1. A. To connect to the welcome server the IP or hostname of the server is required along with the hostname.

B. The way to ID a connection socket is using a 4-tuple which includes the server IP address, server, port number, client IP address, and client port number.

C. To ID the socket use the IP and port number of the client.

2. 100Mb/s \* .1s / 8 bits/byte = 1,250,000 In binary this is 100110001001011010000 which is 21 bits. So the sequence window is 21 bits. The sequence number then 1,250,000 \* 10 (because the window size was the amount transferable in .1 seconds) \* 60 s for the timeout window. This yields 750,000,000 bytes which is 101100101101000001011110000000 which is 30 bits.

3. The largest 16 bit number is 65,535. Converting from bytes to bits that’s 524,280 bits. That’s the max that can be transmitted in 50ms, so multiply by 20 to get seconds. The answer is then 10,485,600b/s

4. C:\Users\Admin\Downloads\Untitled Diagram (2).png

This solution will work, but it equalizes slower than ½ because cutting it by ½ will bring the one getting too much bandwidth into line with the connection not getting enough bandwidth much faster. Decreasing by ¼ isn’t affecting the bandwidth rates as much, making them equalize slower.

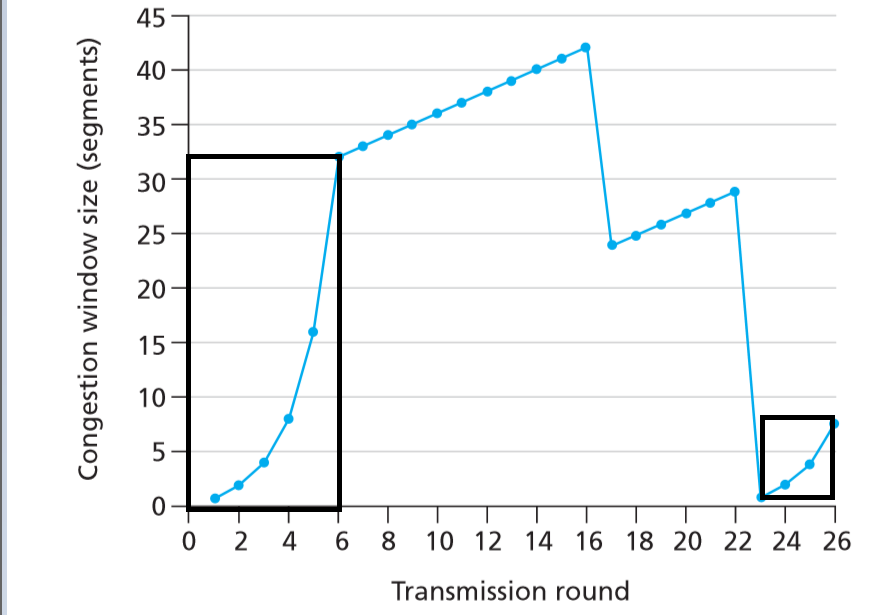
b.

C:\Users\Admin\Downloads\Untitled Diagram (3).png

Because they’re increasing and decreasing by the same amount they will never make any progress towards fairness. It will look like the above.

5. TCP will use N + 1 as acknowledgements are cumulative. Correct segments that are received out of the correct order are not individually acked like SR. Because of this only the window size plus one more for the number to indicate the next sequence starting needs to be kept. So in this regard at least TCP behaves more like GBN than SR, though TCP is also capable of correctly handling out of order segments to a degree which is more like SR, but implementation of that does not affect sequence number size.

6.

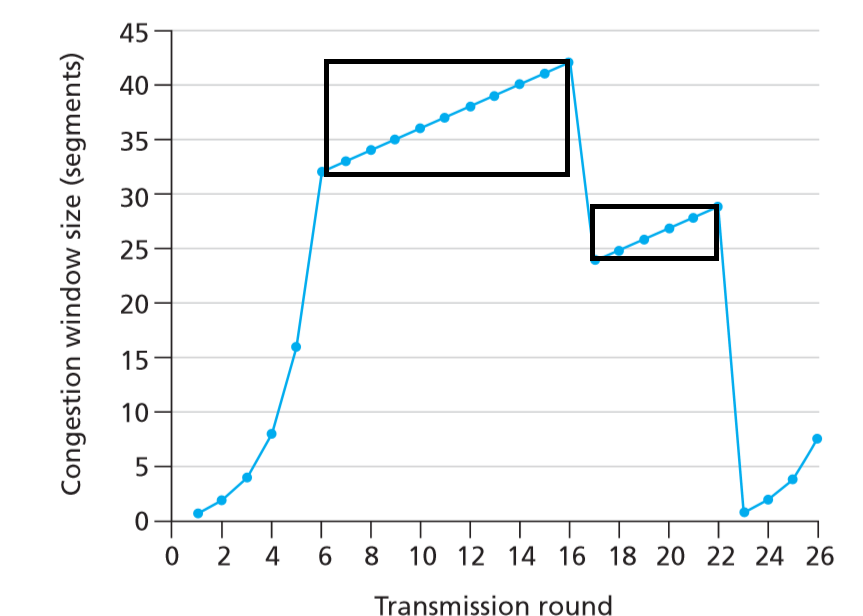


[0,6] and [23, 26]

0 all the way to six show a slow start section at the beginning where speeds ramp up slowly, then near the end ramp up extremely fast (exponentially even). Then it levels off after 6 so the slow start interval is over.

The packet loss at 22 does the same thing dropping speeds all the way back down to 0, so slow start begins again which can be seen increasing in the same fashion as 0 to 6.

b.



The speeds increase additively (linearly) when there is no traffic. So the sections of even increase with a uniform slope are where congestion avoidance is operating. This is [6,16] and [17, 22] because 16 to 17 suffered a packet loss, and the rest is slow start time. (Except for the packet loss at 22)

c. When a timeout occurs the speeds drop all the way to one. If it’s a triple duplicate ack the speed is just reduced not reset, so the loss at 16 is a triple duplicate ack.

d. This time it reset all the way to one instead of being reduced, so at 22 it’s a timeout.

e. SSThresh is the value when congestion avoidance begins. In this case it occurs at 6, and has a value of about 32.

f. SSTresh will be whatever the speed was of the round before. When the packet loss happened at 16 17 had it’s speed set to 21, cutting it in half from 42. So at 18 this will be reflected and SSThresh will be 21.

g. Here even though a timeout occurred at 22 setting the speed to 1, the SSTresh is still only cut in half. So that means until the slow start reaches 26/2 (13) the SSTresh will remain at 13. So at round 24 it’s 13.

h.

* Round 1: speed 1, packet #s trans: 1
* Round 2: speed 2, packet #s trans: 2-3
* Round 3: speed 4, packet #s trans: 4-7
* Round 4: speed 8, packet #s trans: 8-15
* Round 5: speed 16, packet #s trans: 16-31
* Round 6: speed 32, packet #s trans: 32-63
* Round 7: speed 33, packet #s trans: 64-96 (70 is here)

1. If a trip dip ack occurs the speed will be reduced to 4. The SSThresh will also be cut in half from 13 to 8 (round up).

(Optional)

j. TCP Tahoe works for recovery. It will set the congestion windows size to 1 even after a triple dup ack. This means the slow start threshold still remains at 21 (after having from the packet loss at 16). The congestion window size will be 4.

k.

* Round 17: speed 1, packets trans: 1
* Round 18: speed 2, packets trans: 2
* Round 19: speed 4, packets trans: 4
* Round 20: speed 8, packets trans: 8
* Round 21: speed 16, packets trans: 16
* Round 22: speed 21, packets trans: 21

Total is 52 packets

Extra Credit:

The congestion window size from the algorithm shown in class is incremented by a constant sum when an ack arrives. If the current window size = X only X/2 acks are going to be sent back using delayed acks. This means the function: cwnd = cwnd + MSS (MSS/cwnd) used to increase the rate will only be occurring half as often.