TDDE35 lab 3 (Wireshark Lab: TCP)

1: The first packet is packet number 200 and the last one is packet number 202.
2: Client IP: 192.168.1.102, TCP port number: 1161
3: Destination IP: 128.119.245.12, TCP port number: 80
4: The sequence number of the TCP SYN segment in this case is 0. In the 'flags' section of the initial SYN segment, we can see that the 'syn' field is set to 1, which means that this is a SYN segment.
5: The sequence number of the SYNACK segment is 0. The ACKnowledgement field is set to 1. The value of 1 was found by incrementing the initial sequence number of the TCP SYN segment. It is a SYNACK message because both the ACKnowledgement field and SYN field have the value of 1.
6: Sequence number 4 for the segment containing the POST message.
7: The first six segments are 4, 5, 7, 8, 10, 11. The respective sequence numbers are

Number 4: 1
Number 5: 566
Number 7: 2026
Number 8: 3486
Number 10: 4946
Number 11: 6406

Ack to -> segment: 6->4, 9->5, 12->7, 14->8, 15->10, 16->11. (ack = seq+len)

	Sent	RTT	Received (ACK)
Seg 1	0.0265	0.0274	0.0539
Seg 2	0.0417	0.0356	0.0773
Seg 3	0.0540	0.0701	0.1241
Seg 4	0.0547	0.1144	0.1691
Seg 5	0.0774	0.1399	0.2173
Seg 6	0.0782	0.1896	0.2678

EstimatedRTT = (1-a) * EstimatedRTT + a * SampleRTT. Recommended value for a is 0.125, so we get <u>EstimatedRTT = 0.875 * EstimatedRTT + 0.125 * SampleRTT.</u>

(in seconds)
1: EstimatedRTT = RTT = 0.0274
2: EstimatedRTT = 0.875*0.0274 + 0.125*0.0356 = 0.0284
3: EstimatedRTT = 0.875*0.0284 + 0.125*0.0701 = 0.0336
4: EstimatedRTT = 0.875*0.0336 + 0.125*0.1144 = 0.0437
5: EstimatedRTT = 0.875*0.0437 + 0.125*0.1399 = 0.0557
6: EstimatedRTT = 0.875*0.0557 + 0.125*0.1896 = 0.0724

8: Length of the first six TCP segments: 565, 1460, 1460, 1460, 1460, 1460 (bytes).
9: Since gaia.cs.umass is the receiver, we can look at the [SYN,ACK] message, where the window is set to 5480 bytes. The last ACK before the HTTP response has window size 62780 bytes, and no sign of the sender being throttled is found.

10: We looked at the Sequence Numbers Graph (Stevens) and found that the sequence numbers in respect to time is growing, which means that we don't have any retransmitted segments in the trace file. If we would've had that, the sequence number would've dropped in size since we would've needed to "go back" and retransmit segments with lower sequence number than its neighbors.
11:

ACK[i] (refers to question 7)	ACK number	ACK delta
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
ACK 8	10473	1460

The difference between two neighboring ACK numbers is 1460 bytes, in other words the ACK values increases with 1460 every new ACK, so the receiver typically acknowledges 1460 bytes.

12:
The connection's throughput can be calculated by taking the delta of the amount of data sent divided by the transmission time during the whole connection. The total data is the amount of the last ACK – the amount of the first ACK --> 164 091-1 = 164 090. The total transmission the is the time delta between the last and first ACK --> 5,4558 – 0,0265 = 5,4293. Throughput is 164 090/5,4293 = 30,223 bytes/sec = 30 KB/s.

Task A paragraph
We could see in the first SYN message, in the Network layer field, that the IP address of our client computer and the server at gaia.cs.umass.edu are 192.168.1.102 and 128.119.245.12 respectively, and the port numbers for the client and the server are 1161 and 80 respectively. We can also, by checking the flags Syn and ACKnowledgement, deduce that the first three packets where intended for the TCP handshake. We found the sequence numbers for each segment in the Transport layer section, at field "Sequence number".

Estimating RTT is important since if the actual time to receive confirmation of something you've sent is much larger than the expected RTT, we know that it must be some kind of error in the path between server and client. If the network thinks that we have a packet loss, it will trigger congestion control, which reduces the queue time which has been built up due to packet loss. So estimating RTT can help keeping the queue of packets short and can alert the network that packet loss might be happening.

13: After looking at the graph, the TCP slow start begins at packet 1 and finishes at packet number 13 with sequence number 7866. We can see that on the graph since it has more of an exponential growth until sequence number 7866, and then it seems to send packets in groups of 6 somewhat linearly. This linear behavior of the graph is when it enters congestion avoidance. The ideal behavior would be to have this exponential-like graph all the way, by

not reaching the threshold value, but we seem to have reached that at the packet with the sequence number 7866.

14: Congestion window is how many packets that are sent simultaneously, and the receiver advertised window is the receiver telling how many packets at once it can take. The congestion window increases exponentially until the rwnd limit is reached. The highest allowed number of unacknowledged bytes is the minimum of two numbers, specifically either the buffer size of the sender, or the rwnd. The effective window is in turn the minimum of the rwnd value and the cwnd value.

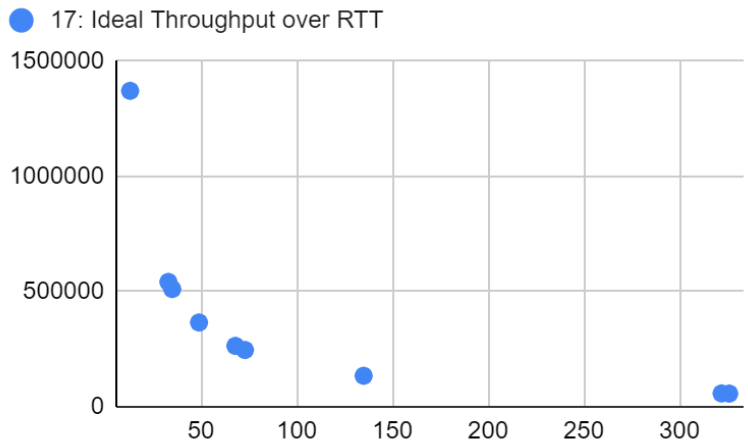
15: Yes, it is generally possible to find the cwnd value. For example in our Sequence number (Stevens) graph we see that it sends segments of 6 at the same time slot, which means that 6 TCP segments can fit in one cwnd.

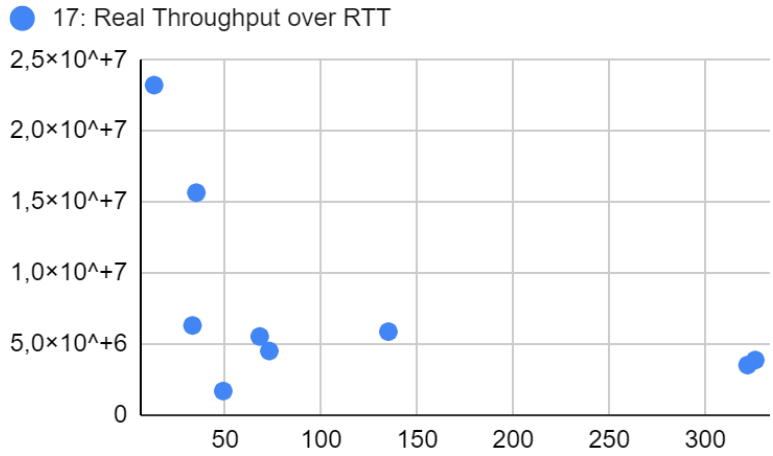
16:
Loss rate, assume 1%, MSS = 1460.

Connection	Real Throughput (bps)	Throughput (ideal)
1	2 535 074	1 484 333
2	2 546 530	1 484 333
3	2 575 234	1 484 333
4	2 550 559	1 484 333

The TCP fairness is good in this case since all the connection holds a similar ratio between the real and ideal throughput.
17:

Connection	Real Throughput(bps)	Throughput (ideal)
1	23 228 368	1 370 154
2	15 644 072	508 914
3	5 501 736	261 941
4	4 478 984	244 000
5	1 654 288	363 510
6	6 279 528	539 758
7	5 843 992	131 941
8	3 841 144	54 638
9	3 486 448	55 317

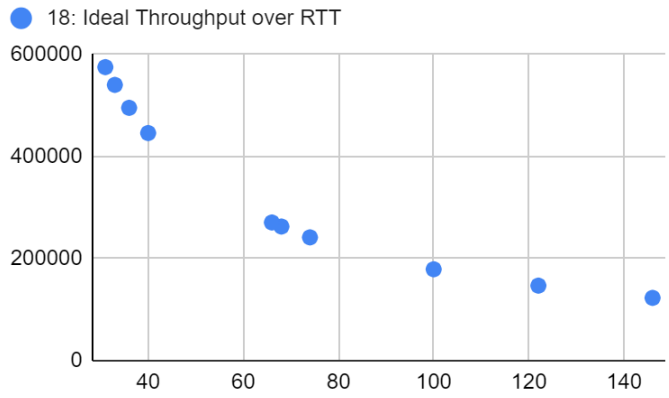


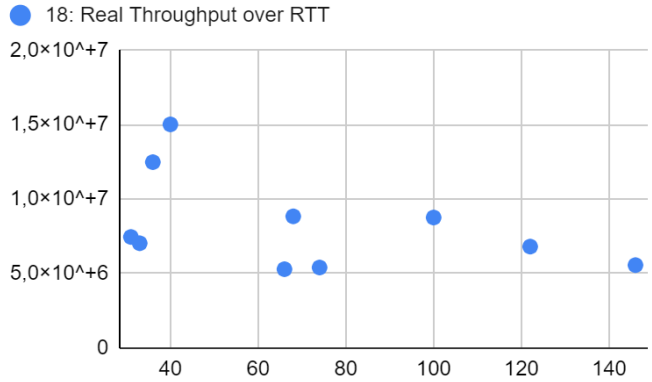


The goal of TCP fairness is if X TCP sessions share the same bottleneck link of bandwidth R each should have and average rate of X/R. As the trendlines shows most of the connection have a lower average then the trendline. And the comparison between the graphs show that it is a good TCP fairness.

18:

Connection	Real Throughput (bps)	Throughput (ideal)
1	15 013 949	445 300
2	12 473 887	494 777
3	8 750 427	178 120
4	8 827 589	261 941
5	7 441 676	574 580
6	7 019 189	539 757
7	6 794 729	146 000
8	5 547 784	122 000
9	5 387 831	240 702
10	5 273 158	269 878





As of TCP fairness in BitTorrents then the average through will vary a lot which shows in the graph, since BitTorrents works in such way the as person who is downloading you are then taking the shared bandwidth of all the seeders to that specific torrent. As we can see in the graph the TCP fairness is not good in this case through the comparison between the graphs.