2. Solution:

Suppose the IP addresses of the hosts A, B, and C are a, b, c, respectively. (Note that a, b, c are distinct.)

To host A: Source port =80, source IP address = b, dest port = 26145, dest IP address = a

To host C, left process: Source port =80, source IP address = b, dest port = 7532, dest IP address = c

To host C, right process: Source port =80, source IP address = b, dest port = 26145, dest IP address = c

15. Solution:

It takes 12 microseconds (or 0.012 milliseconds) to send a packet, as 1500\*8/109=12

microseconds. In order for the sender to be busy 98 percent of the time, we must have

util = 0.98 = (0.012n) / 30.012

Thus, n approximately 2451 packets.

28. Solution:

Since the link capacity is only 100 Mbps, so host A’s sending rate can be at most 100Mbps. Still, host A sends data into the receive buffer faster than Host B can remove data from the buffer. The receive buffer fills up at a rate of roughly 40Mbps. When the buffer is full, Host B signals to Host A to stop sending data by setting RcvWindow = 0. Host A then stops sending until it receives a TCP segment with RcvWindow > 0. Host A will thus repeatedly stop and start sending as a function of the RcvWindow values it receives from Host B. On average, the long-term rate at which Host A sends data to Host B as part of this connection is no more than 60Mbps.

40. Solution:

a. TCP slowstart is operating in the intervals [1,6] and [23,26]

b. TCP congestion avoidance is operating in the intervals [6,16] and [17,22]

c. After the 16th transmission round, packet loss is recognized by a triple duplicate ACK. If there was a timeout, the congestion window size would have dropped to 1.

d. After the 22nd transmission round, segment loss is detected due to timeout, and hence the congestion window size is set to 1.

e. The threshold is initially 32, since it is at this window size that slow start stops and congestion avoidance begins.

f. The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 16, the congestion windows size is 42. Hence the threshold is 21 during the 18th transmission round.

g. The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 22, the congestion windows size is 29. Hence the threshold is 14 (taking lower floor of 14.5) during the 24th transmission round.

h. During the 1st transmission round, packet 1 is sent; packet 2-3 are sent in the 2nd transmission round; packets 4-7 are sent in the 3rd transmission round; packets 8-15 are sent in the 4th transmission round; packets 16-31 are sent in the 5th transmission round; packets 32-63 are sent in the 6th transmission round; packets 64 – 96 are sent in the 7th transmission round. Thus packet 70 is sent in the 7th transmission round.

i. The threshold will be set to half the current value of the congestion window (8) when the loss occurred and congestion window will be set to the new threshold value + 3 MSS . Thus the new values of the threshold and window will be 4 and 7 respectively.

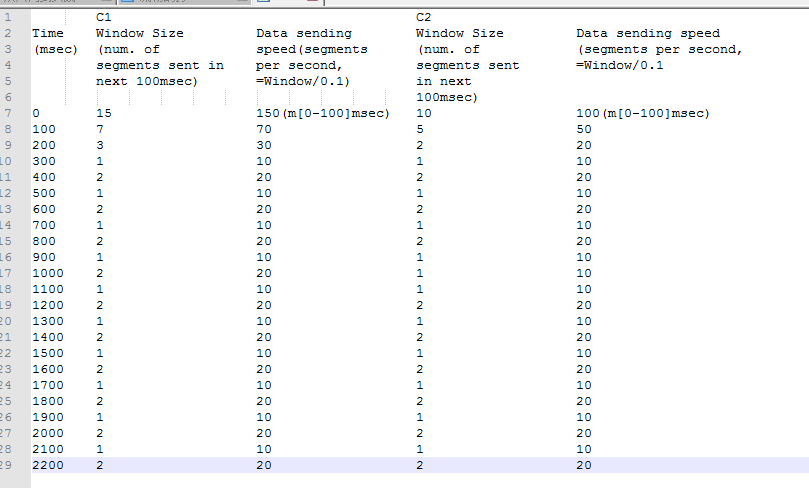
j. threshold is 21, and congestion window size is 1.

k. round 17, 1 packet; round 18, 2 packets; round 19, 4 packets; round 20, 8 packets; round 21, 16 packets; round 22, 21 packets. So, the total number is 52.

(congestion)

51. Solution:

a. Compute their window sizes over time in the following table. Both C1 and C2 have the same window size 2 after 2200msec.



b. Yes, this is due to the AIMD algorithm of TCP and that both connections have the same RTT.

c. Yes, this can be seen clearly from the above table. Their max window size is 2.

d. No, this synchronization won’t help to improve link utilization, as these two connections act as a single connection oscillating between min and max window size. Thus, the link is not fully utilized (assume this link has no buffer). One possible way to break the synchronization is to add a finite buffer to the link and randomly drop packets in the buffer before buffer overflow. This will cause different connections cut their window sizes at different times. There are many AQM (Active Queue Management) techniques to do that, such as RED (Random Early Detect), PI (Proportional and Integral AQM), AVQ (Adaptive Virtual Queue), and REM (Random Exponential Marking), etc.