1. Suppose a reliable data transfer protocol (Stop-and-Wait) that uses only negative acknowledgments (NAK) - i.e., the receiver sends a NAK when it detects a corrupted packet, otherwise it does nothing.

(i) If the sender sends data infrequently, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

A:

No, NAK-only can just let the sender know whether the receiver has received the package a before it has already received package a+1. First, when the sender sends data infrequently, it's a waste of time waiting for the next package.

Secondly, if the sender only sends one package and it's lost, the receiver will never know any information about this package, also the sender wouldn't get the NAK. Similarly, the sender never knows the last package's status.

(ii) If the sender transmits packets frequently and the end-to-end connection suffers from very few losses, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

A:

Yes, under this situation, the error will be detected very soon, also because we don't need to send ACK, we can save a lot of time in this process.

2. If a UDP receiver computes the checksum for a 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.

A: No, the checksum is mainly calculated by adding two breaking segments (usually 16-bit words), and then taking 1s complement. So if we exchange the 0 and 1 in both two breaking segments, the adding results will be the same. Then the checksum will be the same too. For example: adding 1 0 to 0 1 is equal to adding 0 1 to 1 0.

3. (Problem P3 from the textbook, Chapter 3, modified): Suppose you have three bytes: 01010011, 01100110, 01110100. Find the ones complement of the sum of these bytes.

A:First step: 01010011+01100110=10111001

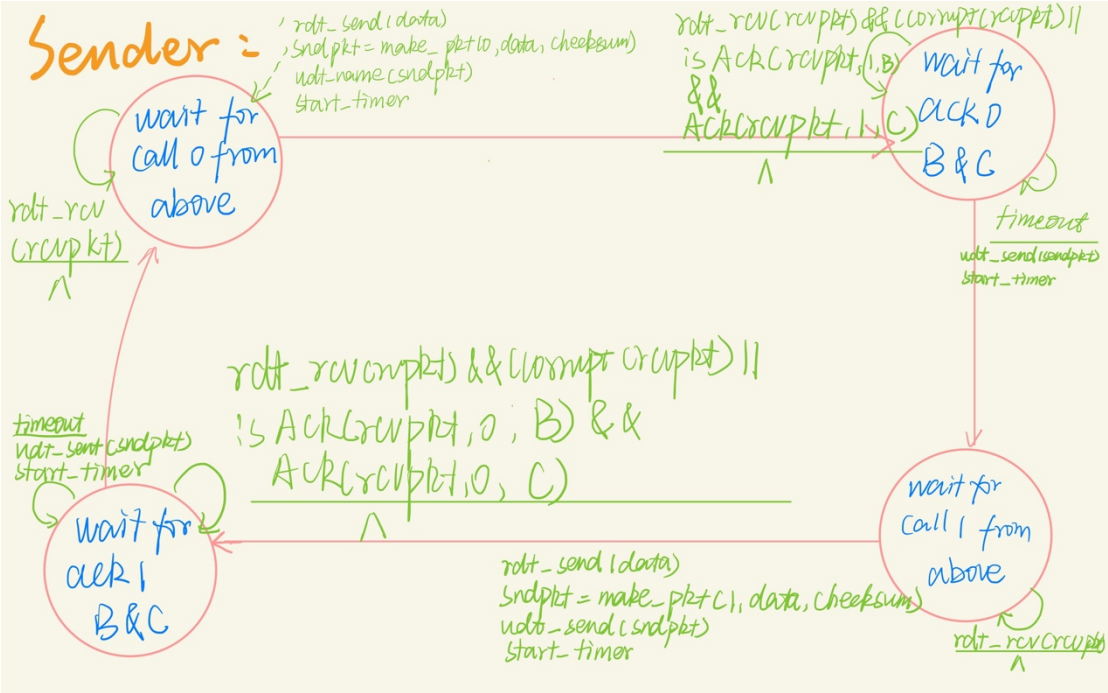
10111001+01110100=100101101

Wraparound----sum: 00101110

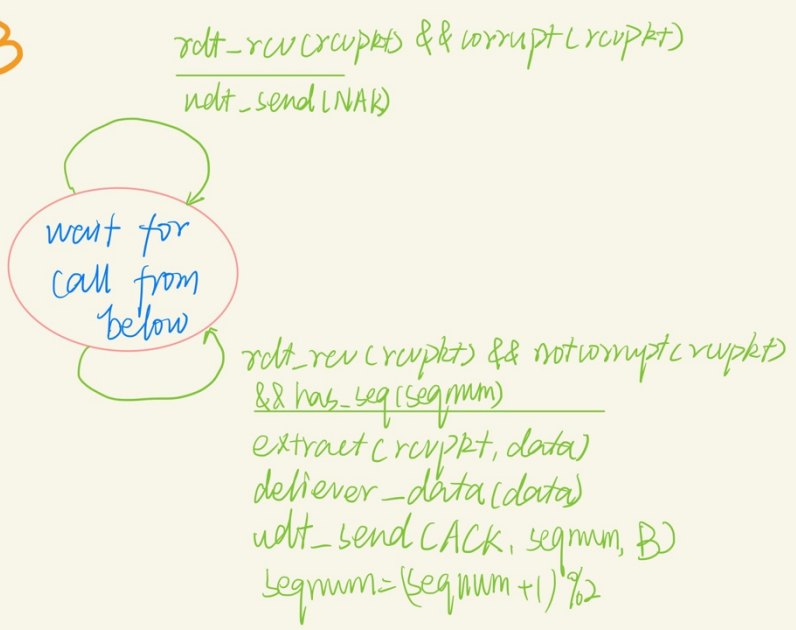
Checksum(taking 1s complement): 11010001

4. (Problem P19 from the textbook, Chapter 3, modified): Consider a scenario where a Host A wants to simultaneously send packets to Host B and C. A is connected to B and C through a broadcast channel - A sends once and both B and C can receive it. However, packet losses or errors are independent. That is, as an example, A might transmit a packet, it is correctly received by B but not by C. Design a stop and wait like error control protocol for *reliably* transferring packets from A to both B and C. A will not accept data from the application layer till *both* B and C have correctly received the current packet. After you design the protocol, draw an FSM that shows what the sender and receiver are doing. Note that the FSM is identical for B and C. Clarify what information is contained in the packets.

A:



Receiver B



Receiver C