TCP Protocol Lab 3

TDTS04

Samuel Lindgren samli627
Gustaf Svanerud gussv448
19/5 2016

Task A

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

First packet is number 4, last is number 199

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

ip address: 192.168.1.102 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 connections?**

ip address: 128.119.245.12 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?**

seq nr: 0, the tcp flag Syn is set to 1 which makes it 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?**

seq nr: 0, the value of the ack field is 1, both syn and ack flags are 1

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

Seq nr: 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 (see page 277 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 277 for all subsequent segments.**

ERTT1= (1-0.125)*0.027460000+0.125*0.027460000 = 0.02746
ERTT2= (1-0.125)*0.02748746+0.125*0.035557000 = 0.028472125

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

619, 1514, 1514, 1514, 1514, 1201

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?**

The window size value is 5840 in the first ack (nr 2). This is increased steadily up to 62780 which is max.
The sender is never throttled by the buffer space which we can see in the sent tcp segments.

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 because the seq nr are in increasing order and are never overlapping. If a segment was retransmitted then some seq nr would be after another in order of time but before in terms of the seq nr. I looked at the time sequence graph (stevens) for the ack statements.

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 285 in the text).**

The typical amount of acknowledged data is 1460. Ack nr 87 acknowledges tcp seg nr 82 and tcp seg nr 51 is never acked. So yes, it happens on several occasions that the receiver acknowledges every other segment.

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

The size of the tcp segment divided by the time it took the segment to be transferred is (164091-1)/(5.455830-0.02477)=30213.2548710565
So the throughput is 30213.2548710565 bytes/second

Reflections and observations:
The estimated RTT is based on the RTT and uses the equation seen in answer 7 to smooth out the RTT. This value is then used when determining if a packet needs to be resent because of packet loss. We never observed any retransmitted packets for this transaction as explained in answer 10.

When we look at the graph for RTT and the throughput for this connection we see that they are both steady, which is what we would expect for a connection with no retransmission of packets. This is also in line with our calculations in answer 7 and 12.

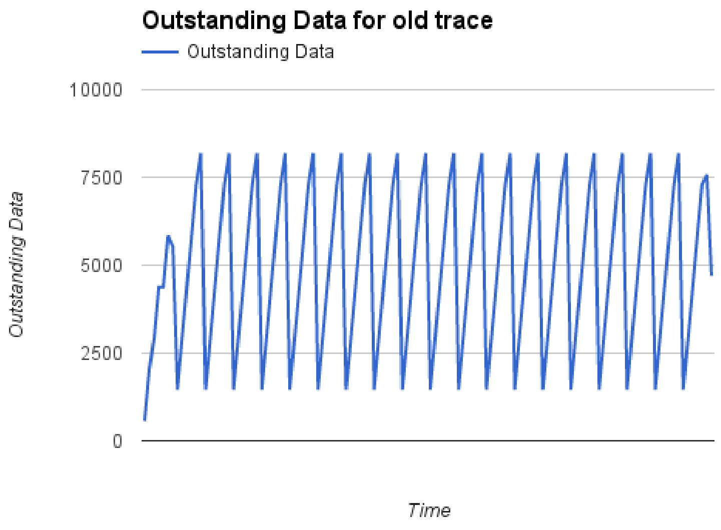
In answer 11, we note that the receiver sometimes only acknowledges every other packet but we a still not experiencing any retransmission of packet. This is because acknowledgments are cumulative and the receiver does not have to acknowledge every packet. In conclusion, the receiver can wait 2 packets and then acknowledge the first packet and then expect the next packet to be number 3.

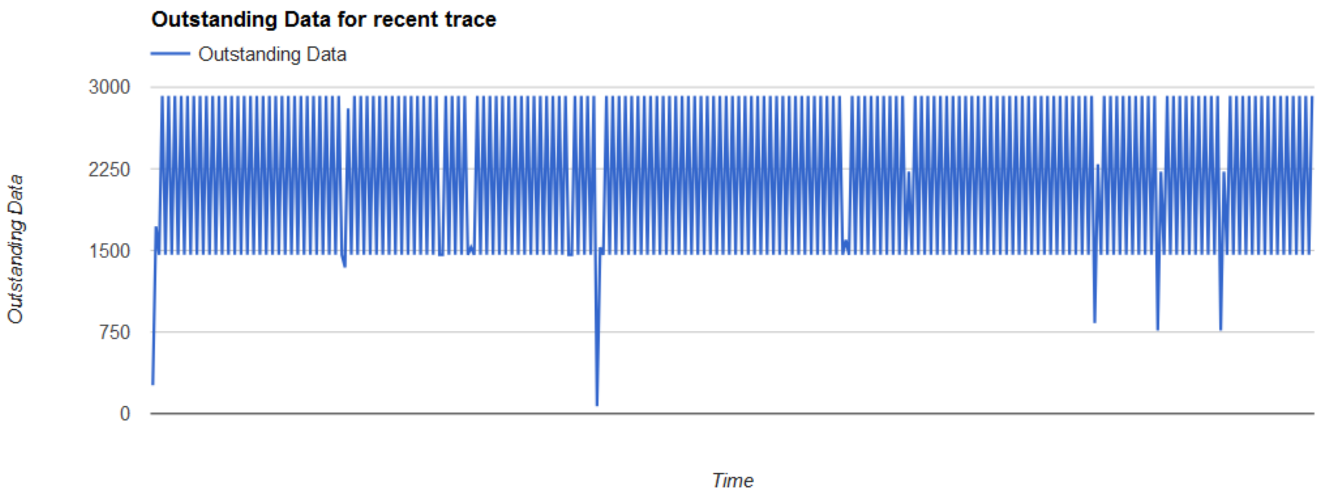
Task B

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 server (Figure 2a and Figure 2b). For each of the two traces, can you identify where TCP's *slow start* phase begins and ends, and where *congestion avoidance* takes**

over? If you can, explain how. If not, explain why not. To better identify these phases, you may need to find the number of unacknowledged packets (or bytes) at different times and plot the unacknowledged packets (y-axis) as a function of time (x-axis). Note that the number of unacknowledged packets at different times can be found by comparing the number of packets that have been sent with the number of packets that have been acknowledged. After plotting the number of unacknowledged packets versus time, comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the text.

Tcp slow start is initiated in the beginning of the transmission, when the first packet is sent in frame 4. The congestion window of the sender determines if and when congestion avoidance will happen. The Congestion window is the number of bytes that can be unacknowledged (outstanding) at any time during the exchange. Wireshark does not capture the size of the sender's congestion window because it is not sent in the header.





The congestion window is increased for every acknowledged segment. If the window size exceeds the receiver's SSthreshold or a timeout occurs, the window size will be reset and congestion avoidance will take over.

The lower bound of the congestion window is partially controlled by the Maximum segment size (MSS) which the receiver sets in the beginning of the connection. If we look at the graph of outstanding data for the old trace, we see that it fluxuates between 1460 and 8192 bytes. This means that the SSthreshold must be greater than 8192.

It is not possible to identify the end of the slow start face and the beginning of congestion avoidance because the sender is not sending the data at a fast enough rate so that the congestion avoidance can be triggered.

This is the case for both of the wireshark traces because of the similarities between the stevens graphs and the outstanding data graphs.

The idealized behavior of TCP is that the sender increases the amount of data until the congestion window is too big and the congestion avoidance takes over. The case we looked at here did not send data at a fast enough pace to trigger the congestion avoidance and there could have been data sent but the sender did not send it. In this case, the tcp slow start may have made this interaction inefficient for that reason.

14. **Explain the relationship between (i) the congestion window, (ii) the receiver advertised window, (iii) the number of unacknowledged bytes, and (iv) the effective window at the sender.**

The number of unacknowledged bytes is what we in the question above called outstanding data. It is the data which has been sent but not acknowledged by the receiver.

The congestion window is the amount of unacknowledged data that is set by the sender. This window increases during the connection.
The receiver advertised window is the amount of unacknowledged data that the receiver can handle.

The effective window is the lesser value of the congestion window and the advertised window.

15. **Is it generally possible to find the congestion window size (i.e. *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. Your answer may also benefit from trying to describe and discuss your answer in the context of the two prior questions, for example.**

The congestion window size is not transmitted in the tcp header and therefor can not be read from the captured trace files. It is possible to try to estimate the size of the congestion window. The acknowledgments sent by the receiver affects the size of the congestion window, so by observing when acks are sent by the receiver we can estimate when the congestion window changes in size. A problem with this method is that the trace file may have observed an ack that did not reach the sender. As observed in question 13, it can be hard to determine if the sender is in slow start or congestion avoidance if the sender is not sending data fast enough. Say that the trace file observes an ack to the sender and the the sender does not get the ack, but because the sender does not send data fast enough, it will not change the sending pattern and we can not determine if the sender got the ack or not and by extension if the congestion window size changed or not.

Task C

To be able to use the equation (1.22*MSS)/(RTT*sqrt(L)) provided, we need to estimate some variables for the connections.

The amount of round trips that a connection made can be estimated by dividing the duration of the connection by the RTT. The maximum segment size can be estimated by dividing the total amount of transferred data by the amount of round trips. We now have an estimated MSS which we can use in the equation.

The loss rate (or bit error rate) that the different cases experience is unknown to us. When calculating the throughputs for the three cases, we discover that the loss rate has a significant effect on the throughput of a connection. But we need to know more information about the connections to be able to make some estimations on the different loss rates that the three cases could have. The loss rate of a connection depends alot on what type of connection it is (Wireless wifi, LAN, fiber etc.) and the distance between the sender and the receiver. We do not know any of these things when it comes to the three cases so we have to set a reasonable loss rate which feels appropriate. After some investigation[1] we set the loss rate to 1*10^-8.

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.**

If we use the estimations mentioned above to calculate the throughput for the connections in case 1 we get these values.

Case 1

Connection	**RTT (ms)**	**Throughput in Gbit/s**
1	12	30.93
2	12	31.07
3	12	31.42
4	12	31.12
	Total bandwidth	**124.53**

The total bandwidth is calculated by adding all the connections throughput together.

All of the connections have similar duration, RTT and total transferred bits. This makes the Gbps throughput for the connections very similar. All the connections are on the same client and they are downloading from the same host. The fairness for this case is good since the connections share the total bandwidth in an expected manner according to our fairness equation.

1 http://www.netcraftsmen.com/tcp-performance-and-the-mathis-equation/
https://www.slac.stanford.edu/comp/net/wan-mon/thru-vs-loss.html

17. **Another case to consider is downloading the same file from different mirror servers around the world. 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.**

Case 2

Connection	RTT (ms)	Throughput in Gbit/s
1	13	283,39
2	35	190,86
3	68	164,72
4	73	152,24
5	49	117,78
6	33	76,61
7	135	71,30
8	326	46,86
9	322	42,53
	Total bandwidth	**1146,29**

The total bandwidth is calculated by adding all the connections throughput together.

In this case all the connections have the same duration but different RTT and total transferred bits.
The connections with a smaller RTT transferred more total bits during the 90 second than the connections with a higher RTT, which is expected. We observe that the RTT and loss rate effects the final throughput a lot, according to the fairness equation. Because we have assumed the same loss rate for all the connections, it is expected that the RTT plays a big part in the calculations of the throughput for a connection.

The throughput for the connections with low RTT is a lot more than the throughput for the connections with high RTT. If the low RTT connections experience a packet loss, they will discover it faster than the high RTT connections and will be able to adjust the congestion window faster. This could be seen as a unfair consequence of the equation.

18. **The final case to consider is a BitTorrent download from multiple peers. Similar to the previous cases, the details of each of the connections is presented in the following table. This time only ten of the connections are presented.**

Discuss the TCP fairness for this case.

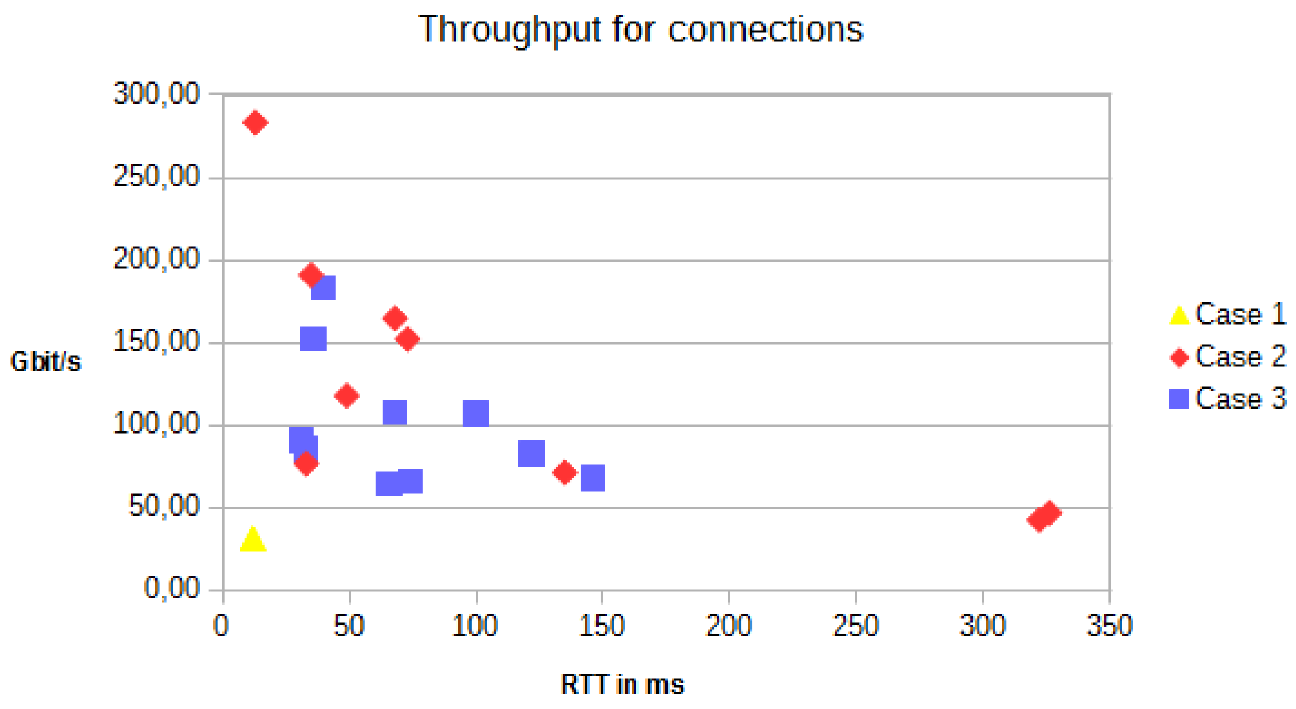
Case 3

Connection	RTT (ms)	Throughput in Gbit/s
1	40	183,17
2	36	152,18
3	100	106,76
4	68	107,70
5	31	90,79
6	33	85,63
7	122	82,90
8	146	67,68
9	74	65,73
10	66	64,33
	Total bandwidth	**Unknown**

The total bandwidth of this connection is unknown since we are only given ten of the connections. We do not know how many more connections that are used for the download and therefor can not calculate the total bandwidth.

As seen in in the previous case, the connections with low RTT have more throughput but in this case the difference in the highest and lowest RTT is not as big as in the previous case and the difference in throughput is not as big either. For that reason, this case could be called more fair than the previous but as mentioned above, we don't see all the connections so it is hard to make that statement.

This scatter graph shows the correlation between RTT and throughput for the different connections in the three cases. Note that the four connections in case 1 are very close to each other since their throughput and RTT is very similar


Throughput for connections
Gbit/s
RTT in ms
Case 1
Case 2
Case 3