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).

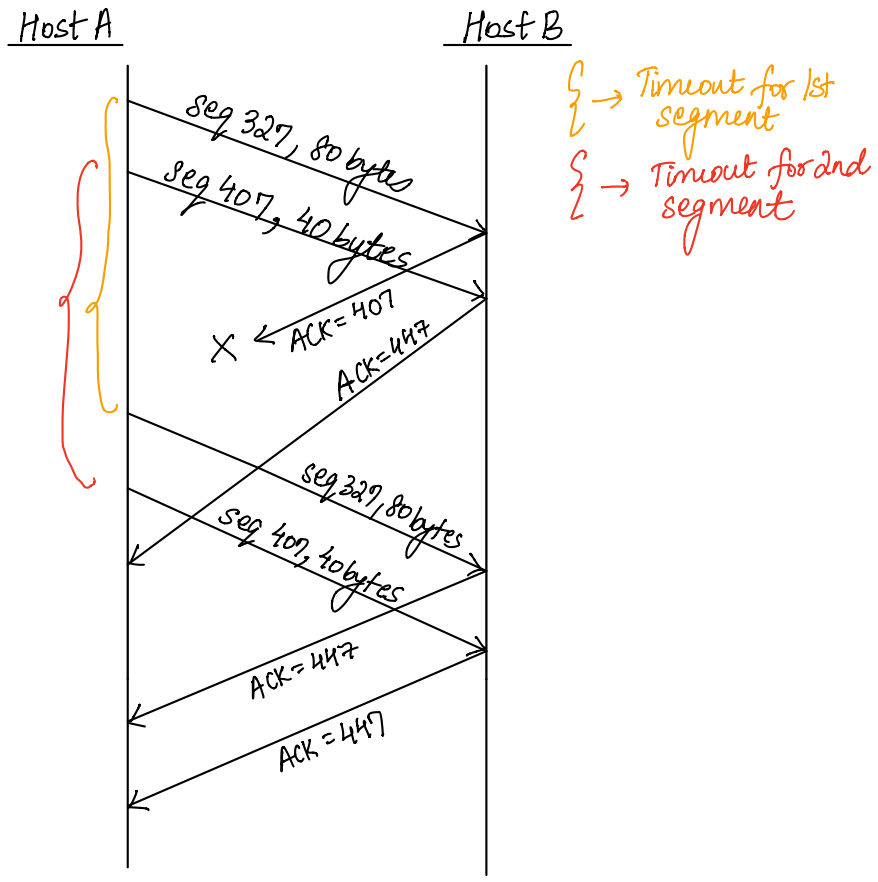
(a) In the second segment sent from Host A to B, the sequence number is 407, source port number is 40200, and destination port number is 80.

(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.

Write your solution to Problem 1 in this box



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

Write your solution to Problem 2 in this box

(a) In Go-back-N, the RWS will always be 1 irrespective of the sender's window size. This is because only the packet being expected can/will be in the buffer.

In the case of SR, the receiver window size must be N, or 23, the same as the SWS.

(b) In general, no. of sequence numbers $\geq$ SWS + RWS

(i) SWS = 6, RWS = 3, SeqNumSize = 6 + 3 = 9

(ii) Go-back-N $\rightarrow$ SWS = 6, RWS is always 1, so SeqNumSize = 6 + 1 = 7

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).

Write your solution to Problem 3 in this box

Assuming $\alpha = 0.125$, $\beta = 0.25$

Estimated RTT $= (1-\alpha)$ EstimatedRTT $+ (\alpha)$ SampleRTT

DevRTT $= (1-\beta)$ DevRTT $+ (\beta)|$SampleRTT $-$ EstimatedRTT$|$

TCP Timeout Interval = EstimatedRTT + 4* DevRTT

① SampleRTT = 106ms

- EstimatedRTT $= (0.875)100 + (0.125)(106) =$ **100.75ms**
- DevRTT $= (0.75)5 + 0.25|106 - 100.75| = 5.0625\text{ms} \approx$ **5.06ms**
- Timeout interval $= 100.75 + 4(5.06) \approx$ **121ms**

② SampleRTT = 120ms

- EstimatedRTT $= (0.875)100.75 + 0.125(120) = 103.15625\text{ms} \approx$ **103.16ms**
- DevRTT $= (0.75)(5.0625) + 0.25|120 - 103.16| = 8.006875 \approx$ **8ms**
- Timeout Interval $= 103.16\text{ms} + 4(8\text{ms}) =$ **135.16ms**

③ SampleRTT = 150ms

- EstimatedRTT $= (0.875)(103.16) + 0.125(150) = 109.015 \approx$ **109.02 ms**
- DevRTT $= (0.75)(8) + 0.25|150 - 109.02| = 16.245\text{ms} \approx$ **16.25ms**
- Timeout Interval $= 109.02\text{ms} + 4(16.25\text{ms}) =$ **174.02ms**

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?

Write your solution to Problem 4 in this box

(a) Selective Repeat:

A sends 11 segments: It first sends the 10 consecutive segments, and then retransmits segment 6 when it does not receive an ACK for 6 the first time.

B sends 10 ACKs: Once for each of the 10 segments it receives
Sequence: ACK1, ACK2, ACK3, ACK4, ACK5, ACK7, ACK8, ACK9, ACK10, ACK6

Go-Back-N:

A sends 15 segments: First it sends 1 through 10, then resends 6-10 on loss of 6th segment. ∴ Total = 10+5 = 15

B sends 14 ACKs: ACK1, ACK2, ACK3, ACK4, ACK5, ACK5, ACK5, ACK5, ACK6, ACK7, ACK8, ACK9, ACK10

TCP:

A sends 11 segments: First sends 1 through 10, and then retransmits 6 on failure/loss.

B sends 10 ACKs: ACK2, ACK3, ACK4, ACK5, ACK6, ACK6, ACK6, ACK6, ACK6, ACK11

(b) Unlike Go-Back-N and SR which wait for timeout to occur, TCP uses fast retransmission which results in the shortest time interval.

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.

Write your solution to Problem 5 in this box

(a) HTTP does not use any timers. This is because HTTP (app layer) uses TCP (Transport layer) to deliver packets, and TCP internally uses a timer and acknowledgements (ACKs) to track packet loss.

(b) Although DNS typically uses UDP which lacks a timer, it uses a timer to handle cases where a client fails to receive a response for a query within a certain duration. In this case, once a "timeout" occurs, the retransmission algorithm implemented within DNS resends the desired query. DNS also uses timers in its caching resolvers, which indicate the expiration of DNS records within the cache.

(c) TCP uses a transmission timer. When a segment is transmitted/sent, TCP starts a timer, and then stops the timer when an acknowledgement/ACK is received from the receiver. If the timeout occurs due to expiration of the timer for that segment, the segment is retransmitted.

(d) UDP does not use any timers. This is largely because of the nature of UDP, which provides no guarantees about timing or reliable data packet transfer. Because of this, UDP can usually transmit data faster since it lacks this reliability. Applications that use UDP must implement their own timers and retransmission triggers.