Computer Networks

Spring 2022

Homework Assignment of Week 3 (Link Layer)

Problem 1: Textbook Chapter 6, P6

P5. Consider the 5-bit generator G=10011, and suppose that D has the value 1010101010. What is the value of R?

If we divide 10011 into 1010101010 0000, we get 1011011100, with a remainder of R=0100.

Problem 2: Textbook Chapter 3, R9

R9. In our *rdt* protocols, why did we need to introduce sequence numbers?

Sequence numbers are required for a receiver to find out whether an arriving packet contains new data or is a retransmission.

Problem 3: Textbook Chapter 3, R10

R10. In our *rdt* protocols, why did we need to introduce timers?

To handle losses in the channel. If the ACK for a transmitted packet is not received within the duration of the timer for the packet, the packet (or its ACK or NAK) is assumed to have been lost. Hence, the packet is retransmitted.

Problem 4: Textbook Chapter 3, R11

R11. Suppose that the roundtrip delay between sender and receiver is constant and known to the sender. Would a timer still be necessary in protocol *rdt 3.0*, assuming that packets can be lost? Explain.

A timer would still be necessary in the protocol rdt 3.0. If the round trip time is known then the only advantage will be that, the sender knows for sure that either the packet or the ACK for the packet has been lost, as compared to the real scenario, where the ACK might still be on the way to the sender, after the timer expires. However, to detect the loss, for each packet, a timer of constant duration will still be necessary at the sender.

Problem 5: Textbook Chapter 3, P9

P9. Give a trace of the operation of protocol *rdt3.0* when data packets and acknowledgment packets are garbled. Your trace should be similar to that used in Figure 3.16.

Suppose the protocol has been in operation for some time. The sender is in state "Wait for call from above" (top left hand corner) and the receiver is in state "Wait for 0 from below". The scenarios for corrupted data and corrupted ACK are shown in Figure 1.

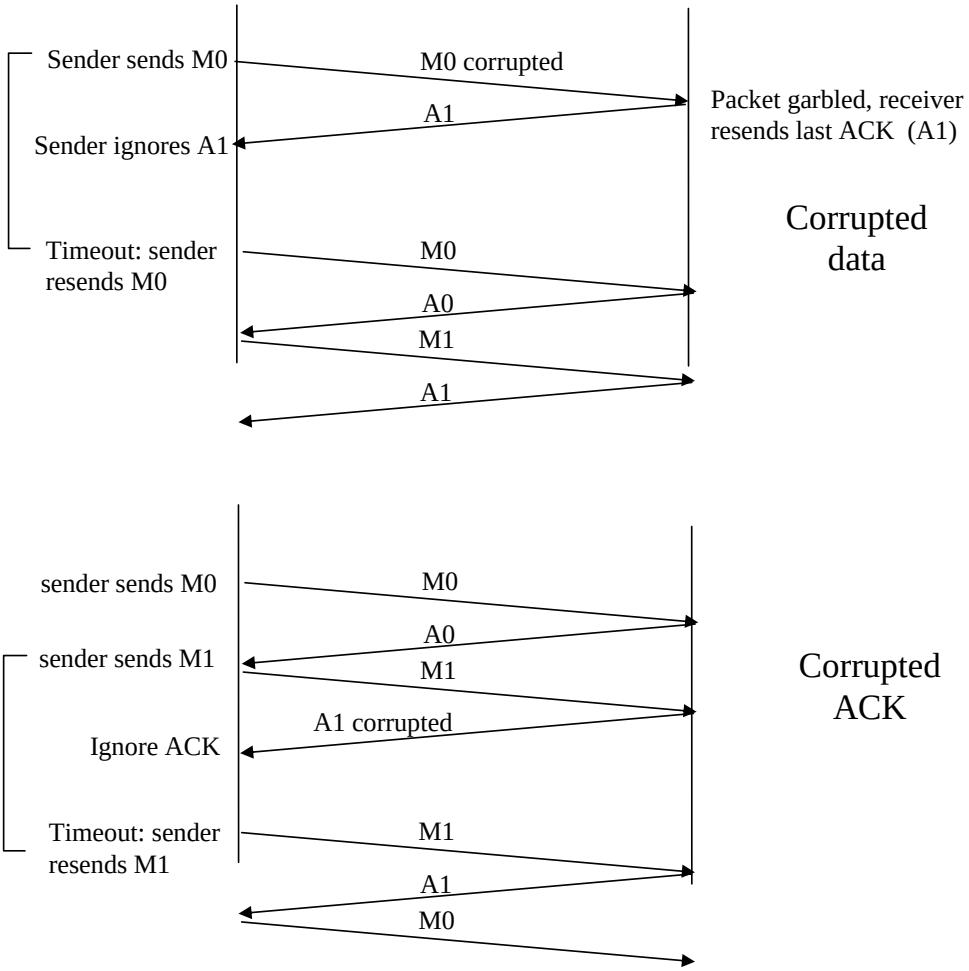

Figure 1: rdt 3.0 scenarios: corrupted data, corrupted ACK

Problem 6: Textbook Chapter 3, P23

P23. Consider the GBN and SR protocols. Suppose the sequence number space is of size *k*. What is the largest allowable sender window that will avoid the occurrence of problems such as that in Figure 3.27 for each of these protocols?

In order to avoid the scenario of Figure 3.27, we want to avoid having the leading edge of the receiver's window (i.e., the one with the "highest" sequence number) wrap around in the sequence number space and overlap with the trailing edge (the one with the "lowest" sequence number in the sender's window). That is, the sequence number space must be large enough to fit the entire receiver window and the entire sender window without this overlap condition. So we need to determine how large a range of sequence numbers can be covered at any given time by the receiver and sender windows.

Suppose that the lowest-sequence number that the receiver is waiting for is packet m. In this case, it's window is [m, m+w-1] and it has received (and ACKed) packet m-1 and the w-1 packets before that, where w is the size of the window. If none of those w ACKs have been yet received by the sender, then ACK messages with values of [m-w, m-1] may still be propagating back. If no ACKs with these ACK numbers have been received by the sender, then the sender's window would be [m-w, m-1].

Thus, the lower edge of the sender's window is m-w, and the leading edge of the receivers window is m+w-1. In order for the leading edge of the receiver's window to not overlap with the trailing edge of the sender's window, the sequence number space must thus be big enough to accommodate 2w sequence numbers. That is, the sequence number space must be at least twice as large as the window size, $k \geq 2w$.