Solution set from Ch3

Problem 6.

The sender side of protocol rdt3.0 differs from the sender side of protocol 2.2 in that timeouts have been added. We have seen that the introduction of timeouts adds the possibility of duplicate packets into the sender-to-receiver data stream. However, the receiver in protocol rdt.2.2 can already handle duplicate packets. (Receiver-side duplicates in rdt 2.2 would arise if the receiver sent an ACK that was lost, and the sender then retransmitted the old data). Hence the receiver in protocol rdt2.2 will also work as the receiver in protocol rdt 3.0.

Problem 9.

The protocol would still work, since a retransmission would be what would happen if the packet received with errors has actually been lost (and from the receiver standpoint, it never knows which of these events, if either, will occur).

To get at the more subtle issue behind this question, one has to allow for premature timeouts to occur (which is not what the question asked, sorry). In this case, if each extra copy of the packet is ACKed and each received extra ACK causes another extra copy of the current packet to be sent, the number of times packet \( n \) is sent will increase without bound as \( n \) approaches infinity.

Problem 16.

a) Here we have a window size of \( N=3 \). Suppose the receiver has received packet \( k-1 \), and has ACKed that and all other preceding packets. If all of these ACK's have been received by sender, then sender's window is \([k, k+N-1]\). Suppose next that none of the ACKs have been received at the sender. In this second case, the sender's window contains \( k-1 \) and the \( N \) packets up to and including \( k-1 \). The sender's window is thus \([k-N,k-1]\). By these arguments, the sender's window is of size 3 and begins somewhere in the range \([k-N,k]\).

b) If the receiver is waiting for packet \( k \), then it has received (and ACKed) packet \( k-1 \) and the \( N-1 \) packets before that. If none of those \( N \) ACKs have been yet received by the sender, then ACK messages with values of \([k-N,k-1]\) may still be propagating back. Because the sender has sent packets \([k-N, k-1]\), it must be the case that the sender has already received an ACK for \( k-N-1 \). Once the receiver has sent an ACK for \( k-N-1 \) it will never send an ACK that is less that \( k-N-1 \). Thus the range of in-flight ACK values can range from \( k-N-1 \) to \( k-1 \).