Homework 4

Sean Beardsley

February 26, 2023

1 P4

1.1 Part a

Adding the two bytes gives 11000001. Taking the one's complement gives 00111110.

1.2 Part b

Adding the two bytes gives 01000000, which means the one's complement gives 10111111.

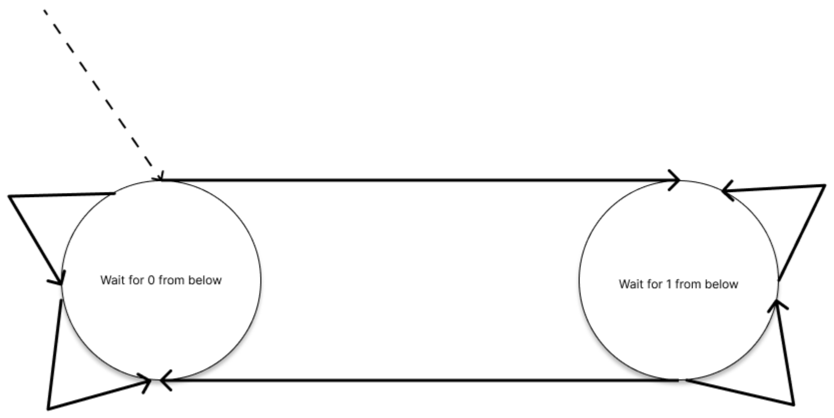
1.3 Part c

For the first byte, you would have: 01010100. For the second byte, you would have: 01101101. This would lead to the same one's complement.

2 P6

Suppose the sender is in the state "Wait for call 1 from above" and the receiver is in the state "Wait for 1 from below." The sender sends a packet with a sequence number of 1, and moves to "Wait for acknowledgement or negative acknowledgement 1," state, waiting for an acknowledgement or a negative acknowledgement. Suppose now the receiver receives the packet with sequence number 1 correctly, sends an acknowledgement, and transitions to state "Wait for 0 from below," waiting for a data packet with sequence number 0. However, the acknowledgement is clearly corrupted. When the rdt2.1 sender gets the corrupted acknowledgement, it sends the packet with sequence number 1 again. However, the receiver is waiting for a packet with a sequence number of 0 and always sends a negative acknowledgment when it does not receive a packet with a sequence number of 0. Hence the sender will always be sending a packet with sequence number 1, and the receiver will always be negatively acknowledging that packet. Neither will progress forward from that state, which means both parties are in a deadlock state.

3 P8



4 P11

If the sending of this message were to be removed, the sending and receiving sides would be in a deadlock state, waiting for an event that would never occur. Here is an example scenario:

- The sender sends packet 0, enters the “Wait for Acknowledgement 0” state, and then waits for a packet back from the receiver.
- The receiver is in the “Wait for 0 from below” state, and receives a corrupted packet from the sender. Suppose it does not send anything back, and enters the ”wait for 0 from below” state again.

Now, the sender is awaiting an acknowledgement of some sort from the receiver, and the receiver is waiting for a data packet form the sender. Since neither have the ability to move to a different state, a deadlock state has occurred.

5 P12

The protocol would still work, since a re-transmission would be what occurs if the packet received with errors was actually been lost, which makes sense, as from the receiver standpoint, it never knows which of these events, if either, will happen. To tackle the more nuanced issue behind this question, which the hint gets at, one has to permit timeouts that are premature. In this instance, if both each extra copy of the packet is acknowledged, and each received extra acknowledgment causes another extra copy of the current packet to be sent, the number of times packet n is sent will increase without bound as n approaches infinity.