Problem 1

The sender side of rdt3.0 simply ignores (that is, takes no action on) all received packets that are either in error or have the wrong value in the acknum field of an acknowledged packet. Suppose that in such circumstances, rdt3.0 were simply to retransmit the current data packet. Would the protocol still work? (Hint: Consider what would happen if there were only bit errors; there are no packet losses but premature timeouts can occur. Consider how many times the nth packet is sent, in the limit as n approaches infinity)

The protocol would still work. This re-transmission of data packets is exactly what occurs in rdt 3.0 when a packet is dropped, or a timeout is reached in the sender. The issue that would arise in this case is an issue of efficiency, which differs based on the value of the timeout. If this version of rdt 3.0 had a very small timeout value for the senders, then the protocol would cause a continuous pattern of retransmitting duplicates of each packet, regardless of if they were necessary or not (ie the nth packet is sent at least twice each time). This is because a premature timeout followed by an incorrect ack of one of the duplicate packets would cause a chain of retransmissions to occur.

If the timeout value is large, however, then this system could actually improve efficiency. This is because if a corrupted ack is received by the sender, the new rdt 3.0 would transmit the previous packet right away, while the old version would continue the timer and wait for a proper ack value. The time of the timeout is thus saved in this new protocol.

Problem 2

Consider a reliable data transfer protocol that uses only negative acknowledgments. Suppose the sender sends data only infrequently. Would a NAK-only protocol be preferable to a protocol that uses ACKs? Why? Now suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In this second case, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

The NAK-only protocol is not advantageous for the situation in which the sender sends data infrequently. This is because for the receiver to realize there was a mistake in transmission, the only way to detect this would be once the packet directly after this one is sent. This is because for the receiver to know that they did not receive a correct packet, they must compare with the packet immediately following it, and send a NAK then. The time spent waiting for the next packet to arrive to send the NAK message is therefore an inefficiency.

This protocol, however, would be more advantageous for the case when data is sent frequently. Since the data will be sent often, it will be a fast process for the receiver to realize that it got a packet with the wrong. Also, since there is a lot of data sent, assuming that the number of NAKs needed is less that the number of ACKS, this system would therefore save on overall messages sent. Therefore, the system is simplified for the case where a lot of data is sent frequently.

Problem 3

Consider the GBN protocol with a sender window size of 6 and a sequence number range of 1,024. Suppose that at time t, the next in-order packet that the receiver is expecting has a sequence number of k. Assume that the medium does not reorder messages. Answer the following questions:

(a) What are the possible sets of sequence numbers inside the senders window at time t? Justify your answer.

(b) What are all possible values of the ACK field in all possible messages currently propagating back to the sender at time t? Justify your answer.

For this problem: [k, k+5] means a window with the sequence of values:
k, k+1, k+2, k+3, k+4, k+5.

(a) If the next in order packet received is k, then this means that the ACK messages for the previous k-1 packets have all been received. There are a few cases in which this is possible:
In the first case, the sender could be waiting to receive all of the ACKs from the previous window. In this case, the receiver has sent back all of the 6 ACKs and now waits for the k'th packet, but the sender is still waiting to receive the ACKs. Thus the entire window of 6 packets would soon be shifting, and k would become the first packet in the new window. This means the sequence of numbers in the sender's current window at this time t would be the values from [k-6, k-1].
In another case, which is related to the first, the kth packet could be the first in the sender's window. This means that all k-1 previous packets have been received by the receiver, and the sender has received the corresponding ACKs for each of them. Thus, the sequence of numbers in the window in this case would be the values from [k, k+5].
In the third case, the window would be a combination of the first two cases. This means that the k'th packet could be in any of the 5 other positions in the current sender's window. This means that the range of values for the window would be from k-5 (ie the case with window from [k-5, k]) up until k+4 (the case where the window is from [k-1, k+4]). Any cases in between this would also be possible such as: [k-2, k+3] or [k-1, k+4], etc. The start of the window can one of the values from k-6 to k. the end of the window can be a value from k to k+5.

(b) At time t, the possible messages being propagated back to the sender would be Acks for packets within one of the possible windows, but before packet k. At the same time, for the lowest possible window case of k-6 to k-1, all previous acks MUST have been sent back. This means that, based on part a, the ACKs that would be possible are ACKs of the range of sequence numbers from k-7 to k-1. The k-7th ACK could still be propagation back to to the sender at time t because once this ACK is received, then it is ensured that all ACKs below this value won't be seen, and we would move into the ACKs that correspond to the possible window values.

Problem 4

Follow the same problem setting in Page 62 of Slides Chapter3-2020.ppt. Suppose packet size is 4KB (*i.e.* 4000 bytes), bandwidth is 8Mbps, and one-way propagation delay is 20 msec. Assume there is no packet corruption and packet loss.

(a) Suppose sender window size is 5, will the sender be kept busy? If yes, explain why. If not, What is the effective throughput?

(b) What is the minimum sender window size to achieve full utilization? Then how many bits would be needed for the sequence number field?

(a) The sender will not be kept busy. since the total transmission time for one packet would be $\frac{L}{R} = \frac{4KB}{8Mb/sec} = 0.004s$, the total transmission time for all 5 packets will be 20 ms. since the RTT time for 1 packet is 40 ms, the ACK message for the first packet is received after 44 ms. Therefore, there is a time of 24 ms in which the sender has to wait for a response and is not transmitting.

Based on the example given in the slides and the posts based on piazza, I took the formula for Effective throughput to be: $\frac{N}{RTT+\frac{L}{R}} = \frac{20KB}{0.44ms} = 454.5KBps = 3636.3Kbps$.

(b) To have a utilization of 1, you would need to be transmitting during the entire time elapsed until the first ACK is received (as this relationship would then propagate to all following ACKs for packets). Thus, you need to transmit during the 24ms in part a that the sender has to wait for the response. With a transmission time per packet of 4ms, this means that you would need to transmit and additional 24/4 = 6 packets in each window. Thus, the minimum sender window size to achieve full utilization would be 5+6 = 11 packets. // For this window size, you would need a total of 4 bits for the sequence number field. This is because with 4 bits, you can hold a maximum number of 15, which is larger than the sequence number we'd need if we had a window size of 11.
in the case of using selective repeat, we would need 5 bits to hold the sequence number, because the sequence Number size would be 2*N, where N is the window size.

Therefore, depending on the protocol implementation, you would need either 4 or 5 bits (4 minimum for GBN, 5 for SR).

Problem 5

Answer True or False to the following questions and briefly justify your answer:

(a) With the Selective Repeat protocol, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.

(b) With Go-Back-N, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.

(c) The Stop-and-Wait protocol is the same as the SR protocol with a sender and receiver window size of 1.

(d) Selective Repeat can buffer out-of-order delivered packets, while GBN cannot. Therefore, SR saves network communication cost (by transmitting less) at the cost of additional memory.

(a) True. Assuming that packet corruption is possible, suppose that with a window of size 4, the sender is currently on packets 5, 6, 7, and 8 in the current window. If the first packet transmitted, packet 5, is then corrupted, then the receiver will send back an ack message for the previous packet, packet 4 in the last window. Thus, the situation is possible.

(b) True; same justification as part can apply here as well, and an ACK from a previous window can also be received. Another possible situation that could apply to BOTH situations(in parts a and b)is that a premature timeout is reached by the sender, which causes the packets in the current window to be retransmitted. If the first set of ACKs returns after this duplicate transmission, then the window shifts, and later on the second set of ACKs for the previous window's duplicate transmission will be received. Therefore, the situation is possible in this way too.

(c) True; Stop and wait protocol is basically the same as selective repeat if there is a window size of 1. This is because in both cases, a single packet is sent at a time and a single ACK is processed before moving onto the next packet.

(d) True; Selective Repeat only sends individual ACKs as well as individual packet retransmission. With out of order delivery, Selective Repeat would save network communication cost of duplicate packet delivery compared to GBN. The increase in memory comes from the need for a buffer with a size of N, where N= the number of packets.