

**1. What are the first and last packets for the POST request?**

From packet 4 which has sequence number 1 to packet 199 which has sequence number 164041.

We can find information about these segments in Wireshark in the packet-detail pane.

```
▼ [122 Reassembled TCP Segments (164090 bytes): #4(565), #5
  [Frame: 4, payload: 0-564 (565 bytes)]
  [Frame: 5, payload: 565-2024 (1460 bytes)]
  [Frame: 7, payload: 2025-3484 (1460 bytes)]
  [Frame: 8, payload: 3485-4944 (1460 bytes)]
  [Frame: 10, payload: 4945-6404 (1460 bytes)]
  [Frame: 11, payload: 6405-7864 (1460 bytes)]
  [Frame: 13, payload: 7865-9011 (1147 bytes)]
  [Frame: 18, payload: 9012-10471 (1460 bytes)]
  [Frame: 19, payload: 10472-11931 (1460 bytes)]

  [Frame: 194, payload: 159388-160847 (1460 bytes)]
  [Frame: 195, payload: 160848-162307 (1460 bytes)]
  [Frame: 196, payload: 162308-163767 (1460 bytes)]
  [Frame: 197, payload: 163768-164039 (272 bytes)]
  [Frame: 199, payload: 164040-164089 (50 bytes)]
  [Segment count: 122]
```

First  
segment

Last segment

**2. What is the IP address and the TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu?**

192.168.1.102 and port number is 1161. See information below!

```
► Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.119.245.12
▼ Transmission Control Protocol, Src Port: 1161, Dst Port: 80, Seq: 0, Len: 0
  Source Port: 1161
```

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

128.119.245.12 and port 80 is used for this connection. See information below!

```
► Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.1.102
▼ Transmission Control Protocol, Src Port: 80, Dst Port: 1161, Seq: 0, Ack: 1, Len: 0
  Source Port: 80
```

**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?**

Actual sequence number for TCP SYN is 232129012. It has a flag (SYN) that indicates this segment is a SYN.

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=1
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=1
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 data length 565 bytes]
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 data length 1460 bytes]
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 data length 1460 bytes]
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 data length 1460 bytes]
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 data length 1460 bytes]
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 data length 1460 bytes]
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 data length 1147 bytes]

```

[Stream index: 0]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 232129012
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
0111 .... = Header Length: 28 bytes (7)
Flags: 0x002 (SYN)

```

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?

Sequence number: **883061785**

ACKnowledgment number: **232129013**

This number is based on the last packet sequence number. In this case the last sequence number is **232129012** and therefore the gaia.cs.umass.edu send acknowledgement with number **232129013** which means that it has received data from to **232129012**.

The flag name (SYN, ACK) is displayed in the segment field.

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=1
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=1
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 data length 565 bytes]
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 data length 1460 bytes]
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 data length 1460 bytes]
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 data length 1460 bytes]
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 data length 1460 bytes]
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 data length 1460 bytes]
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 data length 1147 bytes]

```

[Stream index: 0]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 883061785
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 232129013
0111 .... = Header Length: 28 bytes (7)
Flags: 0x012 (SYN, ACK)

```

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

Packet 4: **232129013**. See picture below!



1	0.000000	192.168.1.102	128.119.245.1
2	0.023172	128.119.245.12	192.168.1.102
3	0.023265	192.168.1.102	128.119.245.1
4	0.026477	192.168.1.102	128.119.245.1
5	0.041737	192.168.1.102	128.119.245.1
6	0.053937	128.119.245.12	192.168.1.102
7	0.054026	192.168.1.102	128.119.245.1
8	0.054690	192.168.1.102	128.119.245.1
9	0.077294	128.119.245.12	192.168.1.102
10	0.077405	192.168.1.102	128.119.245.1
11	0.078157	192.168.1.102	128.119.245.1
12	0.124085	128.119.245.12	192.168.1.102
13	0.124185	192.168.1.102	128.119.245.1

▶ Frame 4: 619 bytes on wire (4952 bits), 619 bytes captured (4952) on interface 0  
 ▶ Ethernet II, Src: Actionte\_8a:70:1a (00:20:e0:8a:70:1a), Dst: L  
 ▶ Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.119.24  
 ▼ Transmission Control Protocol, Src Port: 1161, Dst Port: 80, Seq

Source Port: 1161  
 Destination Port: 80  
 [Stream index: 0]  
 [TCP Segment Len: 565]  
 Sequence number: 1 (relative sequence number)  
 Sequence number (raw): 232129013  
 [Next sequence number: 566 (relative sequence number)]  
 Acknowledgment number: 1 (relative ack number)  
 Acknowledgment number (raw): 883061786  
 0101 .... = Header Length: 20 bytes (5)

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 269 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 270 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.*

○ 4. 232129013	sent time: <b>0.026477</b>	Ack time: 0.053937 <b>(6)</b>
○ 5. 232129578	sent time: <b>0.041737</b>	Ack time: 0.077294 <b>(9)</b>
○ 7. 883061786	sent time: <b>0.054026</b>	Ack time: 0.124085 <b>(12)</b>
○ 8. 232131038	sent time: <b>0.054690</b>	Ack time: 0.169118 <b>(14)</b>
○ 10. 232132498	sent time: <b>0.077405</b>	Ack time: 0.217299 <b>(15)</b>
○ 11. 883061786	sent time: <b>0.078157</b>	Ack time: 0.267802 <b>(16)</b>

Here is an example of how we calculated RTT for each segment. Note that this information can be found in Wireshark also.

**RTT value for segment 4 = 0.053937 - 0.026477 = 0.02746**

**RTT values**

**4 = 0.02746**

**5 = 0.03555**

**7 = 0.07005**

**8 = 0.11442**

**10 = 0.13989**

**11 = 0.18964**

***EstimatedRTT = 0.875 \* EstimatedRTT + 0.125 \* SampleRTT(RTT value)***

An example how we calculate Estimated RTT for packet 5:

**Estimated RTT(5) = 0.875 (constant) \* 0.0274 (prev. EstRTT) +  
0.125 (constant) \* 0.03555(RTT value)**

**Estimated RTT**

**4 = 0.0274**

**5 = 0.0285**

**7 = 0.0337**

**8 = 0.0437**

**10 = 0.0558**

**11 = 0.0725**

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

**4. 565**

**5. 1460**

**7. 1460**

**8. 1460**

**10. 1460**

**11. 1460**

This information is taken from the packet-listing pane in Wireshark but also can be found in packet-detail pane for each segment as well.

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
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 c
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
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
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
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

9. 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?

Minimum amount buffer space is 5840 bytes which has sequence number 883061785. The lack of receiver buffer space never throttles the sender because the available size of buffer at server increases after each ACK sent. Because the maximum amount of bytes in each segment is 1460 there is no limit for sender to decrease the amount of sended data.

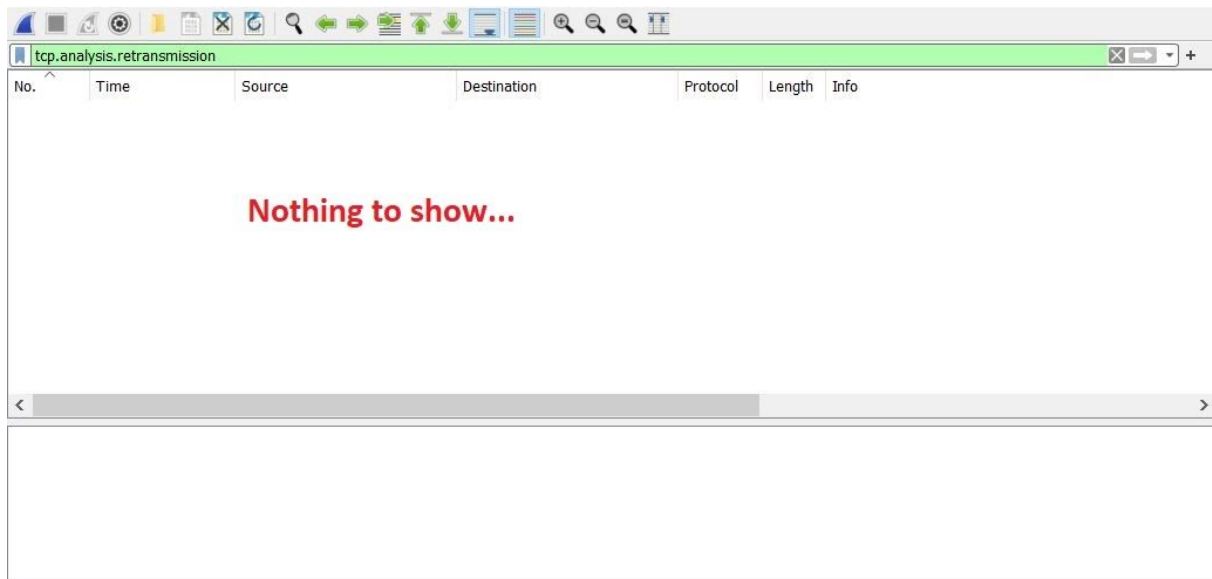
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.102	128.119.245.12	TCP	62	1161 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM=1
2	0.023172	128.119.245.12	192.168.1.102	TCP	62	80 → 1161 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
3	0.023265	192.168.1.102	128.119.245.12	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 reas
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 r
6	0.052027	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 r

Transmission Control Protocol, Src Port: 80, Dst Port: 1161, Seq: 0, Ack: 1, Len: 0
Source Port: 80
Destination Port: 1161
[Stream index: 0]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 883061785
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 232129013
0111 .... = Header Length: 28 bytes (7)
Flags: 0x012 (SYN, ACK)
Window: 5840

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, we used following filter which shows retransmitted segments in Wireshark:  
*tcp.analysis.retransmission*



- 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 278 in the text).**

Receiver typically acknowledge 1460 bytes in an ACK, which after observing the ACKs sent by receiver we discovered that receiver is ACKing every 1460 bytes.

Yes, for example packet No 69 is an ACK that acknowledges two segments (segment No 63 and 64). We can see that both packet No 63 and 64 are 1460 bytes long. The sum of these packets together is 2920 bytes. And when we calculate the difference between ACK No 62 and 69 which is  $44701 - 41781 = 2920$ . We can obviously see that also that there are no ACKs that acknowledge packet 63 but instead ACK No 69 ACKs both packet 64 of length 1460 and packet 63 of length 1460 which is totally 2920.

In the same manner this situation can be found in packets No 79 and 80. We are missing an ACK for packet No 76 and the reason is that packet No 80 ACKs both packet No 76 and 77 together. And again we can observe it by some calculation.

Packet No 79 ACKs for 55813

and then

Packet No 80 ACKs for 58165

which means (packet No 76 length + packet No 77 length) is equal to difference between these two ACKs.



tcp-ethereal-trace-1				tcp-ethereal-trace-1			
No.	Time	Info		No.	Time	Info	
50	0.994715	80 → 1161 [ACK] Seq=1 Ack=29777 Win=61320 Len=0		45	0.855878	1161 → 80 [ACK] Seq=29777 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
51	1.039820	80 → 1161 [ACK] Seq=1 Ack=31237 Win=62780 Len=0		46	0.856802	1161 → 80 [ACK] Seq=31237 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
52	1.117097	80 → 1161 [ACK] Seq=1 Ack=33589 Win=62780 Len=0		47	0.857683	1161 → 80 [PSH, ACK] Seq=32697 Ack=1 Win=17520 Len=892 [TCP segment of a reassembled PDU]	
59	1.200421	80 → 1161 [ACK] Seq=1 Ack=35049 Win=62780 Len=0		53	1.117333	1161 → 80 [ACK] Seq=33589 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
60	1.265026	80 → 1161 [ACK] Seq=1 Ack=37969 Win=62780 Len=0		54	1.118133	1161 → 80 [ACK] Seq=35049 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
61	1.362074	80 → 1161 [ACK] Seq=1 Ack=40889 Win=62780 Len=0				Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
62	1.389886	80 → 1161 [ACK] Seq=1 Ack=41781 Win=62780 Len=0				Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
69	1.488313	80 → 1161 [ACK] Seq=1 Ack=44701 Win=62780 Len=0				Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
70	1.582980	80 → 1161 [ACK] Seq=1 Ack=47621 Win=62780 Len=0		57	1.120902	1161 → 80 [ACK] Seq=39429 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
71	1.661513	80 → 1161 [ACK] Seq=1 Ack=49973 Win=62780 Len=0		58	1.121891	1161 → 80 [PSH, ACK] Seq=40889 Ack=1 Win=17520 Len=892 [TCP segment of a reassembled PDU]	
78	1.758227	80 → 1161 [ACK] Seq=1 Ack=52893 Win=62780 Len=0		63	1.398110	1161 → 80 [ACK] Seq=41781 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
79	1.860063	80 → 1161 [ACK] Seq=1 Ack=55813 Win=62780 Len=0		64	1.398824	1161 → 80 [ACK] Seq=43241 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
80	1.930880	80 → 1161 [ACK] Seq=1 Ack=58165 Win=62780 Len=0		65	1.391683	1161 → 80 [ACK] Seq=44701 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
87	2.029069	80 → 1161 [ACK] Seq=1 Ack=61085 Win=62780 Len=0		66	1.392594	1161 → 80 [ACK] Seq=46161 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
88	2.126682	80 → 1161 [ACK] Seq=1 Ack=64005 Win=62780 Len=0		67	1.393390	1161 → 80 [ACK] Seq=47621 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
89	2.203195	80 → 1161 [ACK] Seq=1 Ack=66357 Win=62780 Len=0		68	1.394202	1161 → 80 [PSH, ACK] Seq=49081 Ack=1 Win=17520 Len=892 [TCP segment of a reassembled PDU]	
96	2.311413	80 → 1161 [ACK] Seq=1 Ack=69277 Win=62780 Len=0		72	1.661734	1161 → 80 [ACK] Seq=49973 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
97	2.404228	80 → 1161 [ACK] Seq=1 Ack=72197 Win=62780 Len=0		73	1.662474	1161 → 80 [ACK] Seq=51433 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
98	2.476576	80 → 1161 [ACK] Seq=1 Ack=74549 Win=62780 Len=0		74	1.663315	1161 → 80 [ACK] Seq=52893 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	
				75	1.664198	1161 → 80 [ACK] Seq=54353 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]	

All ACKs

All segments

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

*Throughput = amount of data transmitted / time incurred*

Throughput = 164090 / (5,455830 - 0,026477) = 30222.753 B/s

We found amount of data transmitted in packet-detail pane (see following screenshot)

[Frame: 185, payload: 155576-156467 (892 bytes)]
[Frame: 192, payload: 156468-157927 (1460 bytes)]
[Frame: 193, payload: 157928-159387 (1460 bytes)]
[Frame: 194, payload: 159388-160847 (1460 bytes)]
[Frame: 195, payload: 160848-162307 (1460 bytes)]
[Frame: 196, payload: 162308-163767 (1460 bytes)]
[Frame: 197, payload: 163768-164039 (272 bytes)]
[Frame: 199, payload: 164040-164089 (50 bytes)]
[Segment count: 122]
[Reassembled TCP length: 164090]
[Reassembled TCP Data: 504f5354202f6574686572655616c2d6c6162732f6c6162332d312d7265706c792e68746d...]

By dividing the amount of data transmitted by the time passed for transmission we calculated the throughput. Note that we calculated the time from which the first TCP segment after triple-handshake was sent which is packet No 4.

## TASK A:

According to our observations from the questions, we got to understand that there is a three-way handshake needed in order to establish a connection between client and the server. This way, the client sends a SYN packet including it's sequence number asking the server for permission to establish the connection. Then the server responds to the client with a SYNACK packet including a sequence number which means that it confirms the request. At last, the client ACKs the server's response and starts sending the segments. The next

steps will be taken recursively, which means that for every packet sent by client, the server ACKs the packets and determines the next packet by telling it's sequence number.

EstimatedRTT is the estimated round trip time that client determines itself to know the time it takes for the server to ACK the packets. In case, there is no signal from the server in the EstimatedRTT, then the client assumes that the packet is lost and will opt to send the packet again, this is how clients interpret packet losses.

## TASK B:

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 if and where TCP's *slow start* phase begins and ends, as well as if 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.

By studying outstanding data as we calculate in the following table we observe that the amount of data that is not being acknowledged never grows enough to reach congestion avoidance. The maximum outstanding data in this connection is **8192** and for the first Data being sent is **565** bytes. By knowing that **(1)** the receiver side buffer size, `rwnd`, is 62780 byte (taken from Wireshark) and **(2)** outstanding data must be the minimum of receiver buffer size and congestion window size at the sender (taken from textbook and showed in the following line):

$$\text{LastByteSent} - \text{LastByteAcked} \leq \min\{\text{cwnd}, \text{rwnd}\}$$

we can say that `cwnd` is **8192** bytes in maximum and **566** bytes in minimum. On the other hand we never see any rapid growth of this number which means that it's not possible to see the end of TCP slow-start in this case. It has never pushed data aggressively enough to reach congestion state.

Type	No.	No.	ACKed seq.	Outstanding data
Data	4	1		565
Data	5	566		2025
ACK	6		566	1460
Data	7	2026		2920
Data	8	3486		4380



ACK	9		2026	2920
Data	10	4946		4380
Data	11	6406		5840
ACK	12		3486	4380
Data	13	7866		5527
ACK	14		4096	4917
ACK	15		6006	3007
ACK	16		7866	1147
ACK	17		9013	0
Data	18	9013		1460
Data	19	10473		2920
Data	20	11933		4380
Data	21	13393		5840
Data	22	14853		7300
Data	23	16313		8192
ACK	24		10473	6732
ACK	25		11933	5272

The difference here is that the idealized TCP is supposed to be more aggressive at sending data to the receiver because it needs to reach the maximum allowed rate of bytes in that specific TCP connection. Idealized TCP follows different algorithms and starts slowly but trying to increase congestion window size exponentially. Here we don't see any aggressive attempt to send data. This might be because of the size of the file which means that TCP connection ends before it shifts to congestion avoidance phase.

- 14. Explain the relationship between (i) the congestion window (cwnd), (ii) the receiver advertised window (rwnd), (iii) the number of unacknowledged bytes, and (iv) the effective window at the sender (i.e., the window effectively limiting the data transmission).**

**congestion window(cwnd):** is an additional variable in TCP congestion-control mechanism which determines the amount of unacknowledged data at a sender.

Suppose Host **A** is sending Host **B** a large file over a TCP connection. The number of **unacknowledged bytes** that **A** sends cannot exceed the size of the receive buffer.

**receiver advertised window(rwnd):** is a variable that provides the flow-control service. It is typically the free spare space in the receiving buffer. How? Host **B** tells Host **A** how much spare room it has in the connection buffer by placing its current value of *rwnd* in the receive window field of every segment it sends to **A**.

**effective window:** The size of the recent advertised window depends on each ACK sent by the server to the client.

$$LastByteSent - LastByteAcked \leq \min\{cwnd, rwnd\}$$

- 15. Is it generally possible to find the congestion window size (cwnd) and how it changes with time, from the captured trace files? If so, please explain how. If not, please explain when and when not. Motivate your answer and give examples.**

According **RFC5681**, CONGESTION WINDOW (*cwnd*): A TCP state variable that limits the amount of data a TCP can send. At any given time, a TCP **MUST NOT** send data with a sequence number higher than the sum of the highest acknowledged sequence number and the minimum of *cwnd* and *rwnd*.

So it is not generally possible to find the **cwnd** size as it is not advertised, However it is possible to find the **rwnd** size, because it is advertised.

#### TASK C:

- 16. What is the throughput of each of the connections in bps (bits per second)? What is the total bandwidth of the host on which the clients are running? Discuss the TCP fairness for this case.**

Connection	Total transferred bytes	Duration (in seconds)	RTT (in milliseconds)
1	165095720	521	12
2	165842766	521	12
3	165458792	514	12
4	163235772	512	12

**Connection1:**  $165095720 / 521 * 8 = 2535059,04$  b/s

**Connection2:**  $165842766 / 521 * 8 = 2546529,99$  b/s

**Connection3:**  $165458792 / 514 * 8 = 2575234,12$  b/s

**Connection4:**  $163235772 / 512 * 8 = 2550558,94$  b/s

**Total bandwidth = 10207382,09 bits**

In this case, we have almost similar durations and identical RTTs.

By looking at the throughput in each connection we do not see very large differences so we assume that it is a fair TCP connection.

**17. What is the throughput of each of the connections in bps (bits per second)? What is the total bandwidth of the host on which the clients are running? Discuss the TCP fairness for this case**

Connection	Total transferred bytes	Duration (in seconds)	RTT (in milliseconds)
1	261319130	90	13
2	175995832	90	35
3	151894552	90	68
4	140388568	90	73
5	108610702	90	49
6	70644690	90	33
7	65744938	90	135
8	43212876	90	326
9	39222524	90	322

**Connection 1:**  $261319130 / 90 * 8 = 23,228,367.1111 \text{ b/s}$

**Connection 2:**  $175995832 / 90 * 8 = 15,644,073.9556 \text{ b/s}$

**Connection 3:**  $151894552 / 90 * 8 = 13,501,737.9556 \text{ b/s}$

**Connection 4:**  $140388568 / 90 * 8 = 12,478,983.8222 \text{ b/s}$

**Connection 5:**  $108610702 / 90 * 8 = 9,654,284.6222 \text{ b/s}$

**Connection 6:**  $70644690 / 90 * 8 = 6,279,528 \text{ b/s}$

**Connection 7:**  $65744938 / 90 * 8 = 5,843,994.4889 \text{ b/s}$

**Connection 8:**  $43212876 / 90 * 8 = 3,841,144.5334 \text{ b/s}$

**Connection 9:**  $39222524 / 90 * 8 = 3,486,446.5778 \text{ b/s}$

**Total bandwidth is : 93958561.06682**

In this case, we have almost identical durations but different RTTs.

On contrary to the previous question, we see a very large difference in throughput between these 9 connections which means that TCP is not fair in this case and bandwidth has been unfairly shared between these 9 clients (running in the same host).

**18. Discuss the TCP fairness for this case. How does it differ from the previous cases, and how is it affected by the use of BitTorrent?**

Connection	Total transferred bytes	Duration (in seconds)	RTT (in milliseconds)
1	108851134	58	40
2	90435681	58	36
3	57971584	53	100
4	32000012	29	68
5	32557334	35	31
6	27199361	31	33
7	26329578	31	122
8	38834490	56	146
9	23571761	35	74
10	36252962	55	66

Group1:

**Connection 1:**  $108851134 / 58 * 8 = 15013949.5172 \text{ b/s}$

**Connection 2:**  $90435681 / 58 * 8 = 12473887.0345 \text{ b/s}$

**Connection 6:**  $27199361 / 31 * 8 = 7019189.93548 \text{ b/s}$

**Connection 5:**  $32557334 / 35 * 8 = 7441676.34286 \text{ b/s}$

Group2:

**Connection 4:**  $32000012 / 29 * 8 = 8827589.51724 \text{ b/s}$

**Connection 9:**  $23571761 / 35 * 8 = 5387831.08571 \text{ b/s}$

**Connection 10:**  $36252962 / 55 * 8 = 5273158.10909 \text{ b/s}$



Group3:

**Connection 3:**  $57971584 / 53 * 8 = 8750427.77358 \text{ b/s}$

**Connection 7:**  $26329578 / 31 * 8 = 6794729.80645 \text{ b/s}$

**Connection 8:**  $38834490 / 56 * 8 = 5547784.28571 \text{ b/s}$

We divided this connection into three groups to discuss TCP fairness. In each group there are connections with roughly the same RTT.

In this case, unlike the previous two questions, we have different durations and RTTs. However when we compare connections throughput with each other in the same group, we see that they are more likely to be **unfair**. For example in group 3, connections 3, 7 and 8 there are some differences between throughput. The same for connections 4, 9 and 10, and same for connections 1, 2, 5 and 6.

However these clients benefit from using BitTorrent and help each other with downloading the same file from the server. Each client may share those parts of data it downloaded to another client and it causes a more fair connection.