Homework 2

0. Academic integrity

I have read and understood the course academic integrity policy.

1. UDP and TCP checksum

a) Suppose you have the following two bytes: 00001001 and 01100101. What is the 1s complement of the sum of these two bytes?

00001001 + 01100101 = 01101110
and its 1s complement is 10010001

b) Suppose you have the following tow bytes: 01101001 and 11101101. What is the 1s complement of the sum of these two bytes?

01101001 + 11101101 = 101010110
and its 1s complement is 010101001

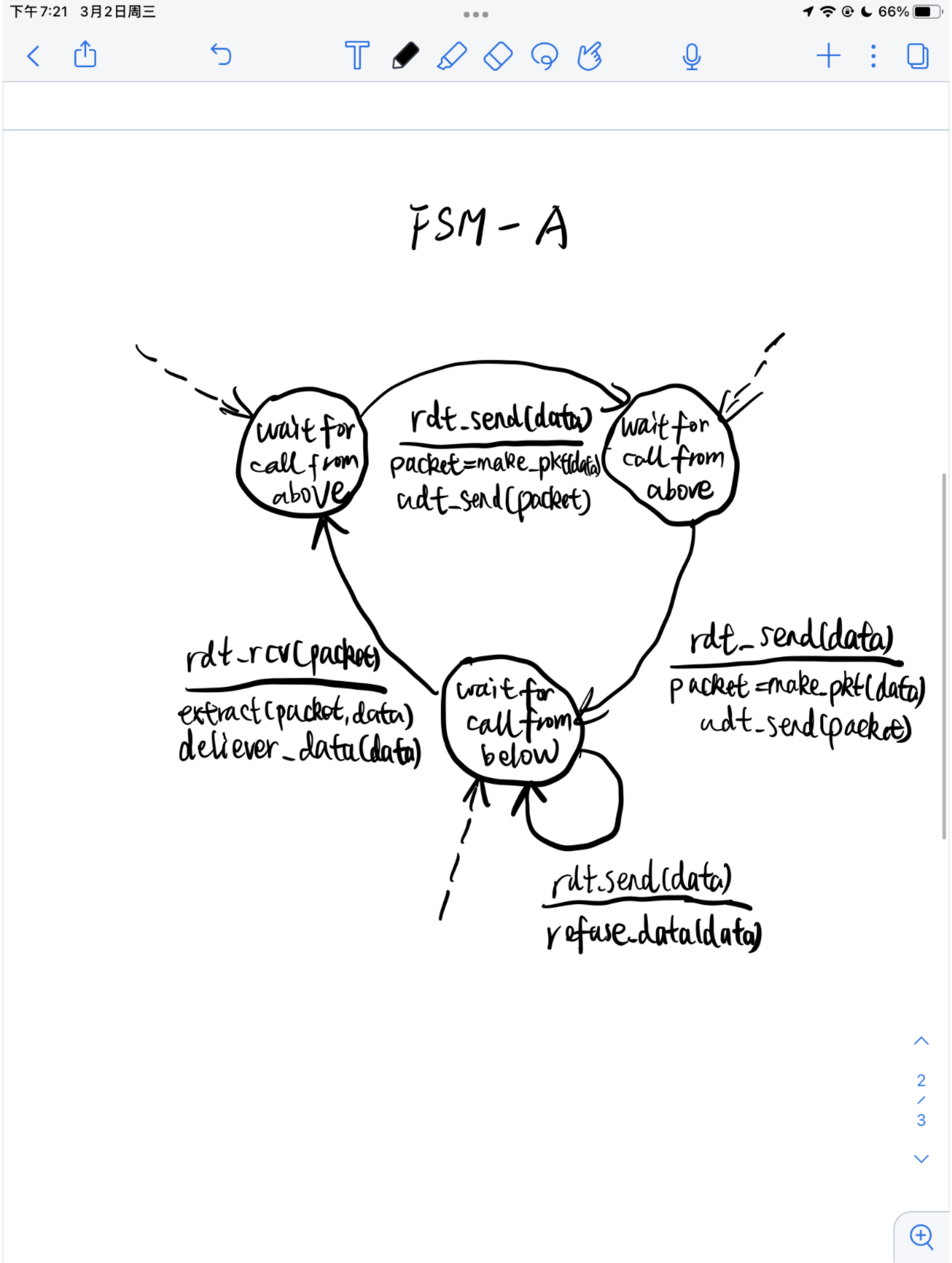
c) For the bytes in part (a), give an example where one bit is flipped in each of the two bytes and yet the 1s complement of the sum doesn't change.

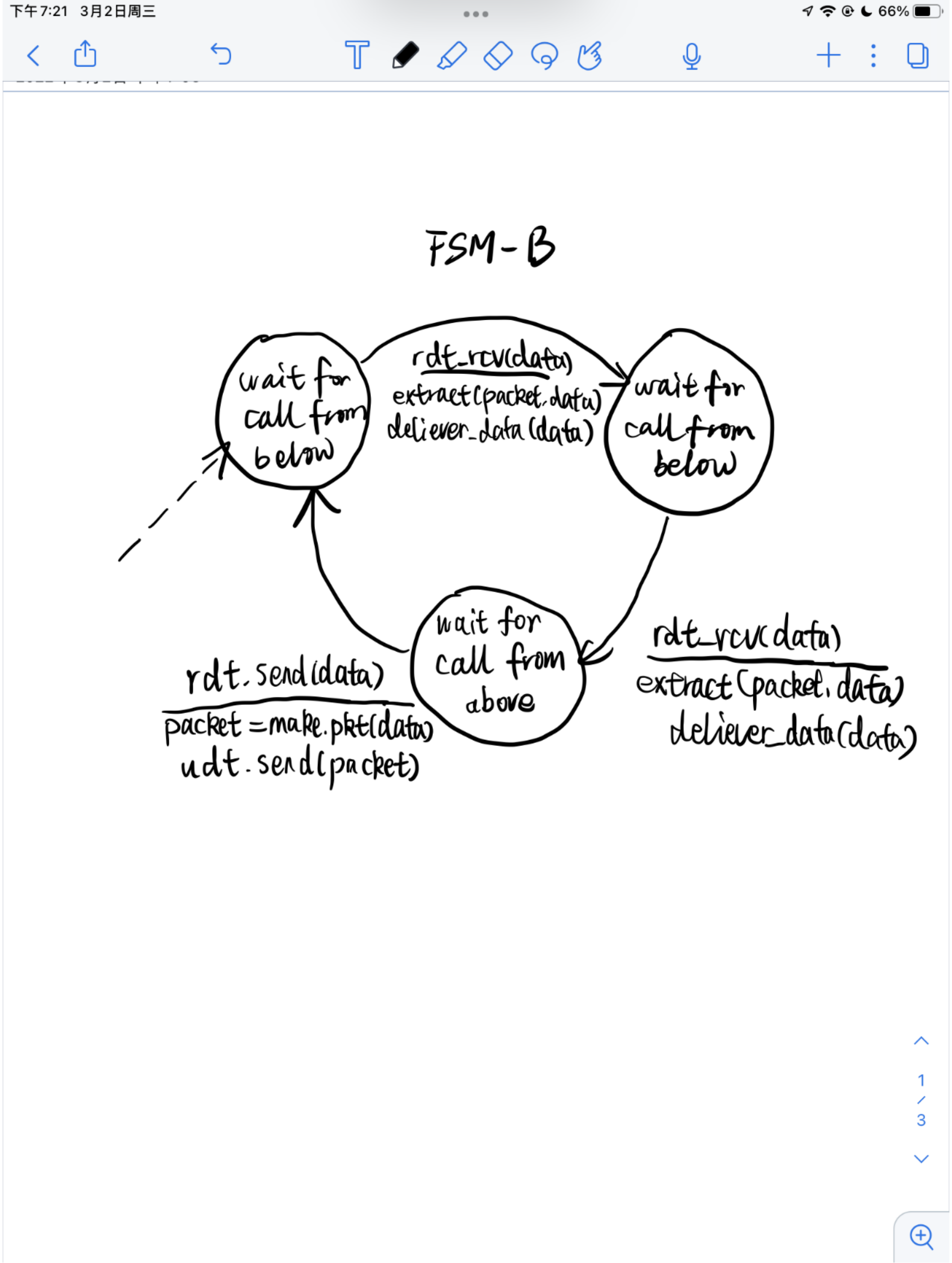
First byte = 00001101; second byte = 01100001

2. **Simple synchronized message exchange protocol** – Consider two network entities, A and B, which are connected by a bi-directional channel that is perfect (that is, any message sent will be received correctly; the channel will not corrupt, lose, or reorder packets). A and B are to deliver data messages to each other in the following manner: A is to send *two* messages to B, then B is to send *one* message to A. Then the cycle repeats.

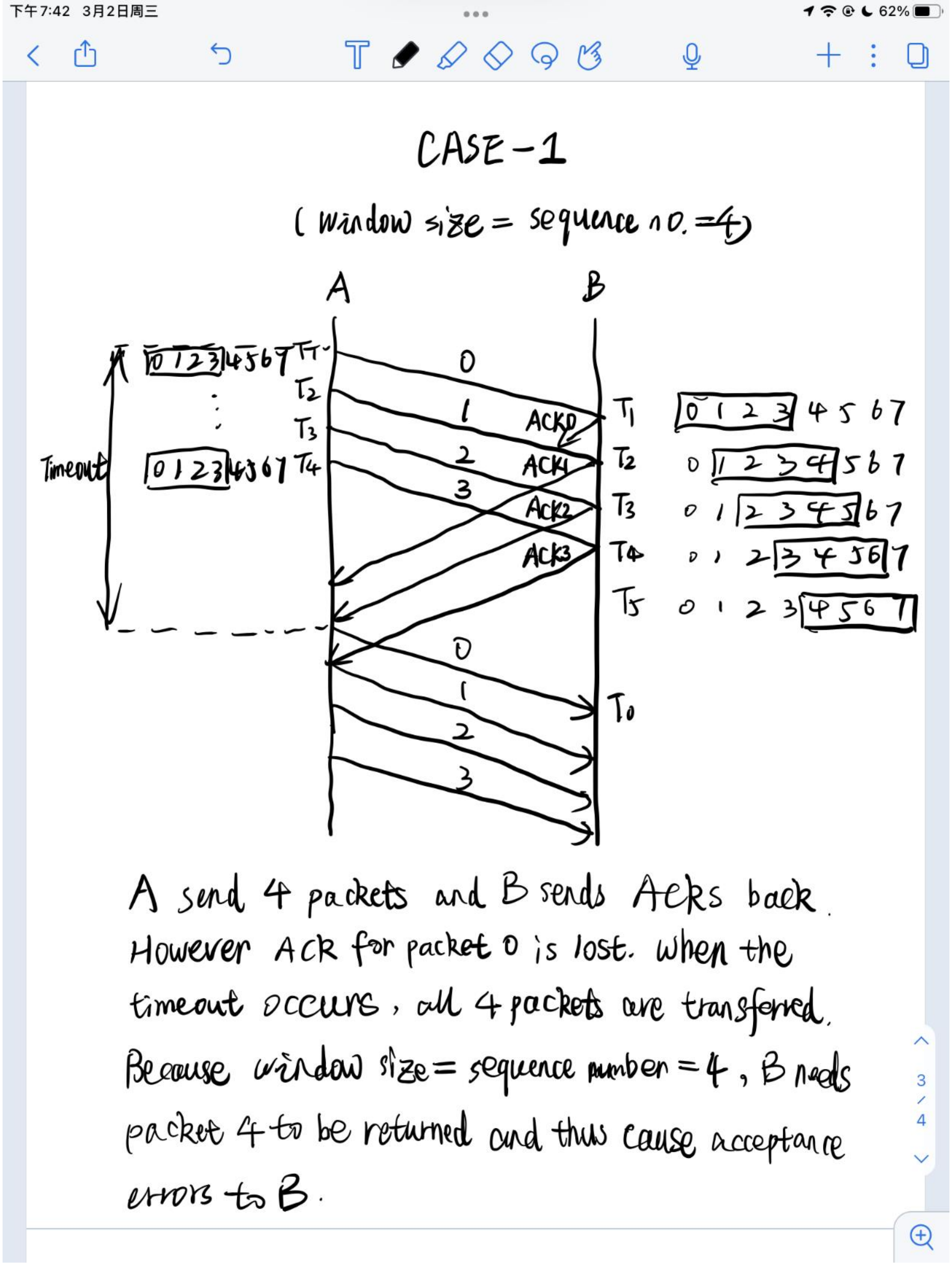
Draw an FSM specification for this protocol (one FSM for A and one FSM for B). Don't worry about a reliability mechanism here; the main point is to create an FSM specification that reflects the synchronized behavior of the two entities. You should use the following events and actions, which have the same meaning as in protocol rdt1.0, shown on page 206 of the textbooks: `rdt_send(data),packet=make_pkt(data),udt_send(packet), rdt_rcv(packet),extract(packet,data),deliver_data(data), refuse_data(data)`

Make sure your protocol reflects the strict alternation of sending between A and B. Also, be sure to indicate the initial states for A and B in your FSM description. (The key idea is to use states to track how many messages A has sent: one or two)





3. **Can a window size be too large for a sequence number space?** – Consider the Go-Back-N protocol. Suppose the size of the sequence number space (number of unique sequence numbers) is N, and the window size is N. Show a scenario (give a timeline trace showing the sender, reciver, and the messages they exchange over time) where the Go-Back-N protocol will not work correctly in this case. What is the maximum window size for which the protocol will work correctly when the size of the sequence number space is N?

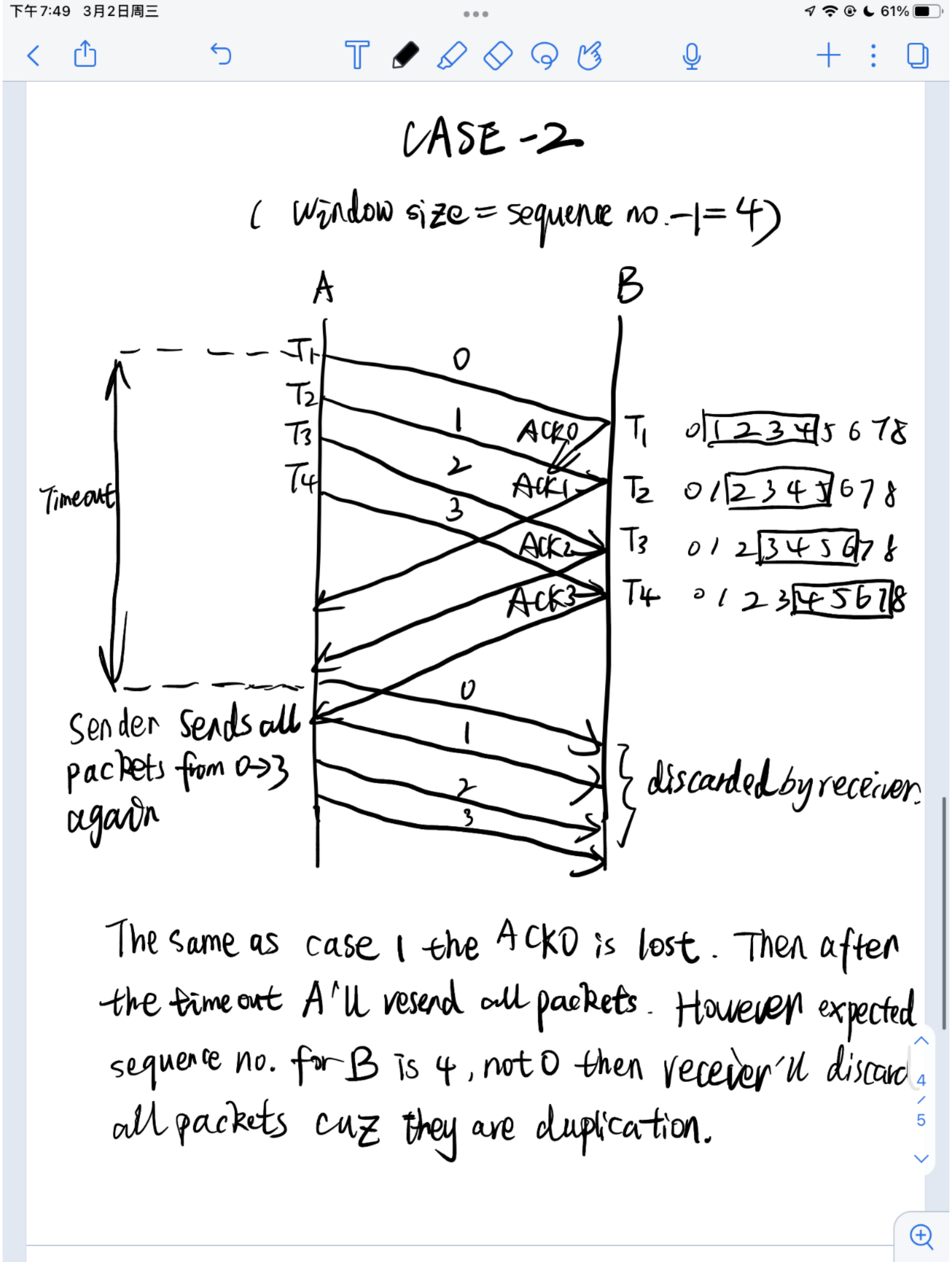


下午7:42 3月2日周三 62%

CASE-1

(window size = sequence no. =4)

A send 4 packets and B sends ACks back. However ACK for packet 0 is lost. when the timeout occurs, all 4 packets are transfered. Because window size = sequence number = 4, B needs packet 4 to be returned and thus cause acceptance errors to B.



CASE -2

(Window size = sequence no. -1 = 4)

Sender sends all packets from 0->3 again

The same as case 1 the ACK0 is lost. Then after the time out A'll resend all packets. However expected sequence no. for B is 4, not 0 then receiver'll discard all packets cuz they are duplication.

As the discussion above, the window size less than or equal to $N - 1$ will work correctly.

4. **TCP sequence numbers** – Consider transferring an enormous file of L bytes from host A to host B. Assume an MSS of 536 bytes.

   a) What is the maximum value of L such that TCP sequence numbers are not exhausted?

The sequence number increments only by the size of the sending data and the maximum value of L is 2^{32} bytes.

b) For the value of *L* you obtained in (a), find how long it takes to transmit the file. Assume that a total of 66 bytes of transport, network, and link layer headers are added to each segment before the resulting packet is sent over a 155 Mbps link. Ignore flow control and congestion control so A can pump out the segment back to back and continuously.

The number of segments: $\left\lceil\frac{2^{32}}{536}\right\rceil = 8012999$.

Total bytes of header: 528857934 bytes.

Total bytes transmitted: $2^{32} + \text{header} = 4.8 \times 10^9$ bytes.

As a consequece, it takes 249s to transmit the file.