Problem 1

Host A sends 5 data segments to Host B, the 2nd segment (sent from A) is lost. This loss is then detected by the protocol-specific means and the protocol invokes recovery actions. In the end, all 5 data segments are correctly received by Host B.

1. If A and B use Go-back-N for the data delivery, what is the total number of segments that Host A sent out (including retransmissions)? And what is the total number ACKs that Host B sent?
2. If A and B use TCP for the data delivery (no delayed ACKs), what is the total number of segments that Host A sent out (including retransmissions)? And what is the total number ACKs that Host B sent?
3. Assume that the retransmission timer is set to 10*RTT, which of the above two protocols (Go-back-N and TCP) finish the data delivery first?

Note: Answering questions 2 and 3 requires the knowledge about TCP fast retransmit, which is described on Lecture-7, slides 24-25, that will be discussed in next Monday lecture. If anyone wants to finish the homework this weekend, you probably can figure out how TCP fast retransmit works by taking a careful look at Lecture-7, slides 24-25.

1. GoBackN:
   A sends 9 segments in total. They are initially sent segments 1, 2, 3, 4, 5 and later re-sent segments 2, 3, 4, and 5.
   B sends 8 ACKs. They are 4 ACKS with sequence number 1, and 4 ACKS with sequence numbers 2, 3, 4, and 5.
2. TCP:
   A sends 6 segments in total. They are initially sent segments 1, 2, 3, 4, 5 and later re-sent segments 2. B sends 5 ACKs. They are 4 ACKS with sequence number 2. There is one ACK with sequence numbers 6. Note that TCP always send an ACK with expected sequence number.
3. TCP. This is because TCP uses fast retransmit without waiting until time out.

Problem 2

Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 126. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 80 and 40 bytes of data, respectively. In the first segment, the sequence number is 127, the source port number is 302, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A.

1. In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number?
2. If the first segment arrives before the second segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number, the source port number, and the destination port number?
3. If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number?

1. In the second segment from Host A to B, the sequence number is 207, source port number is 302 and destination port number is 80.
2. If the first segment arrives before the second, in the acknowledgement of the first arriving segment, the acknowledgement number is 207, the source port number is 80 and the destination port number is 302.
3. If the second segment arrives before the first segment, in the acknowledgement of the first arriving segment, the acknowledgement number is 127, indicating that it is still waiting for bytes 127 and onwards.

Problem 3

Follow the same problem setting in Page 37 of Slides lecture-06. Suppose packet size is 4000 bits, bandwidth is 2Mbps, and propagation delay is 15 msec. Ignore packet loss.

1. Suppose window size is 10, will the sender be kept busy? If yes, explain why. If not, What is the effective throughput?
2. What is the minimum window size to achieve full utilization? Then how many bits would be needed for the sequence number field?

1. No. RTT is 30 ms. Transmission delay is 2ms. It takes 2ms * 10 = 20ms to transmit 10 segments. Therefore, the link is not fully utilized.
   The effective throughput is (2Mbps * 2 * 10) / 32.
2. 16. Calculated by (RTT + transmission delay) / transmission delay. 5 bits since $2^5 \geq 16 * 2$. We need window-size $\leq$ (Max. seq num + 1) / 2

Problem 4

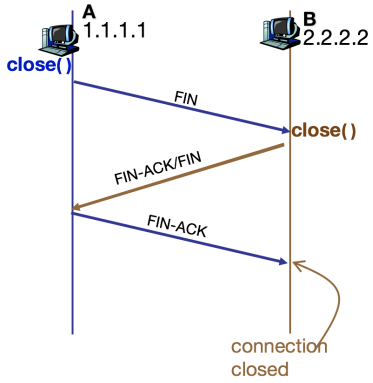
Consider a reliable data transfer protocol that uses only negative acknowledgments. Suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. Would a NAK-only protocol be preferable to a protocol that uses ACKs? Why? Now suppose the sender sends data only infrequently. In this second case, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

In a NAK only protocol, the loss of packet x is only detected by the receiver when packet x+1 is received. That is, the receivers receives x-1 and then x+1, only when x+1 is received does the receiver realize that x was missed. If there is a long delay between the transmission of x and the transmission of x+1, then it will be a long time until x can be recovered, under a NAK only protocol.

On the other hand, if data is being sent often, then recovery under a NAK-only scheme could happen quickly. Moreover, if errors are infrequent, then NAKs are only occasionally sent (when needed), and ACK are never sent a significant reduction in feedback in the NAK-only case over the ACK-only case.

Problem 5

A sends a TCP FIN message to B to close the TCP connection with B, the TCP header of A's FIN message is shown below. When B receives A's TCP FIN, it also decides to close the connection, so B sends a combined FIN and FIN-ACK message, whose TCP header is also shown below. Please fill in all the fields with a question mark in this TCP header.



... src 1.1.1.1, dst: 2.2.2.2	
s_port: 1030	d_port: 4000
seq_no: 548	
ack_no: 1000	
header length / not used / 0 1 0 0 0 1	rcv_window: 200
checksum: ...	0 (ignore this field)

... src 2.2.2.2, dst: 1.1.1.1	
s_port: ?	d_port: ?
seq_no: ?	
ack_no: ?	
header length / not used / 0 ? 0 0 ? ?	rcv_window: 400
checksum: ...	0 (ignore this field)

```
s_port: 4000
d_port: 1030
seq_no: 1000
ack_no: 549
control bits (6-bit): 010001
```