ELEC 5220/6220 Solution 3

Problem 1

	source port numbers	destination port numbers
a) $A \rightarrow S$	467	23
b) $B \rightarrow S$	513	23
c) $S \rightarrow A$	23	467
d) $S \rightarrow B$	23	513

e) Yes.
f) No.

Problem 3

Note, wrap around if overflow.

$$\begin{array}{ccccccccc} & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ + & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ \hline & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array} \qquad \begin{array}{ccccccccc} & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ + & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ \hline & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{array}$$

One's complement = 1 1 0 1 0 0 0 1.

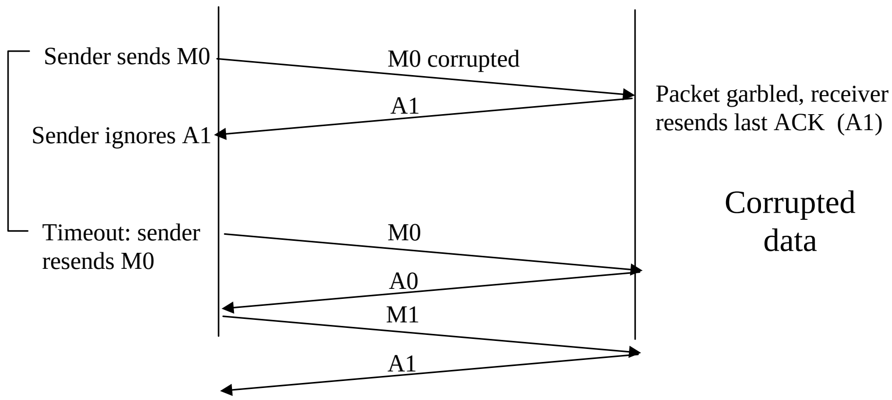
To detect errors, the receiver adds the four words (the three original words and the checksum). If the sum contains a zero, the receiver knows there has been an error. All one-bit errors will be detected, but two-bit errors can be undetected (e.g., if the last digit of the first word is converted to a 0 and the last digit of the second word is converted to a 1).

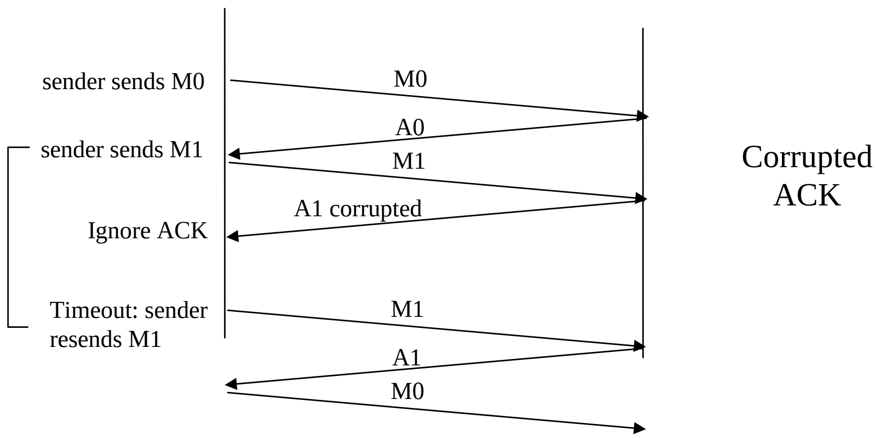
Problem 5

No, the receiver cannot be absolutely certain that no bit errors have occurred. This is because of the manner in which the checksum for the packet is calculated. If the corresponding bits (that would be added together) of two 16-bit words in the packet were 0 and 1 then even if these get flipped to 1 and 0 respectively, the sum still remains the same. Hence, the 1s complement the receiver calculates will also be the same. This means the checksum will verify even if there was transmission error.

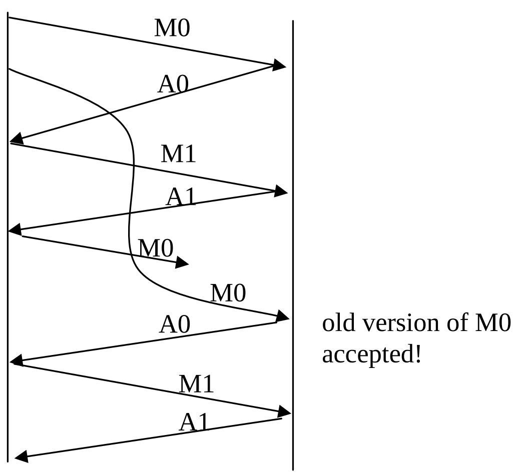
Problem 9

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.





Problem 13



Problem 15

It takes 12 microseconds (or 0.012 milliseconds) to send a packet, as $1500*8/10^9$=12 microseconds. In order for the sender to be busy 98 percent of the time, we must have

$$util = 0.98 = (0.012n) / 30.012$$

or n approximately 2451 packets.

Problem 22

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 senders 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.

Problem 23

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, $k \geq 2w$.

Problem 24

a) True. Suppose the sender has a window size of 3 and sends packets 1, 2, 3 at $t0$. At $t1$ $(t1 > t0)$ the receiver ACKS 1, 2, 3. At $t2$ $(t2 > t1)$ the sender times out and resends 1, 2, 3. At $t3$ the receiver receives the duplicates and re-acknowledges 1, 2, 3. At $t4$ the sender receives the ACKs that the receiver sent at $t1$ and advances its window to 4, 5, 6. At $t5$ the sender receives the ACKs 1, 2, 3 the receiver sent at $t2$. These ACKs are outside its window.
b) True. By essentially the same scenario as in (a).
c) True.
a) True. Note that with a window size of 1, SR, GBN, and the alternating bit protocol are functionally equivalent. The window size of 1 precludes the possibility of out-of-order packets (within the window). A cumulative ACK is just an ordinary ACK in this situation, since it can only refer to the single packet within the window.