CSEE-5110: Problem Set 5, Fall 2015 ANS

1. Consider a reliable data transfer protocol that uses only negative acknowledgements. Suppose the sender sends data only infrequently

   (a) 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.

   (b) Now suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In this second case, would a NAK only protocol be preferable to a protocol that uses ACKS? Why?

   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.

2. UDP and TCP use 1s compliment for their checksums. Suppose you have the following three 8-bit bytes: 01010111, 01110100, 01011100.

   (a) What is the 1s compliment of the sum of these 8-bit bytes?
   11010111

   (b) Why is it the UDP takes the 1s compliment of the sum and not just the sum?
   To detect errors, the receiver adds the four words (the three original words and the checksum).

   (c) With the 1s compliment scheme, how does the receiver detect errors?
   If the sum contains a zero, the receiver knows there has been an error.

   (d) Is it possible that a 1 bit error will go undetected?
   All one-bit errors will be detected.

   (e) Is it possible a 2 bit error will go undetected?
   All one-bit errors will be detected, but two-bit errors can be undetected (e.g., if the last digit of the first word is converted to a 0 and the last digit of the second word is converted to a 1).

3. Suppose two stations A and B are connected with a high speed 1Gbps link. Suppose the round trip propagation delay on the link is 40ms. Packet lengths are 1,000 bytes (8,000 bits).

   (a) How big would the window size have to be for the channel utilization to be 90 percent?
   It takes 8 microseconds (or 0.008 milliseconds) to send a packet, as $1000*8/10^9 = 8$ microseconds. In order for the sender to be busy 90 percent of the time, we must have

$$u = 0.90 = (0.008n)/40.008$$

or n is approximately 4501 packets.

4. Consider a scenario in which Host A and Host B want to send messages to Host C. Hosts A and C are connected by a channel that can lose and corrupt (but not reorder) messages. Hosts B and C are connected by another channel (independent of the channel connecting A and C) with the same properties. The transport layer at Host C should alternate in delivering messages form A and B to the layer above (that is, it should first deliver the data from a packet from A, then from B, and so on).

   (a) Design a stop-and-wait-like error-control protocol for reliably transferring packets from A and B to C, with alternating delivery at C as described above. Give FSM descriptions of A and C. (*Hint:* The FSM for B should be essentially the same as A.)

   (b) Give a description of the packet format(s) used.

5. Consider a GBN protocol with a sender window size of 3 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.

   (a) What are the possible sets of sequence numbers inside the sender's window at time t? Justify your answer.

   Here we have a window size of N=3. Suppose the receiver has received packet k-1, and has ACKed that and all other preceeding packets. If all of these ACK's have been received by sender, then sender's window is [k, k+N-1]. Suppose next that none of the ACKs have been received at the sender. In this second case, the sender's window contains k-1 and the N packets up to and including k-1. The sender's window is thus [k-N,k-1]. By these arguments, the senders window is of size 3 and begins somewhere in the range [k-N,k].

   (b) What are all the possible values of the ACK field in all possible messages currently propagating back to the sender at time t? Justify your answer.
   If the receiver is waiting for packet k, then it has received (and ACKed) packet k-1 and the N-1 packets before that. If none of those N ACKs have been yet received by the sender, then ACK messages with values of [k-N,k-1] may still be propagating back. Because the sender has sent packets [k-N, k-1], it must be the case that the sender has already received an ACK for k-N-1. Once the receiver has sent an ACK for k-N-1 it will never send an ACK that is less that k-N-1. Thus the range of in-flight ACK values can range from k-N-1 to k-1.

6. Consider the GBN and SR protocols. Suppose the sequence number space is of size k.

   (a) What is the largest allowable sender window that will avoid the occurrence of problems caused by exhausting sequence numbers? In order to avoid the scenario of Figure 3.27, we want to avoid having the leading edge of the receiver's window (i.e., the one with the "highest" sequence number) wrap around in the sequence number space and overlap with the trailing edge (the one with the "lowest" sequence number in the sender's window). That is, the sequence number space must be large enough to fit the entire receiver window and the entire sender window without this overlap condition. So - we need to determine how large a range of sequence numbers can be covered at any given time by the receiver and sender windows. Suppose that the lowest-sequence number that the receiver is waiting for is packet m. In this case, it's window is $[m, m + w - 1]$ and it has received (and ACKed) packet $m - 1$ and the $w - 1$ packets before that, where w is the size of the window. If none of those w ACKs have been yet received by the sender, then ACK messages with values of $[m - w, m - 1]$ may still be propagating back. If no ACKs with these ACK numbers have been received by the sender, then the sender's window would be $[m - w, m - 1]$. Thus, the lower edge of the sender's window is $m - w$, and the leading edge of the receivers window is $m + w - 1$. In order for the leading edge of the receiver's window to not overlap with the trailing edge of the sender's window, the sequence number space must thus be big enough to accommodate $2w$ sequence numbers. That is, the sequence number space must be at least twice as large as the window size, $k \geq 2w$.

7. Consider transferring an enormous file of L bytes from Host A to Host B. Assume a MSS of 1,460 bytes.

   (a) What is the maximum value of L such that sequence numbers are not exhausted? There are $2^{32} = 4,294,967,296$ possible sequence numbers. The sequence number does not increment by one with each segment. Rather, it increments by the number of bytes of data sent. So the size of the MSS is irrelevant – the maximum size file that can be sent from A to B is simply the number of bytes representable by $2^{32} \approx 4.19$ Gbytes .

   (b) For the L you have obtained above, find how long it takes to transmit the file over a 100Mbps link if there are 66 bytes of header added to each segment? Ignore flow and congestion control.
   The number of segments is $\lceil \frac{2^{32}}{1{,}460} \rceil = 2,941,759$ 66 bytes of header get added to each segment giving a total of 194,156,094 bytes of header. The total number of bytes transmitted is $2^{32} + 194156094 = 4,489,123,390$ bytes.

$$\frac{4,489,123,390 * 8}{100 \times 10^6} = 359.1298712 \text{ s}$$

8. (PD) Explain the three sequences of state transitions during a TCP connection tear-down.

   (a) This side closes first:
   ESTABLISHED→FIN_WAIT_1→FIN_WAIT_2→TIME_WAIT→CLOSED

   (b) The other side closes first:
   ESTABLISHED→CLOSE_WAIT→LAST_ACK→CLOSED

   (c) Both sides close at the same time :
   ESTABLISHED→FIN_WAIT_1→CLOSING→TIME_WAIT→CLOSED

9. (PD) Consider the following TCP connection teardown sequence: FIN-WAIT-1 to TIME-WAIT and labelled FIN+ACK/ACK. Explain the circumstances that result in this fourth teardown sequence.

Host A has sent a FIN segment to host B, and has moved from ESTABLISHED to FIN_WAIT_1. Host A then receives a segment from B that contains both the ACK of this FIN, and also Bs own FIN segment. This could happen if the application on host B closed its end of the connection immediately when the host As FIN segment arrived, and was thus able to send its own FIN along with the ACK.

10. Explain why during a TCP connection setup, sequence numbers usually will not start at 1.
    The starting sequence number is frequently determined by the SYN Cookie, which is not necessarily 1.