.
   Time at which ACK was recvd = **0.0.124085** seconds.
4) Segment 1: Seq num = 3486
   Time at which it was sent = **0.054690** seconds.
   Time at which ACK was recvd = **0.169118** seconds.
5) Segment 1: Seq num = 4946
   Time at which it was sent = **0.077405** seconds.
   Time at which ACK was recvd = **0.217299** seconds.
6) Segment 1: Seq num = 6406
   Time at which it was sent = **0.078157** seconds.

Time at which ACK was recvd = **0.267802** seconds.

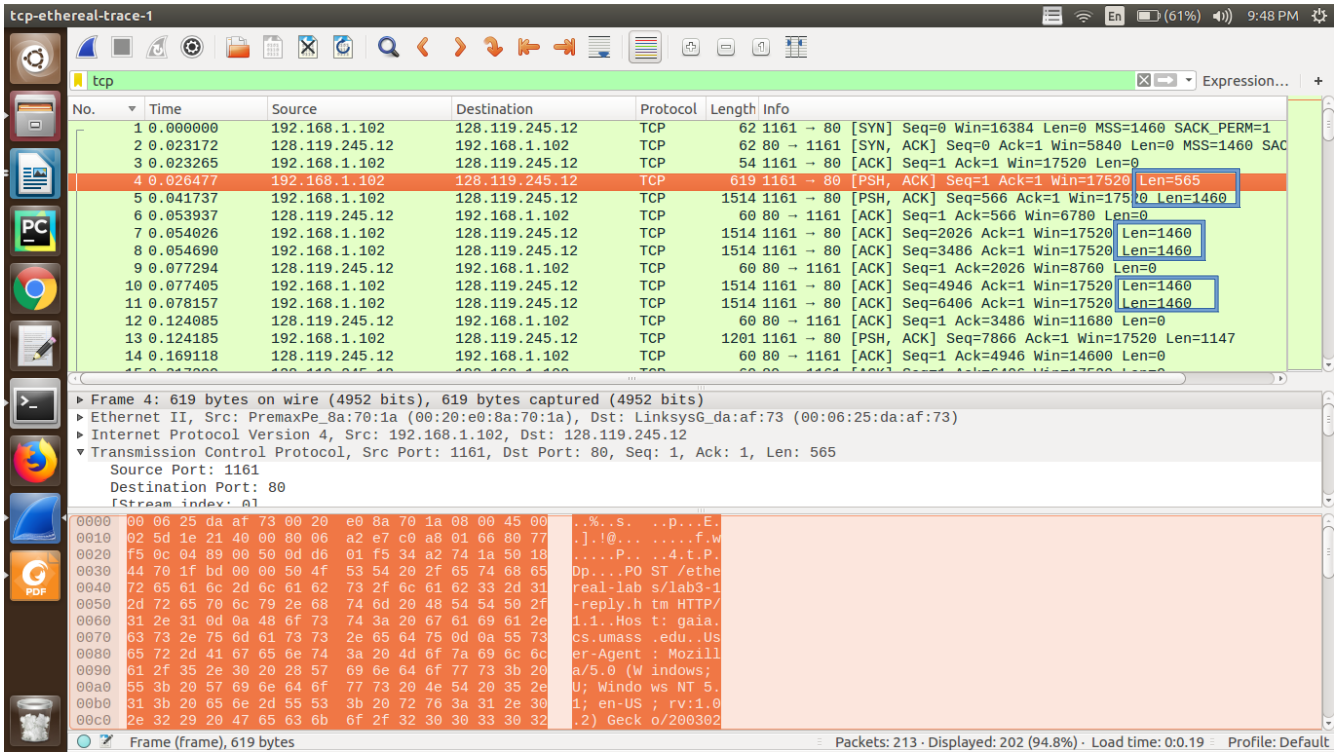
3) RTT value of all six segments

1) Segment 1 RTT1: **0.027460 seconds**
2) Segment 2 RTT2: **0.03555700 seconds**
3) Segment 3 RTT3: **0.070059000 seconds**
4) Segment 4 RTT4: **0.114428000 seconds**
5) Segment 5 RTT5: **0.19894000 seconds**
6) Segment 6 RTT6: **0.189645000 seconds**

4) Estimated RTT for six segments:

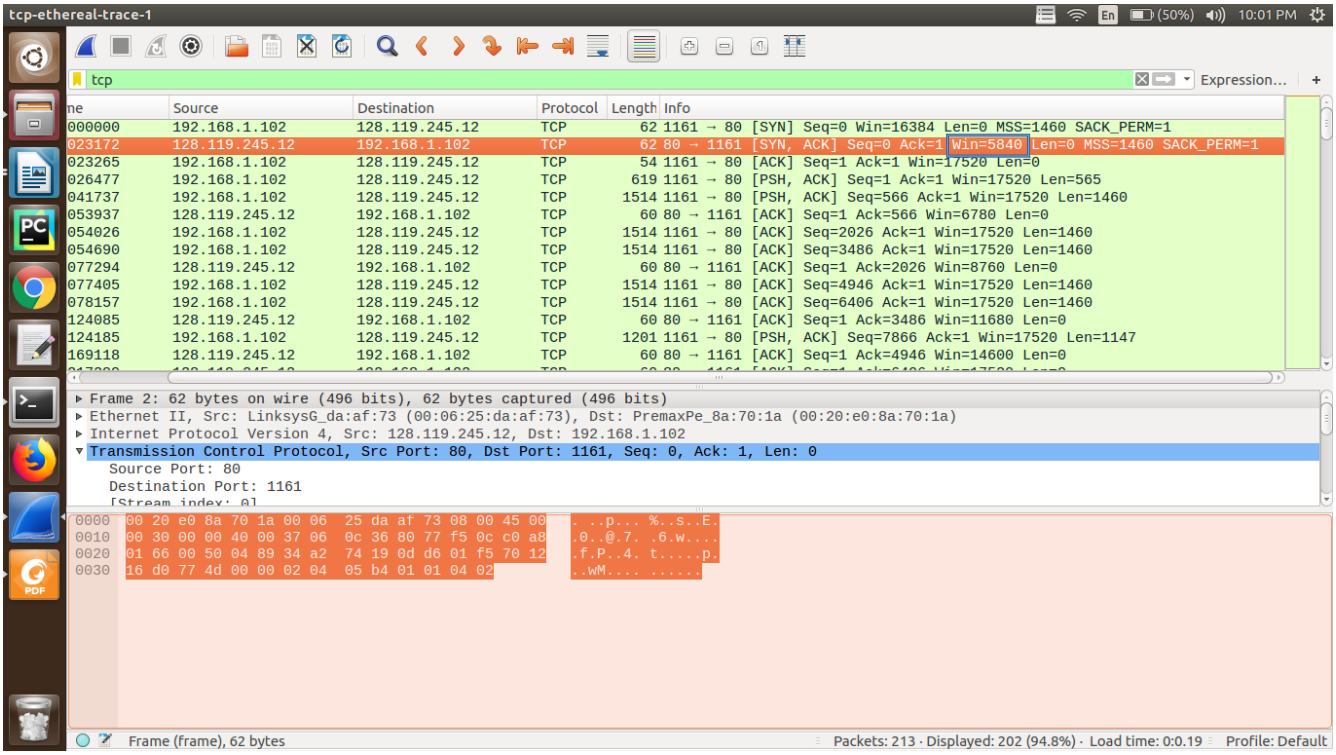
1) ERTT1 = **0.027460** (We will be taking it same as RTT1 as suggested)
2) ERTT2 = ((1 – 0.125) * 0.27460) + 0.125*0.03555700
   = **0.02847375**
3) ERTT3 = (0.875*0.02847375) + (0.125 * 0.070059000)
   = **0.033671906**
4) ERTT4 = (0.875*0.033671906) + (0.125 * 0.114428000)
   = **0.043766417**
5) ERTT5 = (0.875*0.043766417) + (o.125 * 0.139894000)
   = **0.05582364**
6) ERTT6 = (0.875 * 0.055782364) + (0.125 * 0.189645000)
   = **0.072515193**

8. What is the length of each of the first six TCP segments?
Ans: Length of first TCP segment is 565(segment which consists of the POST request.). And length of remaining TCP segments is 1460.

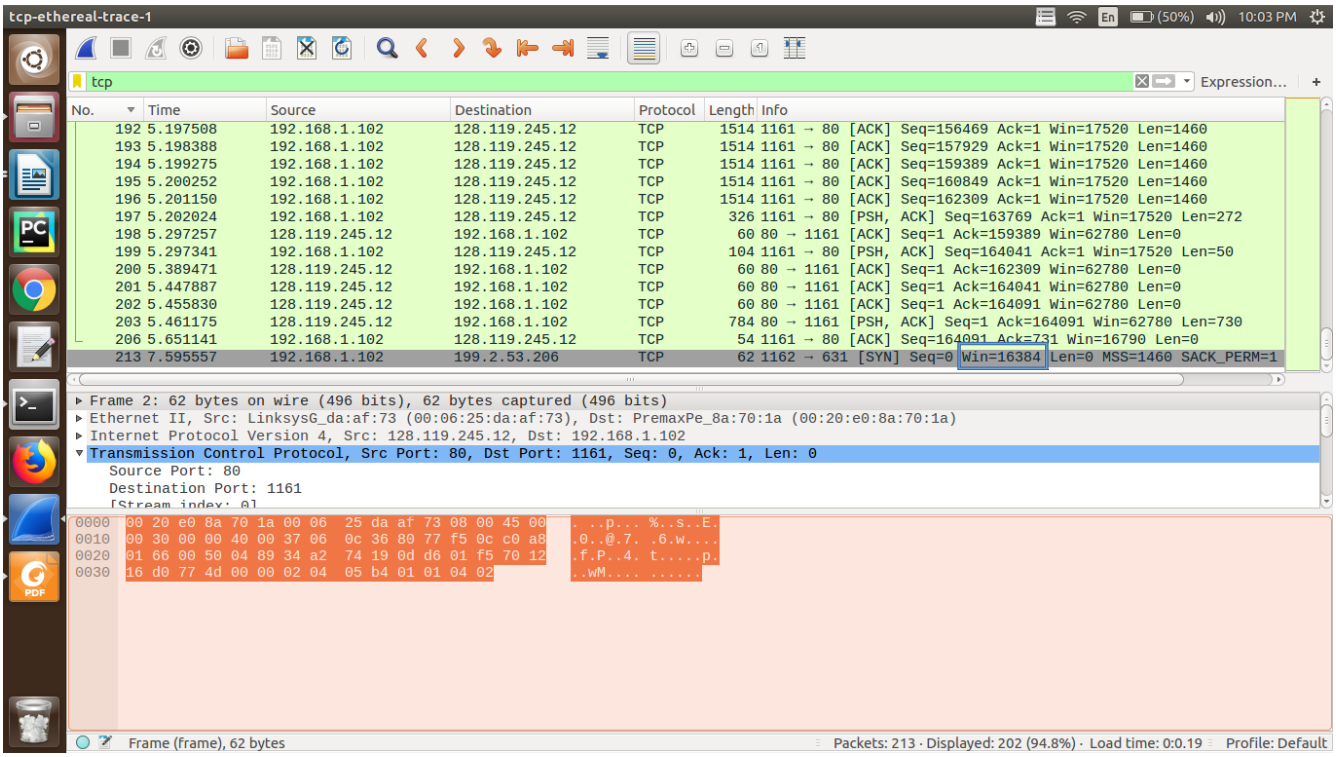


9. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?
Ans:- Minimum amount of Buffer space advertised for the entire space is 5480, which is seen when we receive the first acknowledgment from the server. And the maximum is 62870. Also, by inspecting the trace we can observe that the receiver buffer space never throttle the sender.

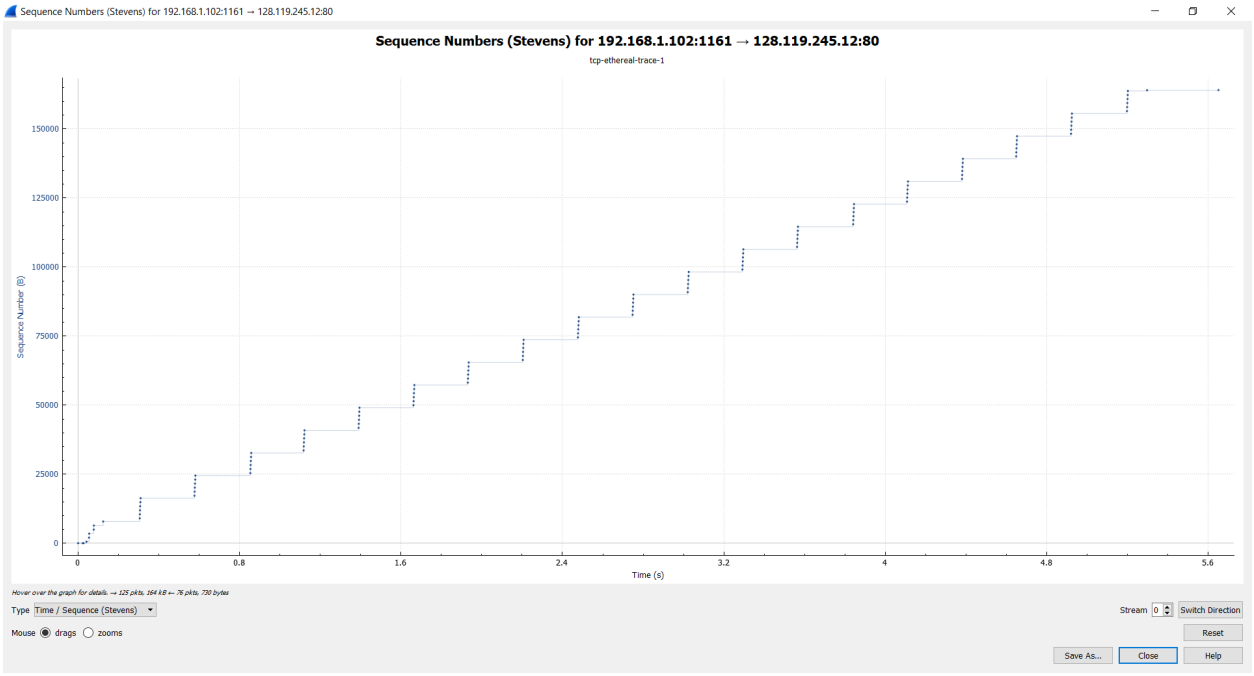
**minimum amount of buffer space advertised in the window field in the segment received from the server **



**maximum amount of buffer space that was advertised in the window filed in the segment received from the server **



10. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?
Ans:- No, there are no retransmitted segments in the trace file. We can check for the same using time sequence graph. As we observe the sequence numbers of the segments sent from host to server we can see that the sequence numbers are growing with time and there cannot be seen a drop of any previously sent packet. Because, if a segment had been retransmitted its sequence number would have been smaller than the previously sent segment which is not in this case and sequence number of the packets are steadily increasing in ascending order.



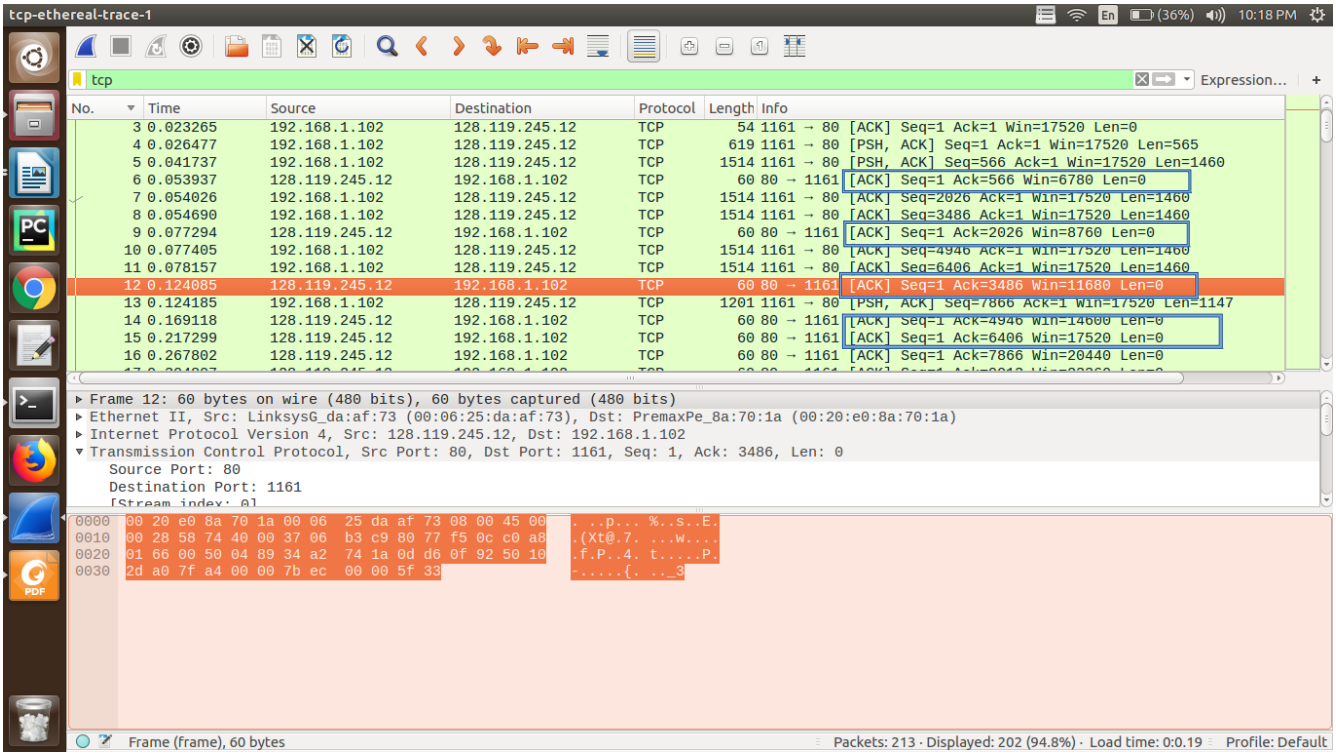
11. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment ?
Ans:- The receiver typically acknowledges 1460 bytes in an ACK. Yes, we can identify cases where receiver is ACKing every other received segment which can be seen from the example below.

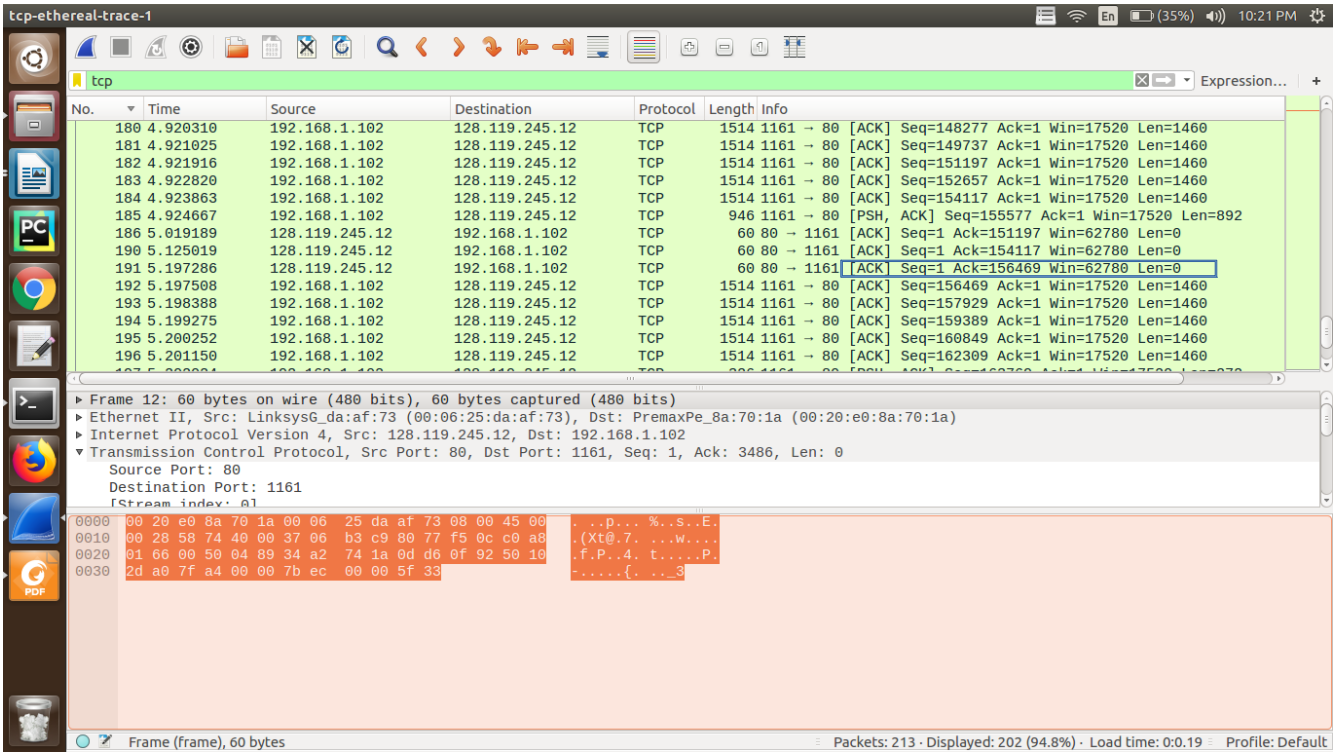
For Segment with sequence no 566 the receiver acknowledges with ACK = 2026 which is the next sequence number expected i.e. it previously acknowledged 2026-566 = 1460 bytes.

Similarly, For segment with sequence no 2026 the receiver acknowledge with ACK = 3486, which determines that it received 1460 bytes.

Similar it is the case for below few segments where for sequence no. 3486 we get an acknowledgement with ACK = 4946 i.e. after 1460 bytes. Similarly, for seq no 4946 we get and acknowledgement with ACK=6406.

Also, when we send segment with sequence number 155577 we receive acknowledgement with ACK=156469. i.e. receiver acknowledges it after receiving 1460 bytes.



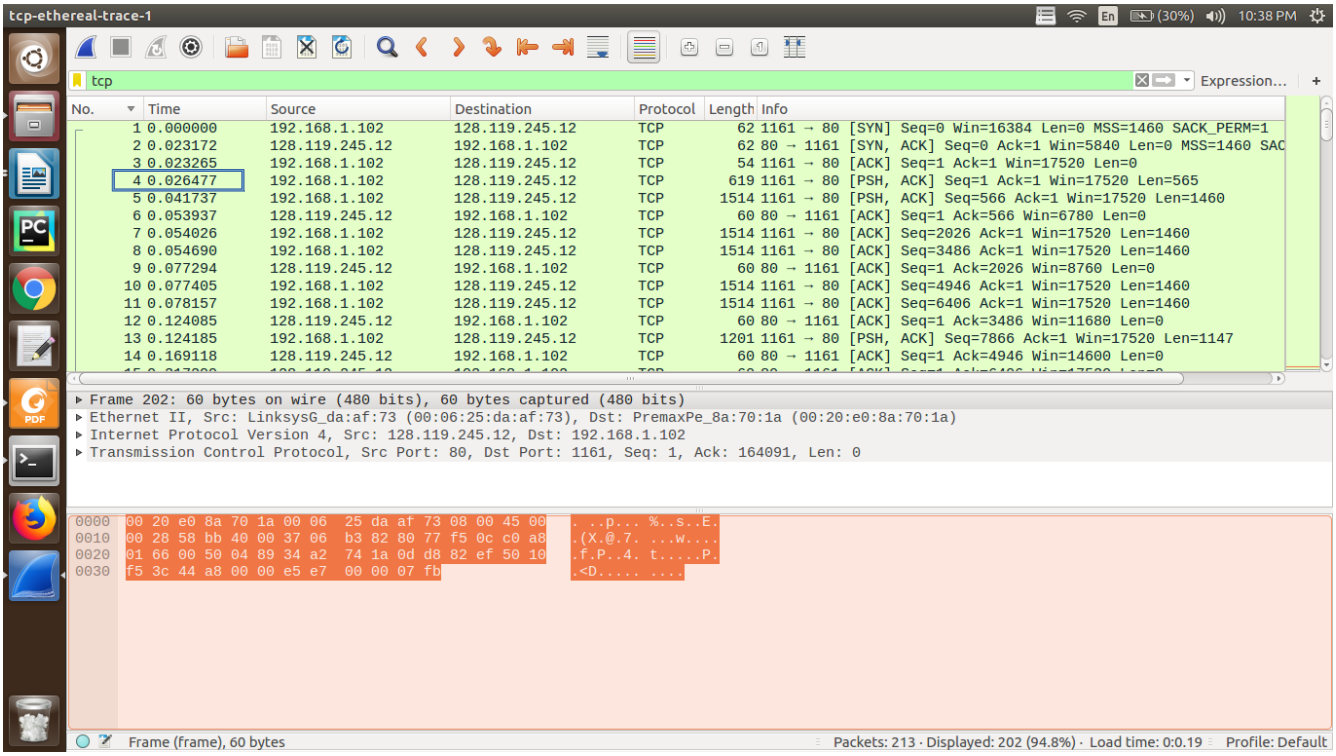


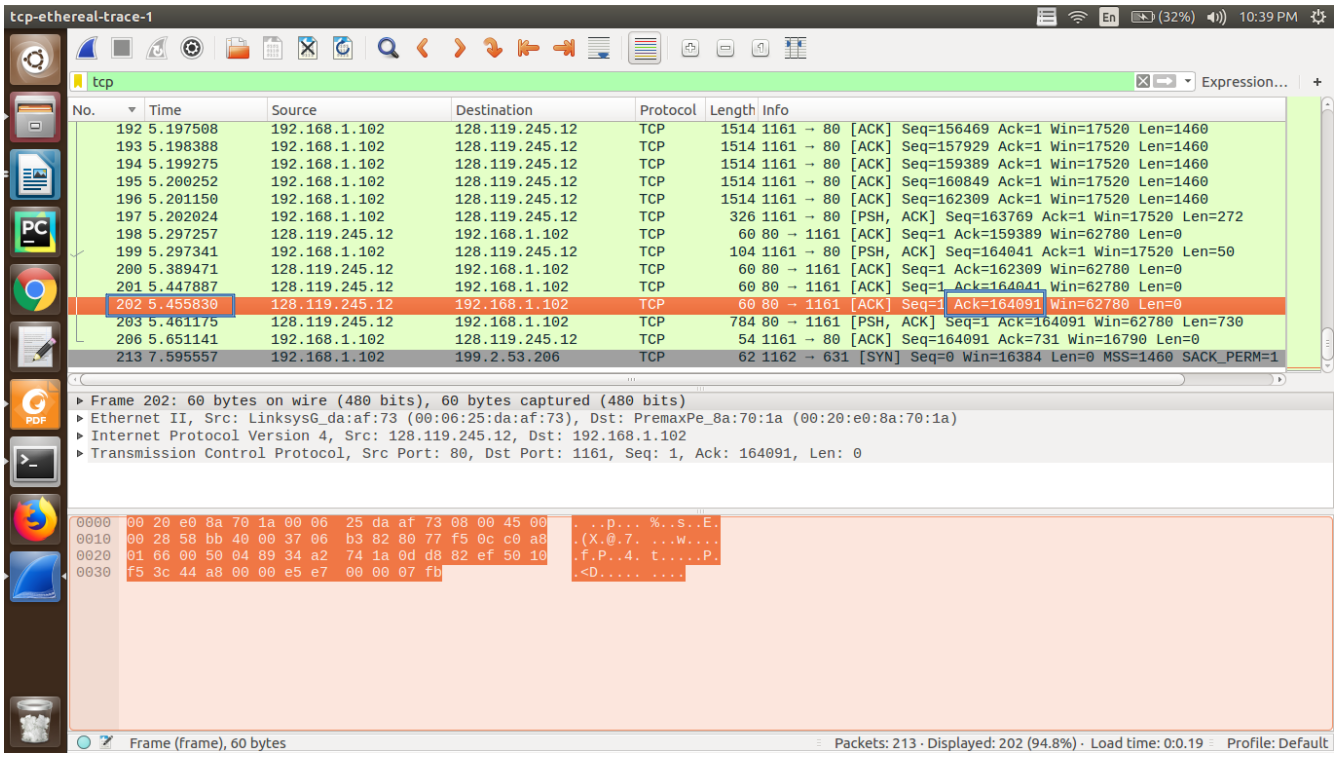
12. What is the throughput (bytes transferred per unit time) for the TCP connection?
Explain how you calculated this value.
Ans:- Total Number of bytes transferred to Server = 164090.
As we can see the last ACK received from server was ACK 164091, it determines that we have sent a total of 164090 bytes.

The Time at which acknowledgement was received for the last byte sent was received 5.455830. Time at which first segment was sent 0.026477.

Total time to transfer data from sender to receiver = 5.455830 – 0.026477
= 5.429353 seconds
The Throughput of TCP connection = (Total Bytes transferred)/ (Time taken to transfer those bytes)
Total TCP Throughput = (164090 / 5.429353) bytes/sec
= 30222.753 bytes/sec
= 30.222753 KiloBytes/sec

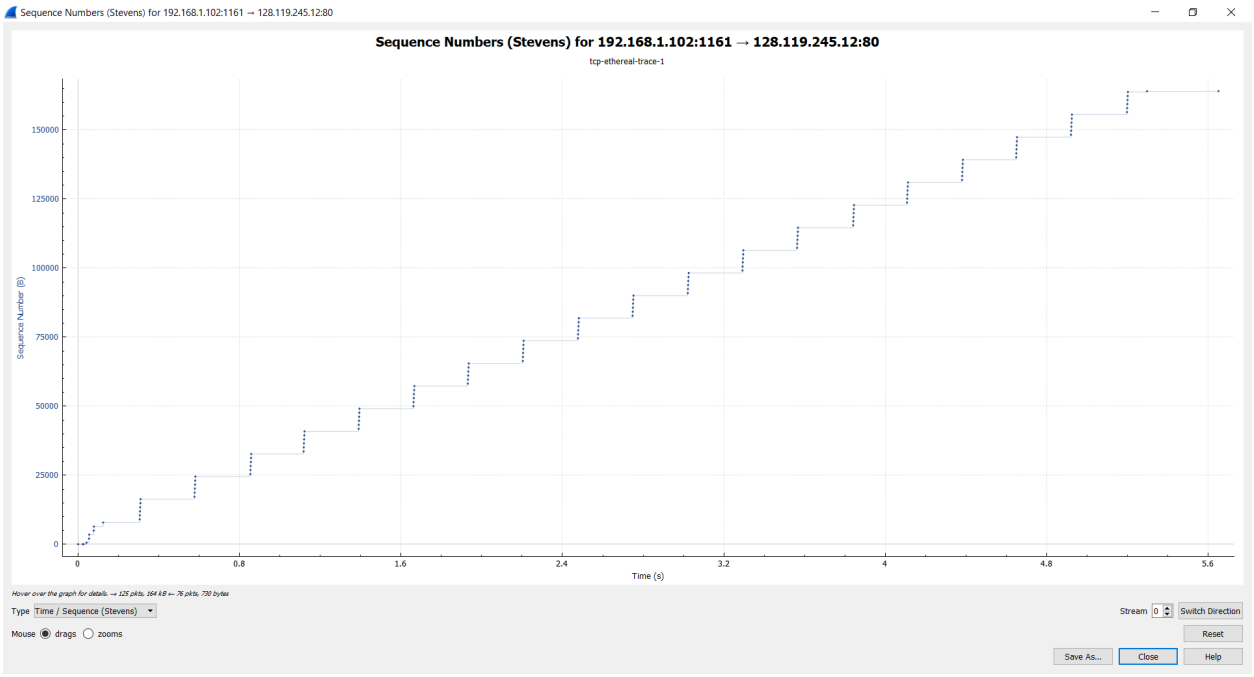
Below Screen Shots shows the times at which the first segment was sent and acknowledgment of the last bytes was received. It also shows the the ACK number received after the last TCP segment was sent which can be used for calculating the total amount that was being transferred from server to client.





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 slow start 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.

Ans:- The slow start phase begins at 0.02648 seconds and then it goes on till 0.1242 seconds and thereafter congestion avoidance takes over . The measured data never utilizes the complete capacity of the window size. We can see that the sender sends at the maximum of 6 packets in one burst and the TCP session is constantly in congestion avoidance state thereafter. And thus the measured data just uses $1/4^{th}$ of the window capacity and never makes use of the full capacity which it should do ideally.



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?

Ans:- The slow start phase begins at around ‘0’ seconds and goes upto 0.8 seconds and there onwards congestion avoidance takes over. Here as we is the same scenario as mentioned in the 13th qns, the sender never sends data to the full capacity of the window and just go on to maximum of 1/4th of the window size and never utilized maximum windows size.