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).

When rdt3.0 sends a data packet, it receives an acknowledgement indicating whether the receiver received the package or not. If the receiver sends an error message or indicates packet lost, rdt3.0 tries to retransmit the data packet again and again until the receiver successfully receives the packet. Constant attempts to retransmit data will result in a premature timeout by sending too many unnecessary packets, and creates increased congestion on the network. Therefore, it may still work, but it lacks efficiency. Additionally, incorrect packages may be sent because of collisions due to retransmission of several packages.

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?

NAK-only protocol recognizes lost data only when a subsequent file/packet is lost. If we send packets 1, 2 and 3 in order and the receiver only receives packet 3, then it will notify the sender that packet 2 is lost.

① In the case where data is sent infrequently, NAK-only protocol is not suitable because it will only realize data loss on arrival of the subsequent packet, which could be sent much later than the initial packet that was lost. Therefore, there will be long delays before we realize a packet is lost.

② In the case of frequent data transfer with minimal losses, NAK-only protocol is very suitable because any data losses are detected quickly without long gaps, and NAK is rarely sent to the sender because of minimal data loss instances. We also would not need to send an ACK every time data is received.

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.

(a) Next in-order packet that receiver is expecting is K, indicating that K-1 has been acknowledged / ACKed by the receiver.
Best case: ACK of K-1 is received by the sender so that the sender can move it out of the window. Then, the window is $[K, K+N-1] = [K, K+5]$
Worst case: None of the ACKs of packet K-1 and the 5 packets preceeding it were ACKed by the sender. Then the window is $[K-N, K-1] = [K-6, K-1]$.
Therefore, all other cases must lie in between these lower and upper bounds and the sets of sequence numbers in the window are in the range $[K-N, K]$ (at time t)

(b) Since the receiver is expecting K, K-1 has been ACKed by the receiver. From the sender's perspective, K-1 has been sent. With a window size of 4, atleast K-5 has been ACKed and removed from the window by the sender. Therefore, the set of possible ACK values in all messages being sent to the sender is $[K-5, K-1]$ (at time t)

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 sener 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) Packet size = 4KB = 4000 bytes = 32000 bits
Bandwidth = 8Mbps = 8×10^6 bps
RTT = propogation delay = 20msec = 0.02sec (one way)

$$d_{Trans} = L/R = \frac{32000}{8 \times 10^6} = \frac{4}{10^3} = 0.004 \text{ sec}$$

$$\text{Utilization} = \frac{NL/R}{RTT + L/R} = \frac{5 * d_{Trans}}{2(0.02) + d_{Trans}} = \frac{5(0.004)}{0.04 + 0.004} = \frac{0.02}{0.044} \cong 0.4545 \cong 0.456$$

Therefore, since utilization is 0.456, the sender is busy approx. half the time and still has room to be busier. Therefore, the sender is not kept busy.

$$\text{effective throughput} = \frac{L}{RTT + L/R} = \frac{32000}{2(0.02) + 0.004} = 7272 \text{ bits/sec} \approx \boxed{7.27 \text{ Kbps}}$$

(b) Using the same formula for utilization, we need to solve for N.

$$1 = \frac{N(0.004)}{0.04 + 0.004} \Rightarrow N = \frac{1(0.044)}{0.004} = 11$$

If we decide to use selective repeat, then we would need a window size of 2N for full utilization → 22.

The number of bits needed for the sequence number field will be $\lceil \log_2(2N) \rceil = \lceil \log_2(22) \rceil = \lceil 4.459 \rceil = \boxed{5}$

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 – This is possible when the receiver sends ACKs back but the sender never receives them so it resends the packets. The sender then receives the original ACKs at the previous timestep and advances its window up n positions. The receiver sends new ACKs for the retransmitted packets back to the sender who will eventually receive them, but it will now fall outside of its original window.

(b) True – This is true using roughly the same logic as that described in (a).

(c) True – This is true because SR can only advance its window when the sender receives an ACK back, even if it sends multiple packets at once. Stop-and-wait sends a packet and only sends the next packet after the sender receives an ACK. Because of the nature of the window, with a window size of 1, these will effectively be the same.

(d) True – SR can buffer out-of-order delivered packets and retransmit only the packet lost individually. As longer as receiver receives a packet, it will move to the next window. GBN takes more time than SR because the whole window needs to wait for a packet to be received. SR takes more memory because each packet in SR has a timeline which costs additional memory but less time than GBN.