Problem 1

Suppose that the UDP receiver computes the Internet checksum for the received UDP segment and finds that it matches the value carried in the checksum field. Can the receiver be absolutely certain that no bit errors have occurred? Explain.

Write your solution to Problem 1 in this box

No, the receiver cannot be certain. Coincidental errors are unlikely but still possible. For example, if the checksum has a bit error WHILE the actual payload has a bit error in the exact same spot then it's possible that when the checksum is checked on the receiver side, it still passes the algorithm's test. The checksum algorithm takes all the bits in the segment and runs a mathematical function that could return true or false depending on equality. The algorithm could return true even if errors happen.

Ex: 1010011 returns 8 = 1000 through the checksum algo.
1000011 returns 5 = 0101 through the checksum algo.

$\downarrow$ if bit error happens here

$\downarrow$ and many bit errors happen here $\rightarrow$ checksum algo would still believe everything is right.

Knowing that checksum uses addition of bits, it's also possible for bits to flip

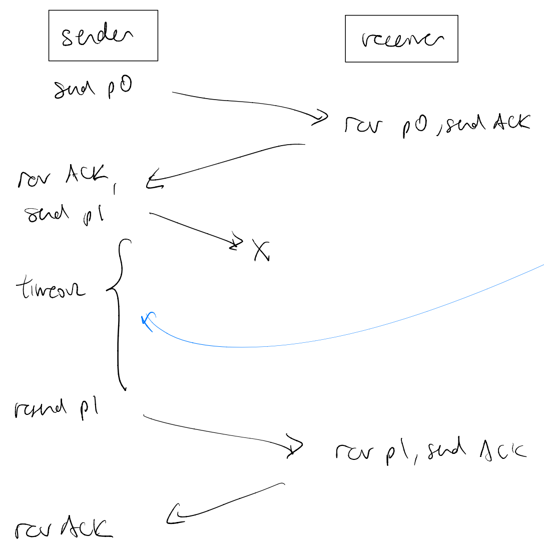
$1+0=1 \rightarrow 0+1=1$ which would lead to the same sum.

Problem 2

Consider a channel that can lose packets but has a maximum delay that is known. Modify protocol *rdt2.1* to include sender timeout and retransmit. Please draw the diagram like Slide 52 and justify why your protocol can communicate correctly over this channel.

Write your solution to Problem 2 in this box

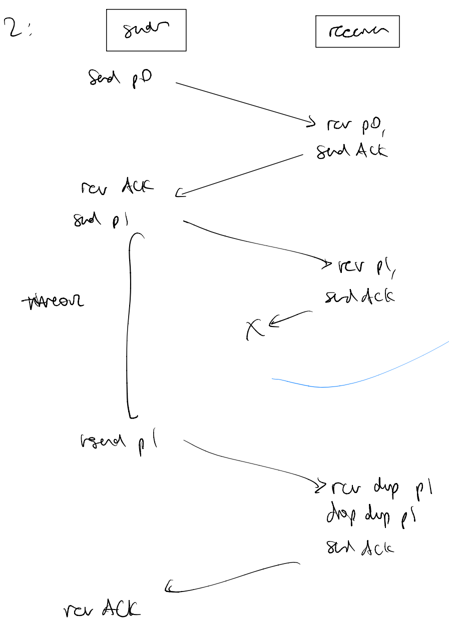
Case1:



create timer that is on average longer than avg RTT delay inside wall clock on sender side + no ACK → retransmit

this case = packet dropped when sender → receiver transmission. Just resend when clock expire.

Case 2:



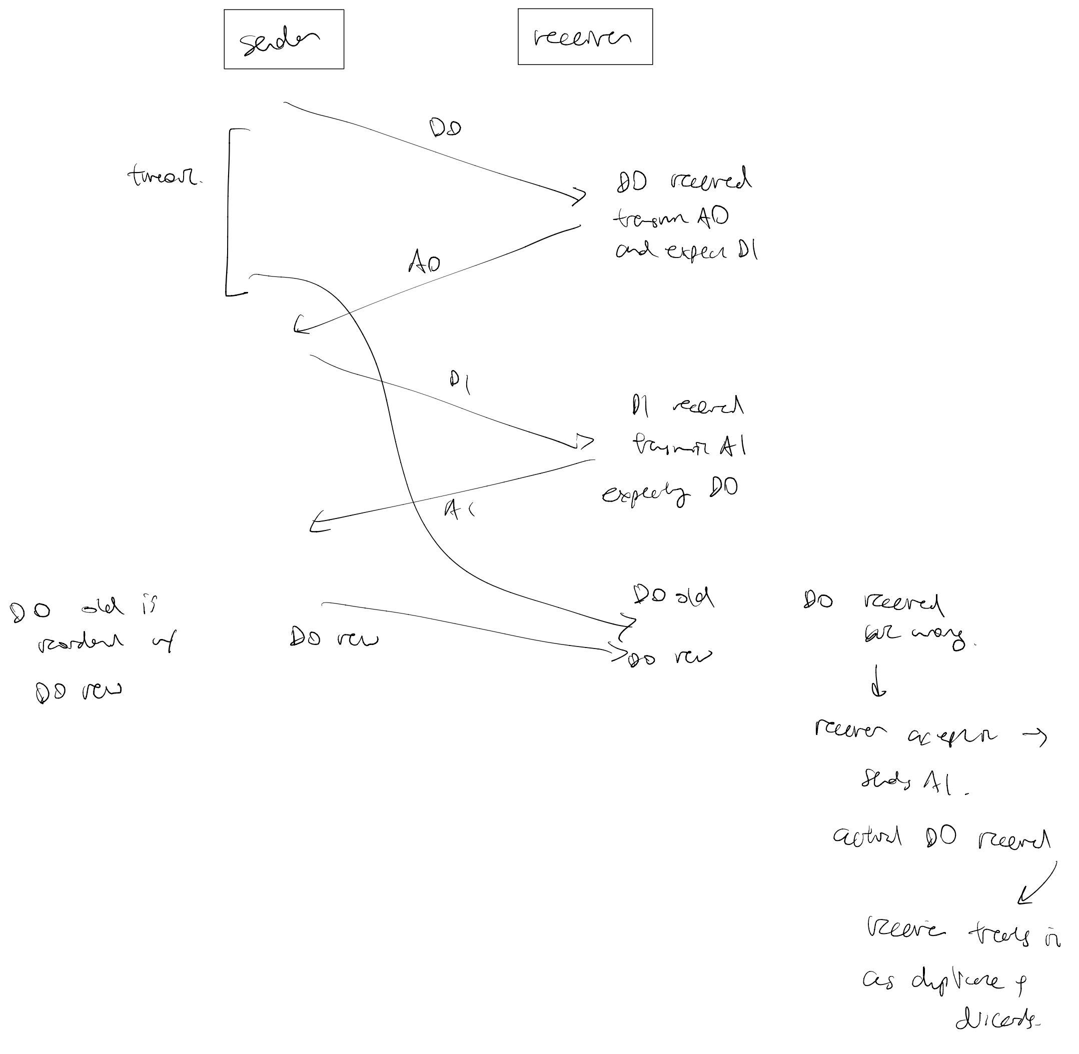
similar to prev case except this time the fail is on the way from receiver → sender when ACK is being transferred

still works though b/c on retransmission, after clock expire, the receiver side will know that duplicate was sent, drop the dup, and send an appropriate ACK

Problem 3

Consider the *rdt3.0* protocol. Draw a diagram showing that if the network connection between the sender and receiver can reorder messages (that is, that two messages propagating in the medium between the sender and receiver can be reordered), then the alternating-bit protocol will not work correctly (make sure you clearly identify the sense in which it will not work correctly). Your diagram should have the sender on the left and the receiver on the right, with the time axis running down the page, showing data (D) and acknowledgment (A) message exchange. Make sure you indicate the sequence number associated with any data or acknowledgment segment.

Write your solution to Problem 3 in this box



D0 old is reordered of D0 new

D0 received but wrong.
↓
receiver accepts →
sends A1.
actual D0 received
↙
receiver treats it as duplicate & discards.

Problem 4

Consider the GBN protocol with a sender window size of 4 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.

Write your solution to Problem 4 in this box

a) case 1: the seq #'s are k, k + 1, k + 2, and k + 3 in our window if no packets are sent out and no waiting is occurring
case 2: k -4, k - 3, k - 2, k - 1 is in our window if all are sent out and all were ACKed by receiver and not received yet by sender. Once received, our window will change again

note that these are the two end possibilities. everything in between is possible as well, such as k - 3, k - 2, k - 1, k; k - 2, k - 1, k, k + 1, and so on.

b) All the ACK values can be between k - 4 and k - 1. this is the most extreme left side possibility so the possible values of the ACK field in all possible messages currently propagating back to the sender at time t are k - N all the way to k - 1.

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&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.

Write your solution to Problem 5 in this box

a) True. if we have a sender window size equal to 10 and we send max 10 packets, and the receiver sends back all 10 ACKs, but none of them arrive on time (aka all timeout). the sender will retransmit all 10 packets, and THEN when the 10 original ACKS finally reach the sender, the receiver will then retransmit the new 10 ACKs for packets outside current window (the window has moved after seeing that the old 10 ACKS arrived back at sender)

b) True again. It is the same thing as part (a) except for the fact that the sender will only timeout from the oldest unACKed packet before beginning to retransmit all 10 packets

c) True because SR and Stop&Wait operate in the same way

d) True because looking at an example from the lecture, assume we send packets 0, 1, 2, and 3. suppose we lose packet 2 (but packet 3 arrives), making the receiver buffer it because it has to wait for 2 to arrive before sending it to the application layer. In fact, the receiver might buffer packet 4 and 5 as well if those arrive properly before 2 arrives. We must use additional memory to buffer all of these later packets and also does not need to resend those packets because they are saved on the receiver side so yes saves network communication cost at the cost of additional memory (in the buffer).