**Assignment #02**

1. Consider the following behavior of a TCP connection (using the congestion control algorithm we learned in class).

At time 0, a TCP sender initiates a connection. As soon as the connection is established, the TCP sender will begin sending data. The MSS is 1KB and RTT is 100 ms.

1) Assuming the connection does not lose any data or experience any timeouts, at what time will the sender’s congestion window be 16KB? (Assuming *threshold* is 32MSS)

My answer:

At the beginning ,the cwnd will be exponential increased for one time in 1 RTT , so the time to be 16KB will be

4 RTT = 400ms

Right after the sender’s congestion window has reached a size of 16KB, a timeout occurs. After the timeout is detected, the sender continues sending more data over the established connection.

2) Assuming no additional packets loss or timeouts, how long (since the observed timeout) will it take for the congestion window to build to size 14KB?

My answer:

After the time out, the congestion will be set to 1 KB , and then it continues to be exponential increased util the threshold, now the threshold becomes:

Threshold=16KB/2=8KB

After the threshold ,the cwnd goes in additive time,

, so the time cost to build the size 14KB is :

Time = (log2(threshold) + (14 - threshold)/1)RTT

= 900ms

3) While its congestion window is at 14KB, the sender receives triple duplicate acknowledgements for the same sequence number. How long after receiving the third duplicate acknowledgement will it take for the sender’s congestion window to be at least 9KB again?

My answer:

If the increasing time doesn’t interrupted by other congestion ,we can get a least time ,

At the time receiving triple duplicate ack, the threshold becomes

Threshold= 14 / 2=7 KB

And the cwnd = cwnd / 2 = 7 KB

So it cost (9KB-7KB)/1KB = 2 RTT= 200ms

1. Consider a scenario with two hosts, Alice and Bob. A web server running on Alice is trying to send data to a browser on Bob. For each TCP connection, Alice’s TCP stack maintains a send buffer of 512 bytes and Bob’s TCP stack maintains a receive buffer of 1024 bytes. For simplicity, assume TCP sequence numbers began at 0 in this problem.

1) Bob’s stack received up to byte 560 in order from Alice, although its browser has only read up to the first 60 bytes. What will be the ***ACK#***and ***rcvr window size***in the TCP headers that Bob next sends to Alice?

My answer:

Obviously ,the ack number will be 561;

The rcvr window size is 1024 - ( 560 - 60) = 524

2) Later in the same connection, Alice’s congestion window is set to 1 MSS = 536 bytes and the advertised flow-control window from Bob is 560 bytes. The last ***ACK#*** that Alice received from Bob is byte 700, and the last byte that Alice sends to Bob is byte 900.

A) What is the smallest byte number that Bob will not accept?

My answer:

According to the question ,the next excepted byte of Bob is 700,

The last sent byte is 900 , and the congestion window is 536 bytes ,

So the largest byte number that Bob will accept is

536 - (900 - 699) + 900 = 1235

So the smallest byte number that Bob will not accept is 1236.

B) Assuming that Alice doesn’t receive any more ACKs and her window does not change, what is the greatest byte number that Alice can send?

My answer:

According to the lastByteSent - lastByteAcked<=AdvertisedWindow

And Alice’s last Acked byte number is 699 , and the AdvertisedWindow

is 560 , and the sender’ last send byte is 900, but the sender’s buffer is only 324 bytes remains,

so the greatest byte number Alice can send is 900 + (512 - 201)= 1211

C) Again assuming that Alice doesn’t receive any additional ACKs, how many more bytes can the web server running on Alice write to its network socket before blocking?

My answer:

According to the second question ,we can know the greatest byte number Alice can send is 1211 , and the last byte sent by Alice is 900 , as a result:

1211 - 900 = 311 Bytes