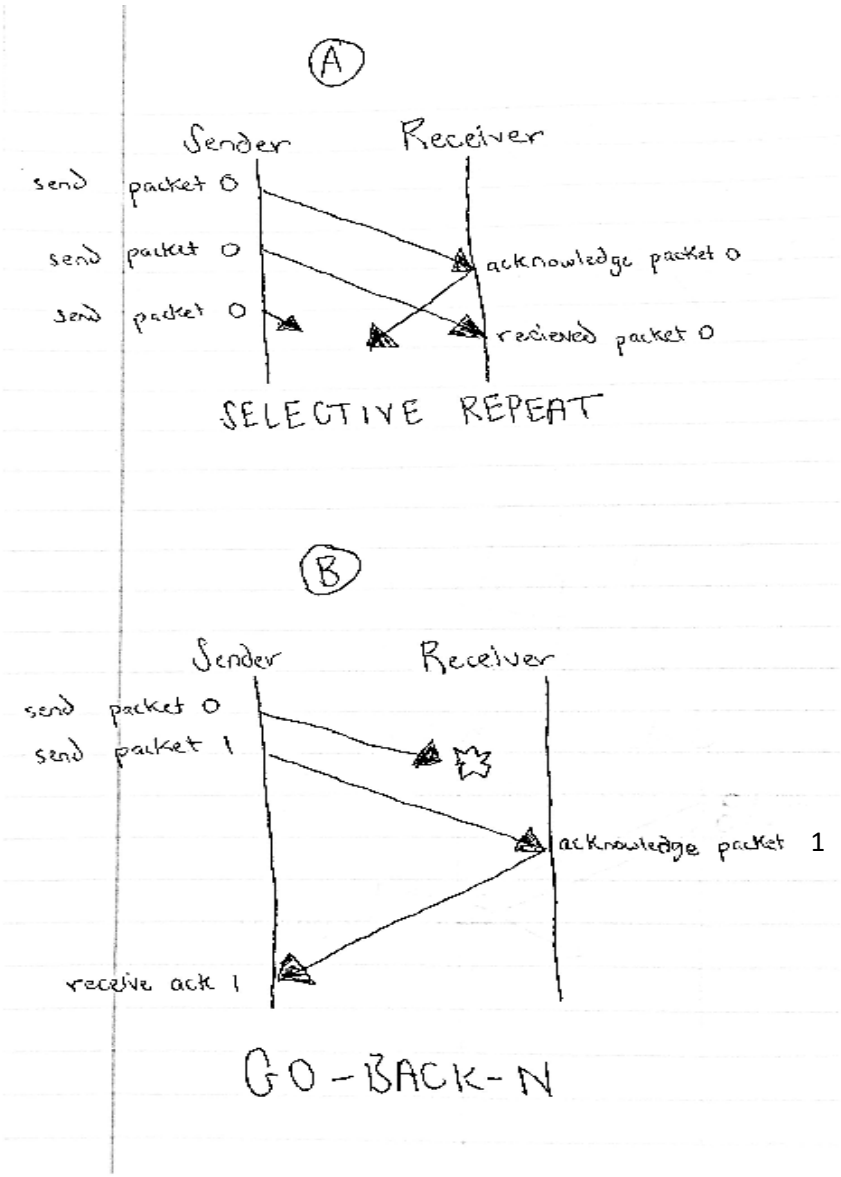
Jacob Alspaw
EECS 325 – Rabinovich
Introduction to Networks
13 October 2016

1) TCP will ensure that the black army will win. TCP implements RDT 3.0 (reliable data transfer) which uses algorithms such as Selective Repeat. With this protocol, the receiving black army will acknowledge each correctly received packet. Once a special acknowledgment packet has been received by the initial sending army, then both army's will know when to attack. If the initial sending army doesn't receive an acknowledgment, then neither army should attack.

2)



In part A, the sender will send packet 0 and wait for the acknowledgment. The sender will timeout and send packet 0 again. Meanwhile, the receiver has received the first packet 0 and sent its acknowledgment. The receiver is in the "wait for packet 1" state. However, it will receive packet 0 from the sender again due to the timeout. This behavior will most likely be found in a system using RDT 3.0 with Selective Repeat. The diagram is for N = 1, a usually improbable size for N.

In part B, the sender will send packet 0, but the packet will be lost somewhere along the network layout. The sender will send packet 1 without a need for acknowledgment of packet 0's arrival at the receiver because of RDT 3.0's use of Go-Back-N. The receiver will acknowledge packet 1 and not packet 0. Once this acknowledgment is received, the sender will be in the "wait for acknowledgment 0" state, but receives acknowledgment 1.

3) This situation is possible with Go-Back-N in RDT 3.0. If ACK K - 1 is lost in transmission, then the sender will be stuck in a "wait for ACK K - 1" state. The sender will continue with packet K because of Go-Back-N logic. If received, then ACK K will be sent. Eventually, after the N packets have been sent, the sender will need to go back to packet K - 1 to send the next N packets because the sender never received ACK K - 1. Therefore, if we send packet K - 1 again, then ACK K - 1 will also be sent.

4) For Go-Back-N, regardless of the packet reordering and with unlimited sequence numbers, if the acknowledgment for the first packet in a sequence is not received by the sender, then the sequence will not increment. The sender will remain in a "wait for first packet" state and continue to resend the N packets until it receives the corresponding acknowledgment. Go-Back-N will work.

For Selective Repeat, regardless of the packet reordering and with unlimited sequence numbers, if a sent packet or an acknowledgment packet is lost or reordered, then the sender will choose to send these packets again. The packets won't be marked received, but instead expected. If the acknowledgment is never received, then the sender will eventually time out and resend any "expected" packets.

5) A. The source sends packets 1 through 4 successfully to the receiver. The receiver sends the corresponding 4 acknowledgments which are lost. The receiver believes it is ready to acknowledge the next sequence of packets, however, because the acknowledgments were lost, the source will resend the same 4 packets. The receiver will acknowledge the wrong packets of data—the next four in the sequence, rather than the current four.

B. Because of the above scenario, the window size must be less than the sequence size. Therefore, with a sequence size of 4, a window size range of 1 to 3 inclusive is needed.

6) A. Suppose the source sends all 120 packets successfully, but the round trip transmission time is 2 seconds. The source will timeout and need to resend the original 120 packets. Further suppose that the round trip transmission time reaches 5 seconds for the packets that are resent. The source will receive the acknowledgments for the packets while the resent packets are in transmission. The next 120 packets are sent at the 2 second round trip time. These acknowledgments are received allowing for the next 120 packets with a round trip transmission time of 2 seconds; a wrap-around will need to happen. Due to the delay and timeout, we will incorrectly receive acknowledgments for the first 120 packets instead of packet numbers 241-360 which have shared sequence numbers.

B. The above scenario tells us that the transmission delay can affect TCP's ability to reliably transfer data. In this scenario, we should ensure sequence number > 2 * window size * maximum round trip transmission time. We get greater than 4800.

7) A. Based on the equation: $EstimatedRTT_n = (1 - \alpha) * EstimatedRTT_{n-1} + \alpha * SampleRTT$
If we put the equation into itself 3 more times, then…
$EstimatedRTT = \alpha * SampleRTT_1 + (1 - \alpha)\alpha SampleRTT_2 + (1 - \alpha)^2 SampleRTT_3 + (1 - \alpha)^3 SampleRTT_4$
If you look at our determined equation, you will see that as n increases, EstimatedRTT is more and more reliant on the value of the most recent RTT. Therefore, the EstimatedRTT value changes exponentially as the weight given to more recent SampleRTT's increases.

B. $EstimatedRTT_{sample} = (1 - \alpha) * EstimatedRTT + \alpha * SampleRTT$
$DevRTT_{sample} = (1 - \beta) * DevRTT + \beta * | SampleRTT – EstimateRTT |$
$TimeoutInterval_{sample} = EstimatedRTT + 4 * DevRTT$

$EstimatedRTT_{106} = 0.875 * 100 + 0.125 * 106 = 100.75$ ms
$DevRTT_{106} = 0.75 * 5 + 0.25 * 5.25 = 5.06$ ms
$TimeoutInterval_{106} = 100.75 + 4 * 5.06 = 120.99$ ms

$EstimatedRTT_{120} = 0.875 * 100.75 + 0.125 * 120 = 103.156$ ms
$DevRTT_{120} = 0.75 * 5.06 + 0.25 * 16.84 = 8.00$ ms
$TimeoutInterval_{120} = 103.156 + 4 * 8.00 = 135.156$ ms

$EstimatedRTT_{140} = 0.875 * 103.156 + 0.125 * 140 = 107.761$ ms
$DevRTT_{140} = 0.75 * 8.00 + 0.25 * 32.239 = 14.05$ ms
$TimeoutInterval_{140} = 107.761 + 4 * 14.05 = 163.961$ ms

8) The following situation illustrates why TCP does not measure SampleRTT for retransmitted segments because it will often times calculate an incorrect result. Say the source sends packet 1, and no response causes the source to timeout. The source then sends a new copy of the same packet. If the source measures SampleRTT for the retransmitted packet (the

copy of packet 1) and, shortly after retransmitting, an acknowledgment for the original packet arrives, then the source will mistakenly take this acknowledgment as an acknowledgment for the source's resent packet and calculate an incorrect value of SampleRTT.