Problem 1

Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 326. 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 327, the source port number is 40200, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A. Fill in the blanks for questions (a) – (c) directly; work out the diagram in the box for question (d).

(a) In the second segment sent from Host A to B, the sequence number is 407, source port number is 40200, and destination port number is 80.

(b) If the first segment arrives before the second segment, in the acknowledgment of the first arriving segment, the ACK number is 407, the source port number is 80, and the destination port number is 40200.

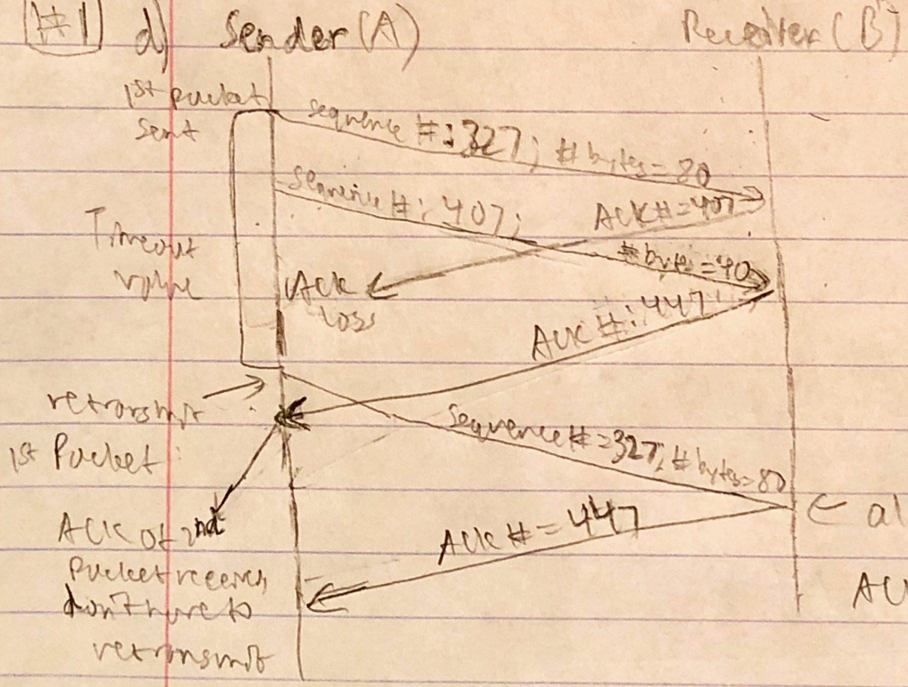
(c) If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, the ACK number is 327.

(d) Suppose the two segments sent by A arrive in order at B. The first acknowledgment is lost and the second acknowledgment arrives after A's timeout intervals for both the first and the second packets. Draw a timing diagram in the box below, showing these segments and acknowledgments until A receives all the acknowledgments of re-transmitted packets. Assume no additional packet loss. For each segment in your diagram, provide the sequence number and the number of bytes of data; for each acknowledgment that you add, provide the ACK number.

d) Diagram is on the next attached page.

CS 118 HW #4

$\boxed{\#1}$ d)



← already received 446 bytes in order

ACK returns highest in order # of bytes

Problem 2

One of the three functions of a sliding window scheme is the orderly delivery of packets which arrive out of sequence. In Go-back-N, the receiver drops packets which arrives out of order. Assume the receiver sends an ACK for every packet it receives.

(a) In Go-back-N, what is the required buffer size (receiver's window size, RWS) at the receiver if sender's window size (SWS) = 23? What the answer will be in Selective Repeat?

(b) In sliding window with SWS = RWS = 5, the minimum required `SeqNumSize` (the number of available sequence numbers) is 10. Calculate the minimum required `SeqNumSize` for

(i) a sliding window scheme with SWS = 6 and RWS = 3

(ii) a Go-back-N scheme with SWS = 6

(a) In Go Back N, a receiver buffer size of 1 is acceptable and the required size. This is because the receiver does not buffer any out of order packets recieved, and instead drops any packets delivered out of order. Only the next expected packet number is accepted, meaning that it is buffered briefly (to verify with the checksum, as well as hold if the application layer is busy) and then delivered competely to the application layer.
In Selective Repeat, a receiver buffer size of 23 is required. This is because here, all out of order packets are buffered until the expected in order packet/packets are delivered, after which these buffered packets can be properly delivered in order. In the worst case, the first packet could be delivered after the last 22 in the window, so these 22 packets would have to be buffered until this first packet is delivered. You would need a buffer size of 23, however, because in the buffer you must verify each packet using the checksum before delivering it to the application layer. Also, there is a possibility that the application layer is busy and cannot accept packages at a given moment, so you may need to hold packets in the buffer; you would need to do this for all of the packets in the window, thus a buffer size of 23.

(b) It appears that an acceptable formula for `SeqNumSize` is : `SeqNumSize` = SWS + RWS

(i) SWS + RWS = 9; Minimum required `SeqNumSize` is 9. This can be seen in the equation stated for the minimum sequence number size.

(ii) For a GBN scheme with SWS = 6, you need a minimum of 6+1 = 7 for the `SeqNumSize`. This is because you need one spot for the temporary buffer on the receiver side, and the other 6 for the sender's window.

Problem 3

Suppose that three measured SampleRTT values are 106 ms, 120 ms, and 150 ms. Compute the EstimatedRTT after each of these SampleRTT values is obtained, assuming that the value of EstimatedRTT was 100 ms just before the first of these three samples were obtained. Compute also the DevRTT after each sample is obtained, assuming the value of DevRTT was 5 ms just before the first of these three samples was obtained. Last, compute the TCP TimeoutInterval after each of these samples is obtained. Round your calculation results to two decimals (e.g., 123.45).

$\alpha = 0.125, \beta = 0.25$
$EstimatedRTT$ =(1-α)∗EstimatedRTT +α*SampleRTT
DevRTT = $(1-\beta)$∗DevRTT+β∗| $SampleRTT - EstimatedRTT$ |
$TimeoutInterval = EstimatedRTT + 4 * DevRTT$

After SampleRTT = 106 ms is obtained:
EstimatedRTT $= (1-\alpha) * 100 + \alpha * 106 = 100.75ms$
DevRTT $= (1-\beta) * 5 + \beta * \mid 106 - 100.75 \mid = 5.06ms$
TimeoutInterval = 100.75 + 4*(5.06) = 121 ms

After SampleRTT = 120 ms is obtained;
EstimatedRTT $= (1-\alpha) * 100.75 + \alpha * 120 = 103.16ms$
DevRTT $= (1-\beta) * 5.06 + \beta * \mid 120 - 103.16 \mid = 8.01ms$
TimeoutInterval = 103.16 + 4*(8.01) = 135.19 ms

After SampleRTT = 150 ms is obtained:
EstimatedRTT $= (1-\alpha) * 103.16$+$\alpha * 150 = 109.01ms$
$DevRTT$ =(1-β) $* 8.01 + \beta * \mid 150 - 109.01 \mid = 16.25ms$
TimeoutInterval = 174.02 ms

Problem 4

Compare Go-Back-N, Selective Repeat, and TCP (no delayed ACK). Assume that timeout values for all three protocols are sufficiently long, such that 10 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 10 data segments to Host B, and the 6th segment (sent from A) is lost. In the end, all 10 data segments have been correctly received by Host B.

(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.

(b) If the timeout values for all three protocols are very long (e.g., longer than several RTTs), then which protocol successfully delivers all 10 data segments in shortest time interval?

(a) GBN: First 10 segments (1-10) are sent at first. Then when the 6th packet is lost, packets 6-10 would be retransmitted. Host B sends the first 5 ACKs, and then the ACK for the 5th packet is send back for the ACKs for the packets after the 6th packet is dropped. Then the ACKs for 6-10 are sent when these are retransmitted: Thus the numbers are:
1 segment each for sequence s 1-5; 2 segments each for sequence s 6-10; 1 ACK each for all sequence s 1-4, 6-10; ACK for Sequence 5 is repeated 5 times. Total of 15 segments, 14 ACKs sent.

SR: 11 segments sent; 10 ACKs total 1 Segment each for sequence s 1-5, 7-10; 2 segments for sequence 6 1 ACK for packets 1-5, 7-10, as well as the 1 ACK for packet with sequence 6 when this is retransmitted.

TCP: 11 Segments sent by A; 10 ACKs total from B The segments for sequence 1-5 are sent, and these return ACKs for sequence 2-6 (ACK of next expected packet). Then, after packet 6 is dropped, the packets with sequence 7-10 are each buffered by the sender, and each send back an ACK with the sequence of 6, the missing packet. When the sender sees three of these Duplicated ACKs with 6 in a row, it then automatically retransmits the packet with sequence 6. This then returns an ACK of 11 when it is finally delivered, because at this point these 10 packets will be in order.

(b) The TCP protocol will deliver all 10 data segments in the shortest time. It is easy to see that this protocol will be faster than GBN because it limits the number of packet retransmissions to only the necessary retransmission for packet 6 in this scenarios. Further, TCP works faster than selective repeat because of its fast retransmission functionality; it will retransmit the missing packet (sequence #6) after seeing 3 duplicate acks, which is much faster than waiting for the timeout in selective repeat.

Problem 5

As we have discussed in the class, a timer is a useful component in various protocol designs: because a communicating end cannot see what is going on either inside the network or at the other end, when needed it sets up an ”alarm”, and takes some action when the alarm goes off.

(a) Does HTTP (not considering the underlying TCP) use any timers? If so, please briefly describe how each is used. If not, please explain why it does not need one.

(b) Does DNS use any timers? If so, please briefly describe how each is used. If not, please explain why it does not need one.

(c) Does TCP use any timer? If so, please briefly describe how each is used. If not, please explain why it does not need one.

(d) Does UDP use any timer? If so, please briefly describe how each is used. If not, please explain why it does not need one.

(a) No. HTTP does not use any timers on its own. This is because the underlying TCP protocol makes the use of timers. Therefore, any timeout messages can be communicated from this lower layer, and HTTP itself does not need a timer.

(b) No. DNS does not use any timers. This is because it is based on UDP, the simplest transport layer protocol that does not provide any type of extra services. While this is true, there is a timer-like functionality in DNS that is responsible for maintaining and updating DNS caches, which removes entries if they have not been requested again in a certain amount of time. Once this cached entry is removed, that entry's IP must be retrieved from it's authoritative DNS server.

(c) Yes. TCP does use timers, specifically for a sender to measure and keep track of the acknowledgement messages you are expecting from a receiver. If the timer runs out before an expected acknowledgement returns, then the sender will retransmit the data and assume that the sent packet was lost.

(d) No. UDP does not use any timers. It is the simplest possible transport layer protocol, and offers no extra functionalities other than an attempt to send a packet from sender to receiver. Timers would not apply to UDP since it is an unreliable transport layer protocol, so the data is just transmitted by the sender and then completely ignored.