**P27**. Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 126. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 80 and 40 bytes of data, respectively. In the first segment, the sequence number is 127, the source port number is 302, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A.

* a. In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number?

Seq Num: 207

Source port: 302

Destination Port: 80

* b. If the first segment arrives before the second segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number, the source port number, and the destination port number?

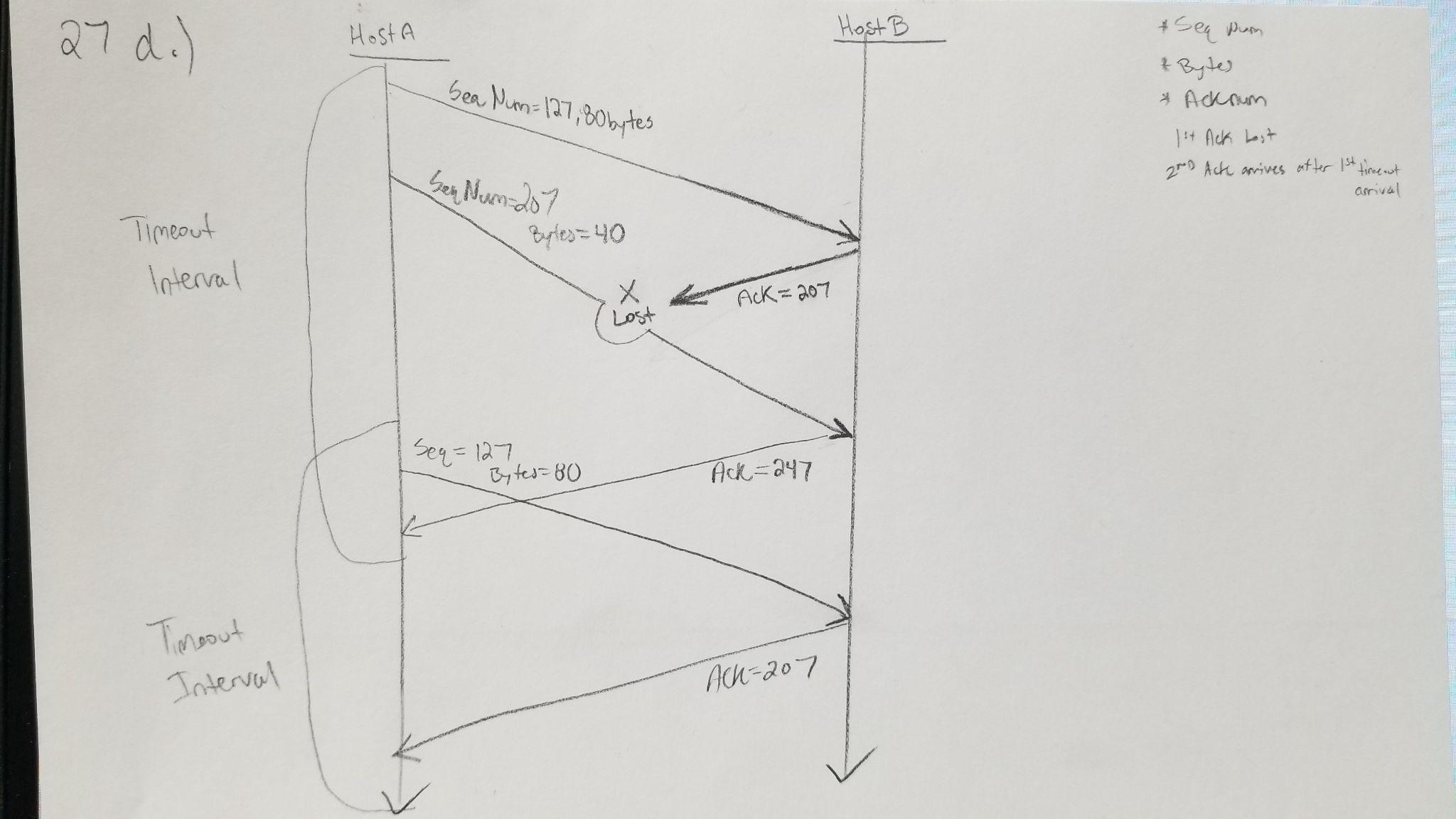
ACK num: 207

Source Port: 80

Destination Port: 302

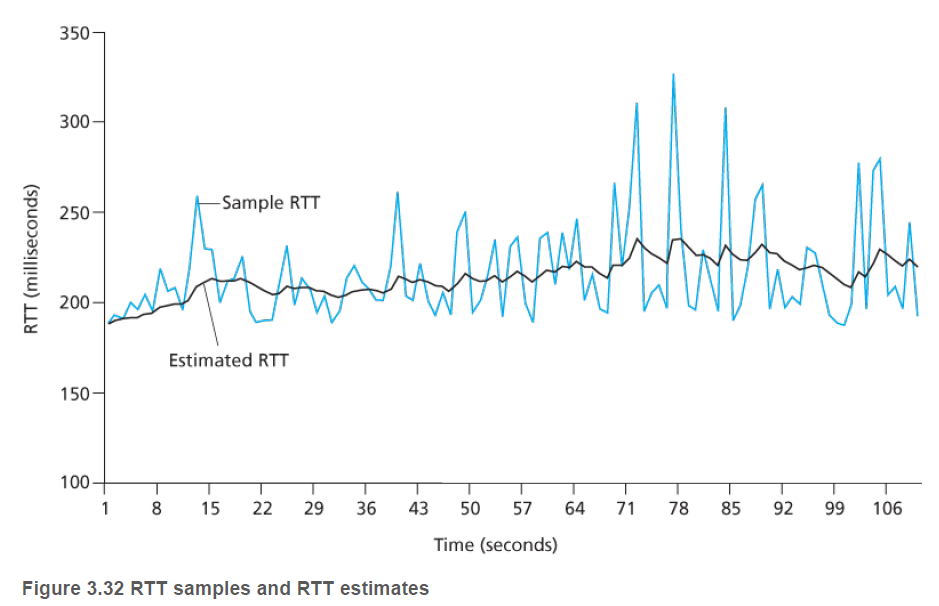
* c. If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number?

ACK num: 127. This is sent back to let Host A know that Host B is still waiting for bytes from 127 and later.

* d. Suppose the two segments sent by A arrive in order at B. The first acknowledgment is lost and the second acknowledgment arrives after the first timeout interval. Draw a timing diagram, showing these segments and all other segments and acknowledgments sent. (Assume there is no additional packet loss.) For each segment in your figure, provide the sequence number and the number of bytes of data; for each acknowledgment that you add, provide the acknowledgment number.

**P28**. Host A and B are directly connected with a 100 Mbps link. There is one TCP connection between the two hosts, and Host A is sending to Host B an enormous file over this connection. Host A can send its application data into its TCP socket at a rate as high as 120 Mbps but Host B can read out of its TCP receive buffer at a maximum rate of 50 Mbps. Describe the effect of TCP flow control.

Host A has a sending capacity that is over the capacity over the Link and Host B’s capability to receive. Once Host B’s receives enough data to fill the Receive Buffer. It will set its Receive Window(rwnd) value to 0. This blocks A from sending any more data to B and overflowing the buffer. TCP requires Host A to send 1 byte segments while Host B’s rwnd is zero. When there is room in the buffer after Host B resets rwnd to a nonzero value Host A can begin sending data again.

**P31.** Suppose that the five measured SampleRTT values (see Section 3.5.3 ) are 106 ms, 120ms, 140 ms, 90 ms, and 115 ms. Compute the EstimatedRTT after each of these SampleRTT values is obtained, using a value α=0.125 of and assuming that the value of EstimatedRTT was 100 ms just before the first of these five samples were obtained. Compute also the DevRTT after each sample is obtained, assuming a value of β=0.25 and assuming the value of DevRTT was 5 ms just before the first of these five samples was obtained. Last, compute the TCP TimeoutInterval after each of these samples is obtained.

DevRTT = (1-𝛃)\*DevRTT+𝛃\*|SampleRTT-EstimatedRTT|

EstimatedRTT = (1-**ɑ**)\*EstimatedRTT + **ɑ\***SampleRTT

TimeoutInterval = EstimatedRTT + 4\*DevRTT

**Start Conditions: EstimatedRTTstart = 100ms, DevRTTstart = 5ms, 𝛃 = 0.25, ɑ = 0.125**

Order of samples: 106 ms, 120ms, 140 ms, 90 ms, and 115 ms.

After 106ms:

DevRTT106 = (1-.25)\*5ms + .25\*(|106ms-100ms|) = 5.25ms

EstimatedRTT106 = (1-.125)\*100ms + .125\*106ms = 100.75ms

TimeoutInterval = 100.75ms + 4\*5.25ms = 121.75ms

After 120ms:

DevRTT120 = (1-.25)\*5.25ms + .25\*(|120ms-100.75ms|) = 8.75ms

EstimatedRTT120 = (1-.125)\*100.75ms + .125\*120ms = 103.16ms

TimeoutInterval = 103.2ms + 4\*8.75ms =138.16 ms

After 140ms:

DevRTT140 = (1-.25)\*8.75ms + .25\*(|140ms-103.16ms|) = 15.77ms

EstimatedRTT140 = (1-.125)\*103.16ms + .125\*140ms = 107.76ms

TimeoutInterval = 107.76ms + 4\*15.77ms = 170.84ms

After 90ms:

DevRTT90 = (1-.25)\*15.77 + .25\*(|90-107.76ms|) = 16.2675ms

EstimatedRTT90 = (1-.125)\*107.76 + .125\*90ms = 105.5ms

TimeoutInterval = 105.5ms + 4\*16.27ms = 170.58ms

After 115ms:

DevRTT115 = (1-.25)\*16.27ms + .25\*(|115ms-105.5ms|) = 14.58ms

EstimatedRTT115 = (1-.125)\*105.5ms + .125\*115ms = 106.69ms

TimeoutInterval = 106.69ms + 4\*14.58ms = 165.01ms

**P37.** Compare GBN, SR, and TCP (no delayed ACK). Assume that the timeout values for all three protocols are sufficiently long such that 5 consecutive data segments and their corresponding ACKs can be received (if not lost in the channel) by the receiving host (Host B) and the sending host (Host A) respectively. Suppose Host A sends 5 data segments to Host B, and the 2nd segment (sent from A) is lost. In the end, all 5 data segments have been correctly received by Host B.

<https://www2.tkn.tu-berlin.de/teaching/rn/animations/gbn_sr/>

* a. How many segments has Host A sent in total and how many ACKs has Host B sent in total? What are their sequence numbers? Answer this question for all three protocols.

Go-Back-N(GBN):

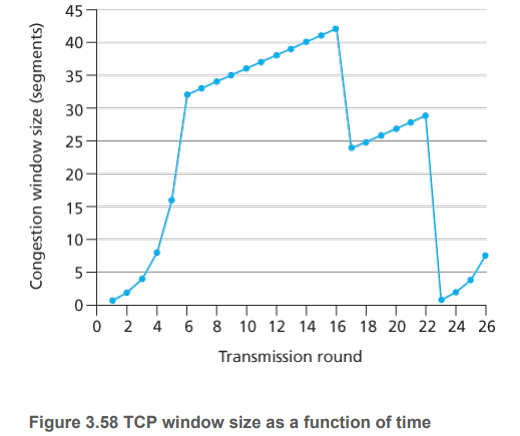
* Host A sends 5 segments at the same time. (1,2,3,4,5)
* Then Packet 2 is Lost.
* 4 ACKs are sent back for 1,3,4,5.
* Host A waits for timeout of 2nd segment.
* Host A retransmits from the lost segment to the end (2,3,4,5). 4 segments.
* Host B Sends back 4 ACKs for 2,3,4,5.
* A sends 9 segments, B sends 8 ACKs.

Selective Repeat(SR):

* Host A sends out 5 segments at the same time. (1,2,3,4,5)
* Then packet 2 is lost.
* Host B sends back 4 ACKs for (1,3,4,5).
* Host A waits for timeout of 2nd segment.
* Host A retransmits the lost segment (2).
* Host B sends back 1 ACK for segment 2.
* A sends 6 segments,B sends 5 ACKs.

TCP:

* Host A sends 5 segments at the same time. (1,2,3,4,5)
* Then Packet 2 is Lost.
* 4 ACKs are sent back for the expected sequence number 2 of missing segment.
* Host A retransmits lost data without waiting for timeout interval.
  + Sends 2nd segment.
* Host B sends back 1 ACK of 6(expected next value).
* A sends 6 segments, B sends 5 ACKs.
* b. If the timeout values for all three protocol are much longer than 5 RTT, then which protocol successfully delivers all five data segments in shortest time interval?

* TCP. TCP utilizes fast retransmit instead of waiting for timeout of the lost packet.
* 

**P40.** Consider Figure 3.58 . Assuming TCP Reno is the protocol experiencing the behavior shown above, answer the following questions. In all cases, you should provide a short discussion justifying your answer.

* a. Identify the intervals of time when TCP slow start is operating.
  + 1-6 and 23-26. Increasing start from 0.
* b. Identify the intervals of time when TCP congestion avoidance is operating.
  + 6-16 and 17-22. High number of segments.
* c. After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
  + Triple Duplicate ACK. If there was a timeout the congestion window size would be set to 1.
* d. After the 22nd transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
  + Timeout. Congestion window was set to 1.
* e. What is the initial value of ssthresh at the first transmission round?
  + 32. ssthresh is the point where slow start stops and congestion avoidance starts.
* f. What is the value of ssthresh at the 18th transmission round?
  + 21. When packet loss is detected(in round 16), the congestion window is halved.
* g. What is the value of ssthresh at the 24th transmission round?
  + 14. When packet loss is detected(22nd round), the congestion window is halved from 29 to 14.5 ( floor(14.5) = 14.)
* h. During what transmission round is the 70th segment sent?
  + Round 1: packet 1 sent (cwnd = 1)
  + Round 2: packets 2-3 sent (cwnd = 2)
  + Round 3: packets 4-7 sent (cwnd = 4)
  + Round 4: packets 8-15 sent (cwnd = 8)
  + Round 5: packets 16-31 sent (cwnd = 16)
  + Round 6: packets 32 - 63 sent (cwnd = 32)
  + Round 7: packets 64 - 96 sent (cwnd = 33)
  + The 70th segment is in round 7.
* i. Assuming a packet loss is detected after the 26th round by the receipt of a triple duplicate ACK, what will be the values of the congestion window size and of ssthresh ?
  + ssthresh would be at half of what it was in round 26(cwnd = 8). Therefore it would be size 4.
  + The congestion window would be ssthresh + 3MSS. Cwnd = 7.
* j. Suppose TCP Tahoe is used (instead of TCP Reno), and assume that triple duplicate ACKs are received at the 16th round. What are the ssthresh and the congestion window size at the 19th round?
  + ssthresh is half congestion window. ssthresh =21
  + Tahoe reset cwnd to 1. cwnd = 1
* k. Again suppose TCP Tahoe is used, and there is a timeout event at 22nd round. How many packets have been sent out from 17th round till 22nd round, inclusive?
  + Tahoe resets cwnd to 1. Then doubles each round until it reaches the threshold.
  + Round 17: 1 packet
  + Round 18: 2 packets
  + Round 19: 4 packets
  + Round 20: 8 packets
  + Round 21: 16 packets
  + Round 22: 21 packets
  + Total = 52 packets
* **P41.** Refer to Figure 3.55 , which illustrates the convergence of TCP’s AIMD algorithm. Suppose that instead of a multiplicative decrease, TCP decreased the window size by a constant amount. Would the resulting AIAD algorithm converge to an equal share algorithm? Justify your answer using a diagram similar to Figure 3.55 .

