Jason Lai
204995126
Dis 1A

CS 118
April 30, 2020

Homework 4

1 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 _____, source port number is _____, and destination port number is _____.

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

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

(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.

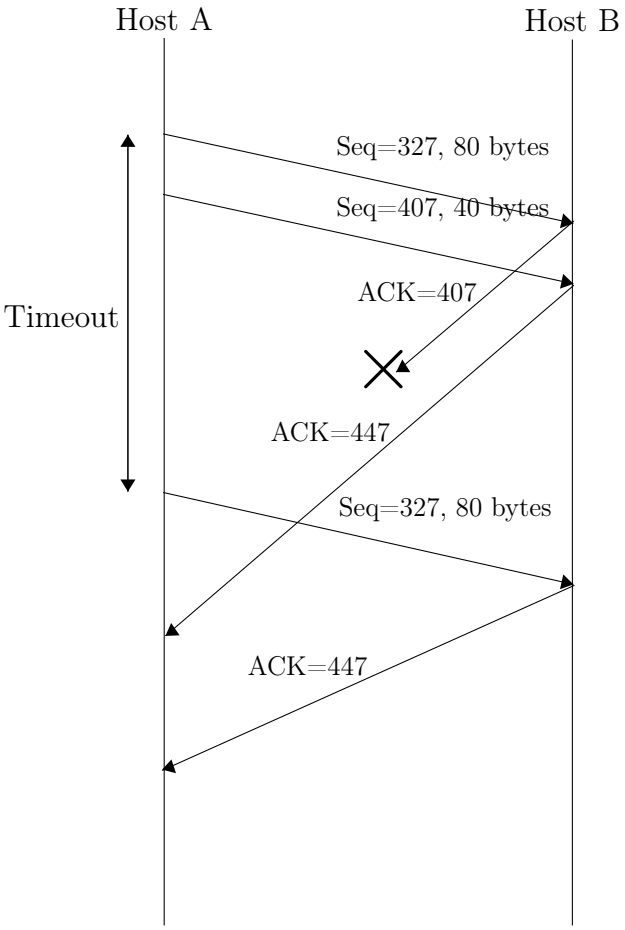
a) In the second segment sent from Host A to B, the sequence number is 407, source port number is 40200, and destination port number is 80.

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.

Part d) on next page.

d) *Piazza stated that the question statement was wrong; below is the timing diagram according to revised statement: "arrives right after A's timeout intervals for the first packet".



2 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, the required buffer size for the receiver's window size (RWS) is always 1, even if the sender's window size (SWS) is 23. In Selective Repeat, the required buffer size would be 23 if the sender's window size is 23.

b)

(i) Since the RWS = 3, the protocol we are using is Selective Repeat, so the minimum required SeqNumSize for a sliding window scheme with SWS = 6 and RWS = 3 would be 12. In Selective Repeat protocol, Window Size has to be less than or equal to half the SeqNumSize in order to avoid ambiguous sequence numbers as a result of retransmissions. Usually RWS = SWS in Selective Repeat, so it follows that the minimum SeqNumSize = SWS + RWS, but in this case SWS > RWS, so we multiply SWS by 2 to get the minimum SeqNumSize: minimum SeqNumSize $= 6 \times 2 = 12$.

(ii) The minimum required SeqNumSize for a Go-Back-N scheme with SWS = 6 would be 7. This is because if a receiver receives all 6 packets in a window with the sequence numbers 0 through 5 and its corresponding cumulative ACK for these packets is lost, then the sender will timeout and resend those packets. However, the receiver would have no way of knowing these packets are duplicates or a new window of packets, since they will have the same sequence numbers 0 through 5, so we need at least one more sequence number than the size of the window to avoid this issue.

3 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).

After 106 ms SampleRTT:
The EstimatedRTT after the 106 ms SampleRTT is **100.75 ms**.
The DevRTT after the 106 ms SampleRTT is **5.06 ms**.
The TimeoutInterval after the 106 ms SampleRTT is **121 ms**.

After 120 ms SampleRTT:
The EstimatedRTT after the 120 ms SampleRTT is **103.16 ms**.
The DevRTT after the 120 ms SampleRTT is **8.01 ms**.
The TimeoutInterval after the 120 ms SampleRTT is **135.19 ms**.

After 150 ms SampleRTT:
The EstimatedRTT after the 150 ms SampleRTT is **109.01 ms**.
The DevRTT after the 150 ms SampleRTT is **16.25 ms**.
The TimeoutInterval after the 150 ms SampleRTT is **174.02 ms**.

Work:

106 ms, 120 ms, 150 ms
Initial EstimatedRTT = 100 ms
Initial DevRTT = 5 ms
$\alpha = 0.125$
$\beta = 0.25$

$$\begin{aligned}\text{EstimatedRTT} &= (1-\alpha)\text{EstimatedRTT}_{\text{Initial}} + \alpha\text{SampleRTT} \\ &= (0.875)(100) + (0.125)(106) \\ &= 100.75 \text{ ms}\end{aligned}$$

$$\begin{aligned}\text{DevRTT} &= (1-\beta)\text{DevRTT}_{\text{Initial}} + \beta|\text{SampleRTT} - \text{EstimatedRTT}| \\ &= (0.75)(5) + (0.25)|106 - 100.75| \\ &= 5.0625 \text{ ms}\end{aligned}$$

$$\begin{aligned}\text{TimeoutInterval} &= \text{EstimatedRTT} + 4\text{DevRTT} \\ &= 100.75 + (4 \times 5.0625)\end{aligned}$$

$$= 121 \text{ ms}$$

$$\begin{aligned}
\text{EstimatedRTT} &= (1-\alpha)\text{EstimatedRTT}_{\text{Initial}} + \alpha\text{SampleRTT} \\
&= (0.875)(100.75) + (0.125)(120) \\
&= 103.15625 \text{ ms} \\
\text{DevRTT} &= (1-\beta)\text{DevRTT}_{\text{Initial}} + \beta|\text{SampleRTT} - \text{EstimatedRTT}| \\
&= (0.75)(5.0625) + (0.25)|120 - 103.15625| \\
&= 8.0078125 \text{ ms} \\
\text{TimeoutInterval} &= \text{EstimatedRTT} + 4\text{DevRTT} \\
&= 103.15625 + (4 \times 8.0078125) \\
&= 135.1875 \text{ ms}
\end{aligned}$$

$$\begin{aligned}
\text{EstimatedRTT} &= (1-\alpha)\text{EstimatedRTT}_{\text{Initial}} + \alpha\text{SampleRTT} \\
&= (0.875)(103.15625) + (0.125)(150) \\
&= 109.0117188 \text{ ms} \\
\text{DevRTT} &= (1-\beta)\text{DevRTT}_{\text{Initial}} + \beta|\text{SampleRTT} - \text{EstimatedRTT}| \\
&= (0.75)(8.0078125) + (0.25)|150 - 109.0117188| \\
&= 16.25292968 \text{ ms} \\
\text{TimeoutInterval} &= \text{EstimatedRTT} + 4\text{DevRTT} \\
&= 109.0117188 + (4 \times 16.25292968) \\
&= 174.0234375 \text{ ms}
\end{aligned}$$

4 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) For **Go-Back-N**, Host A has sent a total of 15 segments, since Host B will receive segments 1 through 5, but will discard the out-of-order segments 7 through 10, since segment 6 was lost; Host A will then have to resend segments 6 through 10. Host B has sent a total of 14 ACKs, since Host B will ACK segments 1 through 5, but since segments 7 through 8 are out of order, it will repeat ACK 5 for those received segments; once the resent segments are received, Host B will then send ACKs for segments 6 through 10. The sequence numbers for the ACKs sent in order are 1 2 3 4 5 5 5 5 5 6 7 8 9 10.

For **Selective Repeat**, Host A has sent a total of 11 segments, since Host B will receive segments 1 through 5, then buffer the out-of-order segments 7 through 10; the timer for segment 6 will then time out, and segment 6 will be resent. Host B has sent a total of 10 ACKs, since Host B will ACK segments 1 through 5, then 7 through 8, and finally 6. The sequence numbers for the ACKs sent in order are 1 2 3 4 5 7 8 9 10 6.

For **TCP**, Host A has sent a total of 11 segments, since Host B will receive segments 1 through 5, and then segments 7 through 10; then, after Host B sends multiple duplicate ACKs, Host A will resend segment 6. Host B has sent a total of 6 ACKs, since after receiving in-order segments 1 through 5, it will ACK them cumulatively, then for each out-of-order segment that arrives, it will send duplicate ACKs, so that's an additional 4 ACKs for segments 7 through 10. Host A will receive the duplicate ACKs, then resend segment 6, which Host B will receive and ACK. The sequence numbers for the ACKs sent in order are 6 6 6 6 6 11, since TCP will send ACKs with the sequence number of the next byte the receiving host expects.

b) The TCP protocol will successfully deliver all 10 segments in the shortest time interval,

since it uses fast retransmit, where the sending host does not need to wait for a timeout to resend a segment, as out-of-order packets will cause the receiving host to send duplicate ACKs, which in turn causes the sending host to resend the unACK'd segment with the smallest sequence number.

5 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) Yes, even if the underlying TCP is not considered, HTTP still does use timers in the form of a configurable timeout interval know as HTTP Keep-Alive, which is used for persistent HTTP connections. HTTP Keep-Alive is managed by the Web server implementing the Keep-Alive feature, which is different from TCP Keep-Alive and other TCP timers that are managed by the operating system. HTTP Keep-Alive is configured by setting the timeout parameter in the Keep-Alive header of an HTTP request; the timeout parameter can be set to the number of seconds that a persistent connection can be idle before the server usually drops the connection to the client; each time the server receives a request from the client, the connection is not considered idle and the timer resets.

b) Yes, DNS uses timers for the resource records of hostname-to-IP-address mappings that are cached in a system. These timers are known as the resource records' Time-To-Live, and specifies a time at which the resource record will expire and be removed from the cache if the record is not updated. In this regard, DNS uses timers to time when a hostname-to-IP-address mapping is deemed too outdated to keep in the cache.

c) Yes, TCP uses several timers including the single retransmission timer, the persistent timer, the keep-alive timer, the time wait timer, and the quiet timer. However, as noted in a TA response on Piazza, I will only talk about the single retransmission timer that is in the very basic TCP protocol. This single retransmission timer is used to wait on the an ACK for the oldest unACK'd segment that was sent to the receiving host. If the timer times out, the then the oldest unACK'd segment i.e. the unACK'd segment with the smallest sequence number will be retransmitted to the receiving host by the sender. However, since TCP uses

fast retransmit, wherein if there are out-of-order segments that come after a lost segment, the receiving host will send duplicate ACKs requesting for the segment with the expected sequence number. These duplicate ACKs will cause the sending host to immediately resend the lost segments, so the single retransmission timer may be cut short in these cases. Regardless, the TCP protocol uses several different timers.

d) No, UDP does not use any timers, because the protocol is "connectionless" and "best effort"; this means UDP allows packet loss and packet errors, and does not use timers to detect if an issue in transmission has occurred, as reliable data transfer is not its priority. Instead, it is up to the application running at the receiving host to detect if packet loss has occurred or to compute the checksum of the received UDP packet to detect errors, in which case, the host can then request retransmission of the packet.