Exercise 1: Understanding TCP using Wireshark

For this particular experiment, download the trace file: tcp-wireshark-trace-1 .

The following indicates the steps for this experiment:

Step 1 : Launch Wireshark by searching for it in the application finder.

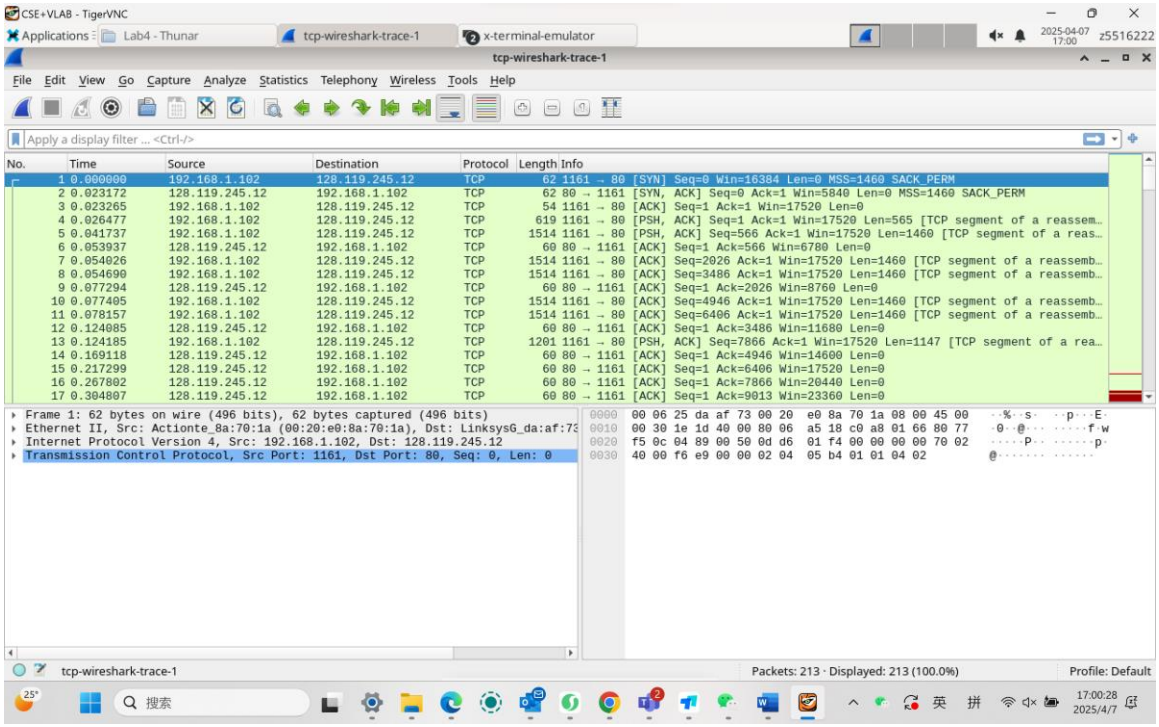
NOTE: If wireshark does not launch then type the following at a terminal while you are in the VLAB environment - "3331 wireshark".

Step 2: Load the trace file *tcp-ethereal-trace-1* by using the ***File*** pull-down menu, choosing ***Open*** and selecting the **appropriate trace file(downloaded ealier)** .

This file captures the sequence of messages exchanged between a host and a remote server (gaia.cs.umass.edu).

The host transfers a **150 KB text file** containing the text of Lewis Carrol's *Alice's Adventure in Wonderland* to the server.

Note that the file is being transferred from the host to the server using an HTTP POST message.



Step 3: Now filter out all non-TCP packets by typing " **tcp** " (without quotes) in the filter field towards the top of the Wireshark window. You should see a series of TCP segments between the host in MIT and gaia.cs.umass.edu.

The first three segments of the trace consist of the **initial three-way handshake containing the SYN, SYN ACK and ACK messages** .

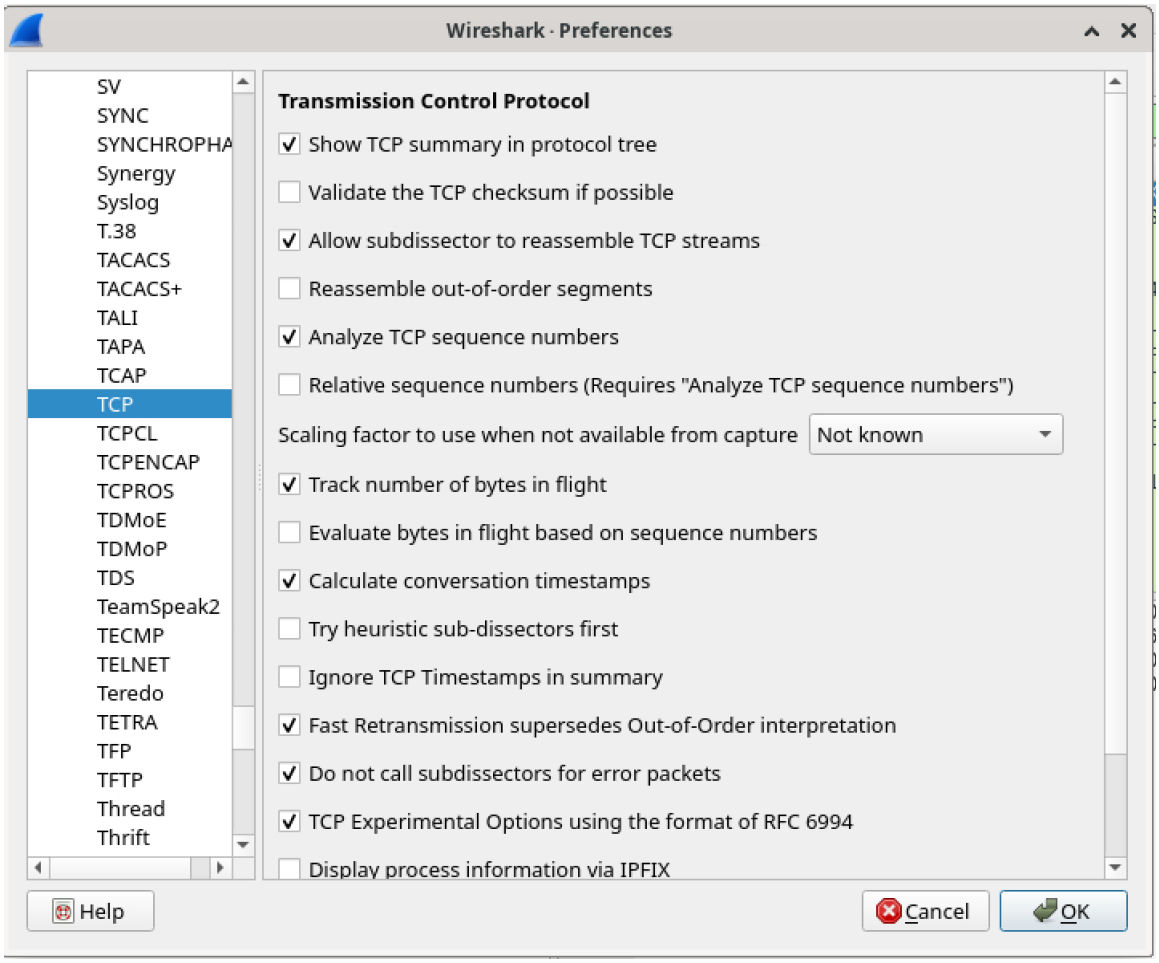
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.102	128.119.245.12	TCP	62	1161 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM
2	0.023172	128.119.245.12	192.168.1.102	TCP	62	80 → 1161 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM
3	0.023265	192.168.1.102	128.119.245.12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

You should see an HTTP POST message in the 4 th segment of the trace being sent from the host in MIT to gaia.cs.umass.edu (check the contents of the payload of this segment) .

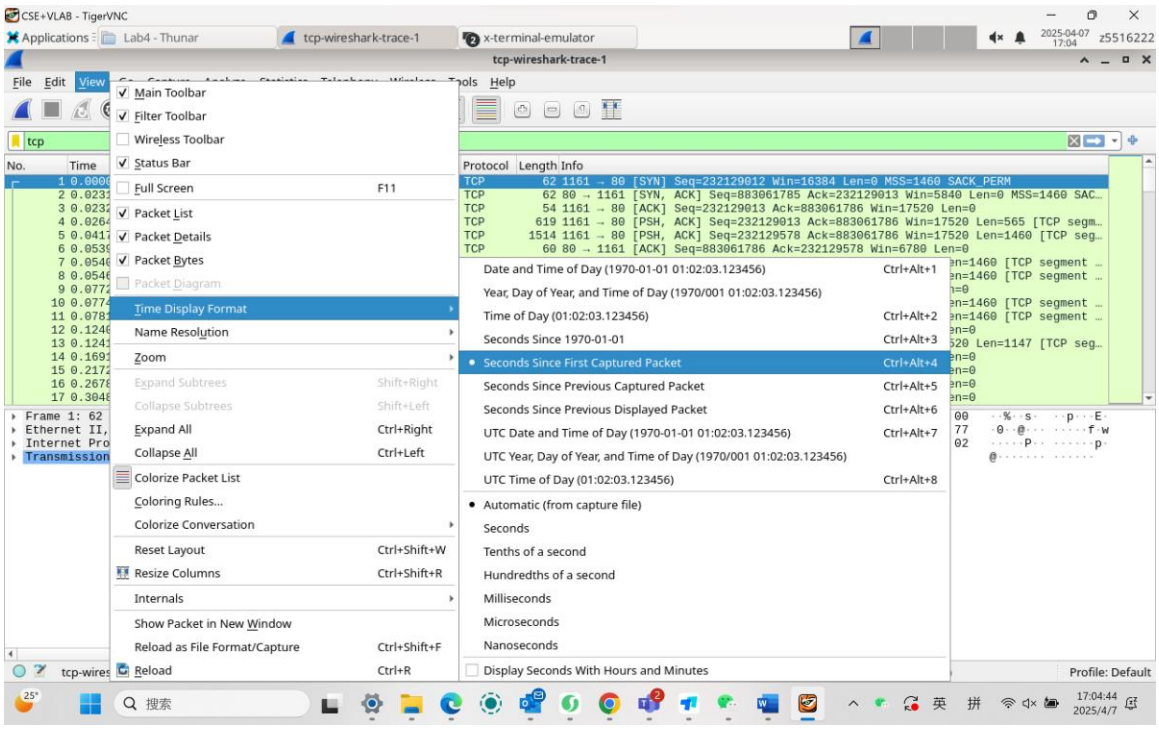
You should observe that the text file is transmitted as multiple TCP segments (i.e. a single POST message has been split into several TCP segments) from the client to the server (gaia.cs.umass.edu).

You should also see several TCP ACK segments being returned reversely.

IMPORTANT NOTE: Do the sequence numbers for the sender and receiver start from zero? This is because Wireshark, by default, scales down all real sequence numbers such that the first segment in the trace file always starts from 0. To turn off this feature, you have to click Edit->Preferences>Protocols->TCP (or Wireshark->Preferences->Protocols->TCP) and then disable the “Relative Sequence Numbers” option. Note that the answers in the solution set will reflect this change.



If you conduct the experiment without this change, the sequence numbers you observe will differ from those in the answers. Also, set the time shown in the 2nd column as the " Seconds since first captured packet " or " Seconds since begining of capture " under view->Time display format.



Question 1 . What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection? What are the IP address and TCP port numbers used by the client computer (source) that is transferring the file to gaia.cs.umass.edu?

```
Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.119.245.12
    0100 .... = Version: 4
    Destination Address: 128.119.245.12
Transmission Control Protocol, Src Port: 1161, Dst Port: 80,
    Source Port: 1161
    Destination Port: 80
    [Stream index: 0]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 0]
    Sequence Number: 232129012
    [Next Sequence Number: 232129013]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    0111 .... = Header Length: 28 bytes (7)
```

Answer 1:

gaia.cs.umass.edu: IP = 128.119.245.12, port = 80

Client: IP = 128.119.245.12, port = 1161

Question 2 . What is the sequence number of the TCP segment containing the HTTP POST command?

Note that 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.

```
Transmission Control Protocol, Src Port: 1161, Dst Port: 80, Seq: 232293053, Ack: 883061786
    Source Port: 1161
    Destination Port: 80
    [Stream index: 0]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 50]
    Sequence Number: 232293053
```

Answer 2: The sequence number is 232293052

Question 3. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection.

(a) What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST) sent from the client to the webserver (Do not consider the ACKs received from the server as part of these six segments)?

(b) 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?

(c) What is the *EstimatedRTT* value (see relevant parts of Section 3.5 or lecture slides) after receiving each ACK? Assume that the initial value of *EstimatedRTT* is equal to the measured RTT (*SampleRTT*) for the first segment and then is computed using the *EstimatedRTT* equation for all subsequent segments. Set alpha to 0.125.

Note: Wireshark has a nice feature that allows you to plot the RTT for each TCP segment 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* . However, do not use this graph to answer the above question.

(d) What is the length of each of the first six TCP segments?

Answer 3:

(a) tcp.srcport == 1161 && tcp.dstport == 80 && tcp.flags.syn == 0

No.	Time	Source	Destination	Protocol	Length	Info
3	0.023265	192.168.1.102	128.119.245.12	TCP	54	1161 → 80 [ACK] Seq=232129013 Ack=883061786 Win=17520 Len=0
4	0.026477	192.168.1.102	128.119.245.12	TCP	619	1161 → 80 [PSH, ACK] Seq=232129013 Ack=883061786 Win=17520 Len=565 [TCP segm…
5	0.041737	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [PSH, ACK] Seq=232129578 Ack=883061786 Win=17520 Len=1460 [TCP seg…
7	0.054026	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=232131038 Ack=883061786 Win=17520 Len=1460 [TCP segment …
8	0.054690	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=232132498 Ack=883061786 Win=17520 Len=1460 [TCP segment …
10	0.077405	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=232133958 Ack=883061786 Win=17520 Len=1460 [TCP segment …
11	0.078157	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=232135418 Ack=883061786 Win=17520 Len=1460 [TCP segment …
13	0.124185	192.168.1.102	128.119.245.12	TCP	1201	1161 → 80 [PSH, ACK] Seq=232136878 Ack=883061786 Win=17520 Len=1147 [TCP seg…
18	0.305040	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=232138025 Ack=883061786 Win=17520 Len=1460 [TCP segment …
19	0.305813	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=232139485 Ack=883061786 Win=17520 Len=1460 [TCP segment …
20	0.306692	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=232140945 Ack=883061786 Win=17520 Len=1460 [TCP segment …
21	0.307571	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=232142405 Ack=883061786 Win=17520 Len=1460 [TCP segment …
22	0.308699	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=232143865 Ack=883061786 Win=17520 Len=1460 [TCP segment …
23	0.309553	192.168.1.102	128.119.245.12	TCP	946	1161 → 80 [PSH, ACK] Seq=232145325 Ack=883061786 Win=17520 Len=892 [TCP segm…

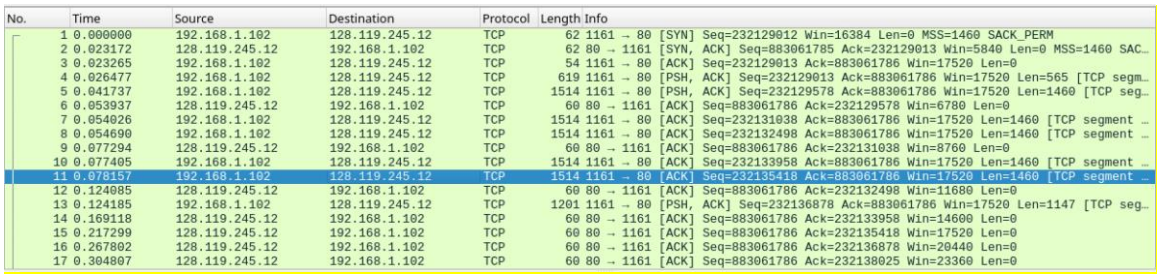
The sequence numbers of the first six segments in the TCP connection are: 232129013, 232129578, 232131038, 232132498, 232133958, 232135418.

RTT = Time ACK – Time Sent

$EstimatedRTT_1 = SampleRTT_1 = 0.028213$

$EstimatedRTTnew = 0.875 \times EstimatedRTTold + 0.125 \times SampleRTT$

(b)(c) and (d)Are Shown in the table below:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.102	128.119.245.12	TCP	62	1161 → 80 [SYN] Seq=232129012 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=883061785 Ack=232129013 Win=5840 Len=0 MSS=1460 SAC…
3	0.023265	192.168.1.102	128.119.245.12	TCP	54	1161 → 80 [ACK] Seq=232129013 Ack=883061786 Win=17520 Len=0
4	0.026477	192.168.1.102	128.119.245.12	TCP	619	1161 → 80 [PSH, ACK] Seq=232129013 Ack=883061786 Win=17520 Len=565 [TCP segm…
5	0.041737	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [PSH, ACK] Seq=232129578 Ack=883061786 Win=17520 Len=1460 [TCP seg…
6	0.053937	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=883061786 Ack=232129578 Win=6780 Len=0
7	0.054026	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=232131038 Ack=883061786 Win=17520 Len=1460 [TCP segment …
8	0.054690	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=232132498 Ack=883061786 Win=17520 Len=1460 [TCP segment …
9	0.077294	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=883061786 Ack=232131038 Win=8760 Len=0
10	0.077405	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=232133958 Ack=883061786 Win=17520 Len=1460 [TCP segment …
11	0.078157	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=232135418 Ack=883061786 Win=17520 Len=1460 [TCP segment …
12	0.124085	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=883061786 Ack=232132498 Win=11680 Len=0
13	0.124185	192.168.1.102	128.119.245.12	TCP	1201	1161 → 80 [PSH, ACK] Seq=232136878 Ack=883061786 Win=17520 Len=1147 [TCP seg…
14	0.169118	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=883061786 Ack=232133958 Win=14600 Len=0
15	0.217299	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=883061786 Ack=232135418 Win=17520 Len=0
16	0.267802	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=883061786 Ack=232136878 Win=20440 Len=0
17	0.304807	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=883061786 Ack=232138025 Win=23360 Len=0

ACK Receiving No.	Time Sent (s)	ACK Receiving Time (s)	RTT (s)	Length	EstimatedRTT (α=0.125)
6	0.026477	0.053937	0.02746	565	0.02746
9	0.041737	0.077294	0.035557	1460	0.028472125
12	0.054026	0.124085	0.070059	1460	0.033670484
14	0.05469	0.169118	0.114428	1460	0.043765174
15	0.077405	0.217299	0.139894	1460	0.055781277
16	0.078157	0.267802	0.189645	1460	0.072514242

Question 4. What is the minimum amount of available buffer space advertised at the receiver for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

Answer 4: The minimum amount of available buffer space is 5840. Yes.

Question 5. Are there any retransmitted segments in the trace file? To answer this question, what did you check for (in the trace)?

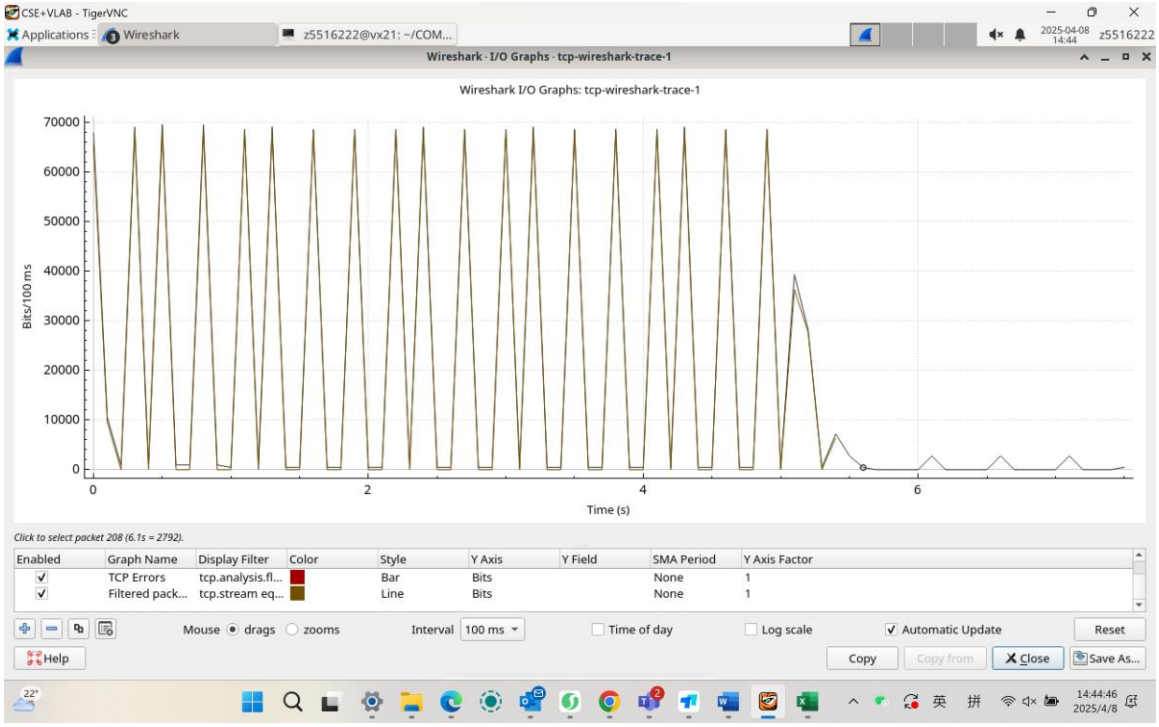
Answer 5: There are no retransmitted segments, checked by tcp.analysis.retransmission command.

Question 6. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (recall the discussion about delayed acks from the lecture notes or Section 3.5 of the text)?

Answer 6: The receiver does not strictly follow delayed ACK but acknowledges each data packet individually. The amount of data acknowledged depends on the packet length (565 bytes or 1460 bytes).

Question 7. What is the TCP connection's throughput (bytes transferred per unit of time during the connection)?

Explain how you calculated this value.



Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
125	171 kB	76	5 kB	0.000000	5.6511	241 kbps	7,483 bits/s

Answer 7: throughput A to B: 241kbps; throughput B to A: 7483bps

Exercise 2: TCP Connection Management

Consider the following TCP transaction between a client (10.9.16.201) and a server (10.99.6.175).

No	Source IP	Destination IP	Protocol	Info
295	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [SYN] Seq=2818463618 win=8192 MSS=1460
296	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [SYN, ACK] Seq=1247095790 Ack=2818463619 win=262144 MSS=1460
297	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [ACK] Seq=2818463619 Ack=1247095791 win=65535
298	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [PSH, ACK] Seq=2818463619 Ack=1247095791 win=65535
301	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [ACK] Seq=1247095791 Ack=2818463652 win=262096
302	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [PSH, ACK] Seq=1247095791 Ack=2818463652 win=262144
303	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [ACK] Seq=2818463652 Ack=1247095831 win=65535
304	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [FIN, ACK] Seq=2818463652 Ack=1247095831 win=65535
305	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [FIN, ACK] Seq=1247095831 Ack=2818463652 win=262144
306	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [ACK] Seq=2818463652 Ack=1247095832 win=65535
308	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [ACK] Seq=1247095831 Ack=2818463653 win=262144

Answer the following questions:

Question 1 **.** What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and server?

Answer 1: In packet No.295, the client (10.9.16.201) sends a SYN with Seq=2818463618.

Question 2. What is the sequence number of the SYNACK segment sent by the server to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did the server determine that value?

Answer 2: The server's SYNACK (No.301) acknowledges the client's SYN by adding 1 to its Seq (124705790 → 1247095791). The server's Seq is its own ISN.

Question 3 **.** What is the sequence number of the ACK segment sent by the client computer in response to the SYNACK? What is the value of the Acknowledgment field in this ACK segment? Does this segment contain any data?

Answer 3: The sequence number is 1247095791, confirming the server's SYNACK. No payload is present.

Question 4 **.** Who has done the active close? Is it the client or the server? How you have determined this? What type of closure has been performed? 3 Segment (FIN/FINACK/ACK), 4 Segment (FIN/ACK/FIN/ACK) or Simultaneous close?

Answer 4: Active closer: Client (10.9.16.201).

Evidence: Client sends first FIN,ACK (No.304).

Closure type: 4-segment (FIN/ACK/FIN/ACK).

Client FIN → Server ACK → Server FIN → Client ACK. The full exchange requires 4 segments, indicating a standard bidirectional close.

Question 5 **.** How many data bytes have been transferred from the client to the server and from the server to the client during the whole duration of the connection? What relationship does this have with the Initial Sequence Number and the final ACK received from the other side?

Answer 5: Client→Server: 35 bytes (2818463653 - 2818463618). Server→Client: 0 bytes (only control packets).