ECE 358 - A5

Alexander Maguire
amaguire@uwaterloo.ca
20396195

June 17, 2014

$$\begin{bmatrix} i \\ 1 \end{bmatrix} \begin{bmatrix} -i & 1 \end{bmatrix} = \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix}$$

Question 4

Consider the Go-Back-N protocol with a sender window size of 3 and a sequence number range of 1024. Suppose that at time t, the next in-order packet that the receiver expects has sequence number k. Assume that the medium does not reorder messages.

What are the possible sets of sequence numbers inside the sender's window at time t?

The possible set of sequence numbers inside the sender's window at time t are

$$\begin{aligned} \exp lol\{c, c+1, c+2\} \quad & \text{for} \quad && c = k - i \\ & \text{where} \quad && i = [0, 1, 2, 3] \end{aligned}$$

The values for i represent the number of unreceived ACKs by the sender. If this number is 0, then the sender and receiver have synchronized windows (thus the sender's window begins at k as well). Should this number be 3, the receiver has all of the sent packets, but the sender is unaware.

What are all possible values of the ACK field in all possible messages currently propagating back to the sender at time t?

At time t, the possible ACKs propagating to the sender are:

$$\mathcal{P}\{k-3, k-2, k-1\} \cup \mathcal{P}\{k-1, k-1\}$$

where $\mathcal{P}$ denotes the power-set function (though technically this is a multi-set due to the second term).

The first term is the case if the sender has yet to send packet k, while the second term is if the sender *has* sent packet k, but it was lost in transport (prompting the receiver to ACK the previous packet for each packet it is unable to reorder).

Question 5

Consider the Go-Back-N and Selective Repeat protocols. Suppose the sequence number space is of size k. What is the largest allowable sender window for each protocol?

For Go-Back-N the largest allowable sender window is a size of $k - 1$. Since the receiver only ACKs packet n once it has received all data up to n, the only problematic case is if the sender is attempting to send packet (without loss of generality) $k + n \doteq 0 \mod k$ but the receiver is waiting for packet $n \doteq 0 \mod k$. This case can be avoided by ensuring at most $k - 1$ packets are in flight simultaneously, thus our limit on the window size.

The largest allowable sender window for Selective Repeat is $\lfloor k/2 \rfloor$. In SR, the worst possible case be that the receiver receive all packets in flight, but all ACKs are lost. The sender will then resent the entire window, none of which may intersect with the receiving window. For this reason, the sending window may be no larger than $\lfloor k/2 \rfloor$ lest packets be out-of-order on the receiving side.

Question 6

Consider Go-Back-N, Selective Repeat and TCP with no delayed ACKs. Assume that the timeout values for each is sufficiently long that 5 consecutive data segments and their corresponding ACKs can be received (if not lost in the channel) by the receiving host B and the sending host A respectively within the timeout interval. Suppose Host A sends 5 data segments to Host B, and the 2^{nd} segment sent from Host A is lost. Eventually, all 5 data segments are correctly received by Host B.

a) For each of the three protocols, how many segments has Host A sent in total, and how many ACKs has Host B sent in total? What are their sequence numbers?

GBN:

$\rightarrow$ **send** $\leftarrow$ **ack**

$\rightarrow 1 \leftarrow 1$
$\rightarrow 2$ (dropped)
$\rightarrow 3 \leftarrow 1$
$\rightarrow 4 \leftarrow 1$
$\rightarrow 5 \leftarrow 1$
(timeout)
$\rightarrow 2 \leftarrow 2$
$\rightarrow 3 \leftarrow 3$
$\rightarrow 4 \leftarrow 4$
$\rightarrow 5 \leftarrow 5$

Host A has sent 9 segments. B Host B has sent 8 ACKs.

SR:

$\rightarrow$ **send** $\leftarrow$ **ack**

$\rightarrow 1 \leftarrow 1$
$\rightarrow 2$ (dropped)
$\rightarrow 3 \leftarrow 3$
$\rightarrow 4 \leftarrow 4$
$\rightarrow 5 \leftarrow 5$
(timeout)
$\rightarrow 2 \leftarrow 2$

Host A has sent 6 segments. B Host B has sent 5 ACKs.

TCP:

Note: although TCP is supposed to ACK the *next* expected packet, in the trace below we are ACKing the last received packet in order to conform with the examples above.

$\rightarrow$ **send** $\leftarrow$ **ack**

$\rightarrow 1 \leftarrow 1$
$\rightarrow 2$ (dropped)
$\rightarrow 3 \leftarrow 1$
$\rightarrow 4 \leftarrow 1$
$\rightarrow 2 \leftarrow 4$
$\rightarrow 5 \leftarrow 5$

Host A has sent 6 segments. B Host B has sent 5 ACKs.

b) If the timeout values for all three protocols are must larger than $5RTT$, then which protocol successfully delivers all 5 data segments in the shortest time interval?

Assuming a large timeout value, TCP will perform this transfer the fastest since it doesn't rely on a timeout to retransmit the 2nd packet.

Question 7

Why do we avoid measuring the SampleRTT of retransmitted segments?

We avoid measuring retransmitted segments for SampleRTT since retransmissions represent an exceptional event and doing so will artificially inflate SampleRTT. There is no way to determine which direction (the original packet or the ACK) was lost, so measuring other packets is likely to be more accurate for the general case.

Why does fast retransmit wait for 3 ACKs of the same sequence number?

TCP waits for 3 ACKs of the same sequence number before retransmitting because duplicate ACKs will be generated if packets are reordered in-flight (a fairly common occurrence).

Question 8

In what way does TCP's 3-way handshake mitigate the problems depicted on slide 3-79?

3-way handshaking mitigates the problem of half-open connections by keeping track of which connection and accepting packets are in process. Since both sides are aware of the state of the connection in process, any duplicate packets are not considered new connections.

The second failure scenario is very similar, but the client is connected to two endpoints on the server. Again, this can be mitigated by allowing both parties to confirm the presence of the channel before any data is sent across it.