ECE 4380/6380: Computer Communications **Spring 2017**
Problem Set 3 **Due: 2:30 pm in class, Friday, February 17**

Assigned reading: Peterson and Davie, Chapter 2.1-2.5. All problems have equal weight.

1. **Example implementation of receiver's protocol for sliding window.** At the end of this assignment is a C code implementation of how the receiver operates with the sliding window protocol. In class we discussed the information that the receiver must keep track of and we defined `LFR` (largest frame received). The code segment below uses `NFE` (next frame expected), which is one greater than `LFR`.
   (a) In class we described two rules for determining when the receiver should send an acknowledgement. Approach one always sends an acknowledgement, and approach two suppresses acknowledgements in some circumstances. See slide "When Should Receiver Send ACK". For the code segment below which acknowledgement approach is used? Modify the code to implement the other approach. (Just state the lines you need to change or insert).
   (b) Assume a receiver gets the following frames at the following times. At *t*=1 frame 0 is received, at *t*=2 frame 1, *t*=3 frame 4, *t*=4 frame 5, *t*=6 frame 2, *t*=7 frame 3, *t*=8 frame 4. At each time describe what (if any) acknowledgement is sent, if the received frame is stored in the receiver's buffer or discarded, what frames are delivered to the application, and what `NFE` is equal to. Do this for both approaches for sending acknowledgments. Here is how the solution starts:

Time	**frame received**	**action with frames**	**NFE**	**Send ACK (approach 1)**	**ACK (approach 2)**
1	0	deliver 0 to app	1	0	
2	1	deliver 1 to app	2	1	
3	4	buffer 4, do not send to app	2	1	
4					

2. **Sliding Window Algorithms**. Consider three store-and-forward switches and two full-duplex point-to-point links. Node *A* is connected to node *B* by a 10 Mbps link and the propagation delay is 0.5 msec. Node *B* is connected to node *C* by a 10 Mbps link and the propagation delay is 1 msec. Node *A* sends 5,000 bit frames to node *C*, and node *C* replies to *A* with a 500-bit acknowledgement frame for each frame received. Assume that each frame generated at *A* contains 500 bits for the headers, and the remaining 4,500 bits are for payload.
   (a) Assume node *A* has an endless supply of frames to send to node *C*. What throughput can node *A* achieve if the stop-and-wait protocol is utilized? Assume that there are no errors and calculate the throughput in payload-bits per second.
   (b) Under the same situation, what is the maximum throughput that can be achieved if the sliding window protocol is utilized?
   (c) What is the smallest possible size for the sender's window that achieves the maximum throughput found in part (b)
   (d) For $RWS = \lfloor SWS/2 \rfloor$, construct an example demonstrating that SWS + 2 sequence numbers (e.g., from 0 to SWS + 1), where SWS is your answer to part (c) are not enough to guarantee correct operation of the sliding window algorithm.

3. **Sliding Window Algorithms**. Consider two full-duplex point-to-point links connecting host A, B, and C. This distance of the link between A and B is 2000 miles, and the distance of the link between

B and C is 500 miles. The propagation delay is 10 μsec/mile for both links. The link between A and B has a bandwidth of 100 kbps (you are not given the bandwidth for the link between B and C but will calculate it below). All data frames are 1000 bits long, but ACK frames are separate frames of negligible length. Between A and B a sliding-window protocol with window size of 3 is used, and Between B and C, a stop-and-wait protocol is used. Assume that there are no errors on the links. Notice that there is a reliable transmission protocol used on each link instead of an end-to-end reliable transmission protocol.

(a) What throughput (in bits per second) can be achieved on the link between A and B? (Ignore the link from B to C for this part).
(b) Frames are generated at node A and are relayed to node C by node B. Determine the minimum bandwidth in bits per sec required between nodes B and C so that the buffers of node B are not flooded. Hint: in order not to flood the buffers of B, the throughput entering node B must be less than or equal to the throughput leaving node B.
(c) For link A-B, the sliding window protocol with a window size of 3 limits the throughput. What is the smallest window size that should be used to maximize the throughput on the link between A and B? What problem occurs if this window size is used for the A-B link?

4. **Comparison of sliding window and concurrent logical channels**. Consider a full-duplex point-to-point link that connects two nodes. The transmission time for a frame is 1 second and the round trip time is 4 seconds. That is, if a node begins to transmit a frame at time *t*=0, it will receive the acknowledgment a *t*=4 seconds. The source node starts to transmit frames at time 0, and it has a total of 8 frames to send. For all parts of this problem assume that, due to errors on the channel, the transmission of the first frame is not received correctly, but that all other transmissions are successful. Define the *transfer time* as the time until the source nodes has determined that the destination node has successfully received all frames. The source sets a timer for each outstanding frame and if an acknowledgment is not received in 4 seconds the frame is retransmitted
(a) What is the transfer time for the stop-and-wait protocol?
(b) What is the transfer time if the sliding window protocol is used and both the sender's and receiver's window sizes are 4?
(c) What is the transfer time if the go-back-N protocol is used and the sender's window size is 4 and the receiver's window size is 1?
(d) What is the transfer time if 4 concurrent logical channels are used?

5. **Sliding Window with Relay**. Chapter 2, number 37

6. **ARQ .** Suppose that host *A* is connected to host *B* via an intermediate router *R*. The link between *A* and *R* is instantaneous, but the link between *R* and *B* transmits only one packet each second, one at a time. All links are full-duplex so that the two directions transmit independently. Hosts *A* and *B* run an ARQ protocol to control the exchange of packets between them. Assume that at time *t* = 0 host *A* has 7 frames to send to host *B.* Assume that the router has a queue size of 1; that is, it can hold one packet in addition to the one it is sending (in each direction). Let *A*'s timeout be 6 seconds.
(a) Assume the sliding window protocol with SWS=RWS=3 is utilized between host A and B. Using a diagram show the times that the first 7 frames are transmitted (or retransmitted) and when they are correctly received at host B. Also show what ACKs are sent. **(**Hint: to check your work the ACK for frame 7 is received at time 11).
(b) Now repeat the same scenario as in part (a) but instead use 3 concurrent logical channels. Provide a diagram like for part (a). **(**Hint: to check your work the final ACK is received at time 8, and it is not for frame 7!).

7. **Sequence numbers.**

(a) In a sliding window protocol with RWS=SWS=5, a very large set of possible sequence numbers (assume no wrapping), and in-order packet arrivals, why can the receiver be assured that it will never again receive the frame with sequence number 10 if it is currently expecting frame 17?

(b) For the sliding window protocol, if the sender's window size (SWS) equals the receiver's window size (RWS), what is the minimum number of sequence numbers that are required? How many bits must be reserved in the packet header for sequence numbers?

(c) For the go-back-N protocol with N, the send window size, equal to 8, what is the smallest number of sequence numbers that are needed to ensure reliable delivery of the frames?

Example sliding window code for receiver.

```
#define RWS         8   /* receive window size */
#define NUM_SEQ_NO 16   /* max. sequence number+1 */
                        /* (must be multiple of */
                        /* RWS for this code) */
#define FRAME_SIZE 1000 /* constant for simplicity */

char buf[RWS][FRAME_SIZE]; /* RWS frame buffers */
int present[RWS];        /* are frame buffers full? */
                         /* (initialized to 0's) */
int NFE = 0;             /* next frame expected */

extern void send_ack (int seq_no);
extern void pass_to_app (char* data);
void recv_frame (char* data, int seq_no);

void
recv_frame (char* data, int seq_no)
{
    int idx;      /* index into data structures */
    int i;        /* loop index */
    /* Map sequence numbers NFE...predecessor (NFE)
       into 0...NUM_SEQ_NO - 1, then see if seq_no
       falls within the receive window. */
    if (((seq_no + (NUM_SEQ_NO - NFE)) % NUM_SEQ_NO)
          < RWS) { /* Frames outside the window */
                   /* are discarded */
      /* Calculate index into data structures. */
      idx = (seq_no % RWS);
      if (!present[idx]) {   /* frame is not duplicate */
          present[idx] = 1;  /* mark as received */
          memcpy (buf[idx], data, FRAME_SIZE);
                             /* copy data into buffer */

          /* Got a new frame; pass frames up to host? */
          for (i = 0; i < RWS; i++) {
            idx = (i + NFE) % RWS; /* Re-use idx. */
            /* The first missing frame becomes NFE */
            /* after this loop terminates. */
            if (!present[idx])
                break;
            /* Frame is present-send it up! */
            pass_to_app (buf[idx]);
            /* Mark frame buffer as empty. */
            present[idx] = 0;
          }
          /* Advance NFE to first missing frame. */
          NFE = (NFE + i) % NUM_SEQ_NO;
      }
      /* Frame handled (might have been a duplicate). */
    }
    /* Now send acknowledgement for predecessor frame (NFE). */
    send_ack ((NFE + NUM_SEQ_NO - 1) % NUM_SEQ_NO);
}
```

(This code is posted on Canvas if you would like to experiment. See slidewin.c)