ECE 4380/6380: Computer Communications **Spring 2018**
Problem Set 3 Solutions

Each problem is worth 10 points

1. **Example implementation of receiver's protocol for sliding window**.
   (a) The implementation given here sends an acknowledgment for each frame received. There are many possible ways to change the code so that an acknowledgment is sent except for frames buffered and received out of order. Here is one possible change:

```
20.    int i = -1;         /* loop index */
```

Change line 52 to the following

```
52.    if (i != 0)
           send_ack ((NFE + NUM_SEQ_NO - 1) % NUM_SEQ_NO);
```

If the received frame is out of the window then the frame is not buffered and the variable *i* is never changed from its initial value of –1, and an acknowledgment should be sent. If the frame is in the window and equal to NFE, then the for loop will deliver (at least) this frame to the application and *i* is first set to zero and incremented at least once. If the received frame is in the receiver's window but cannot be delivered to the application because it is out of order then the for loop exits with $i = 0$, and this is the situation in which an acknowledgment does not need to be sent.

(b) Here is the state of the receiver after processing each received frame

Time	frame received	action with frames	NFE	ACK
1	0	buffer 0, deliver 0 to app	1	0
2	1	buffer 1, deliver 1 to app	2	1
3	4	buffer 4, nothing to app	2	1 (no ack for approach 2)
4	5	buffer 5, nothing to app	2	1 (no ack for approach 2)
6	2	buffer 2, deliver 2 to app	3	2
7	3	buffer 3, deliver 3, 4, 5 to app	6	5
8	4	discard 4 (outside window)	6	5

A key point to notice is that at time t=7 when frame 3 is received, the acknowledgement returned is equal to 5 **not** 3. The acknowledgment is **cumulative** and indicates to the transmitter that frames 3, 4, and 5 have all been accepted.

2. **Sliding Window Algorithms**.
   (a) First find the round trip delay, which is the sum of transmission time and propagation delay on each link

   Tx time for packet $A \rightarrow B$ and $B \rightarrow C$ = 5,000 bits / 10 Mbps = 500 μs
   Tx time for ACK $C \rightarrow B$ and $B \rightarrow A$ = 500 bits / 10 Mbps = 50 μs
   Propagation delay $A \leftrightarrow B$ = 500 μs
   Propagation delay $B \leftrightarrow C$ = 1000 μs
   The total delay is (500 + 500 + 50 + 50) + (1000 + 500 + 1000 + 500) = 4.1 ms. The throughput is the total payload bits delivered divided by the time taken = 4,500 bits / 4,100 μs = 1.1 Mbps.
   (b) With the sliding window protocol we can transmit frames continuously. Each frame takes 500 μs to transmit and delivers 4,500 payload bits. The throughput is 4,500 bits / 500 μs = 9 Mbps.

(c) Find out how many frames the source can transmit in one RTT. 4,100 μs / 500 μs = 8.2 frames. To keep the pipe full must transmit at least 8.2 frames per RTT. So, the minimum window size is 9.

(d) The RWS = 4 and the SWS = 9. The number of sequence numbers is 11 and the sequence numbers are 0 to 10.

frame number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
sequence number	0	1	2	3	4	5	6	7	8	9	10	0	1	2	3

Assume the sender transmits frames 1 through 9 (with sequence numbers 0 through 8) and all the frames are successfully received. The receiver advances its receive window to expect sequence numbers [9, 10, 0, 1], and sends the ACK's. However, assume that all of the ACK's are lost. The sender times out and resends frames 1 through 9 (numbered 0 through 8). The receiver stores the frames numbered 0 and 1, thinking they correspond to the 12^{th} and 13^{th} frames, and it ignores the other frames because they are outside the receiver's window. The receiver transmits ACK's each with sequence number 8. Assume the ACK's are successful this time. The sender now transmits its 10^{th} and 11^{th} frame. The receiver accepts these frames and incorrectly assumes that it already has the 12^{th} and 13^{th} frames even through the sender has not sent them yet. If we used 13 instead of 11 sequence numbers this problem would not occur, and the receiver could always distinguish between new and duplicate frames.

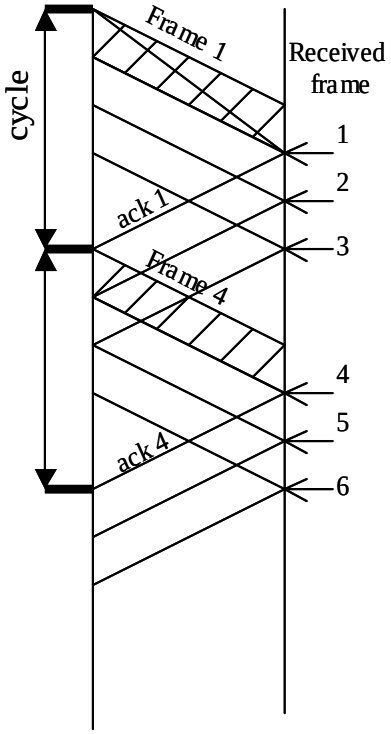
3. **Sliding Window Algorithms**.

(a) For the link between A and B we need the transmission time of a frame and the propagation delay. The transmission time for one frame is

Tx = (1000 bits) / (100,000 bits per sec) = 10 ms

Prop delay (one-way) = 2000 miles × 10 μs per mile = 20 ms

As shown in the figure below, if host A starts to transmit frame 1 at time 0, the acknowledgement for frame 1 will arrive at time 50 ms. While we have assumed that the transmission time for the acknowledgment packet is negligible, the propagation delay is not. Even if the acknowledgment packet is only a few bits, the bits cannot travel faster than the speed of light (or in this problem 10 μs per mile).



To calculate the throughput, we need to determine how often the transmitter can start a group of three frames. Note that the transmitter can begin sending frame 4 after receiving the ack for frame 1. So, the cycle time is 2 × prop delay + tx time for one frame = 2×20 + 10 = 50 ms. In one cycle host A can transmit 3 frames or 3,000 bits. So, the throughput = 3000 bits / 50 ms = 60 Kbps.

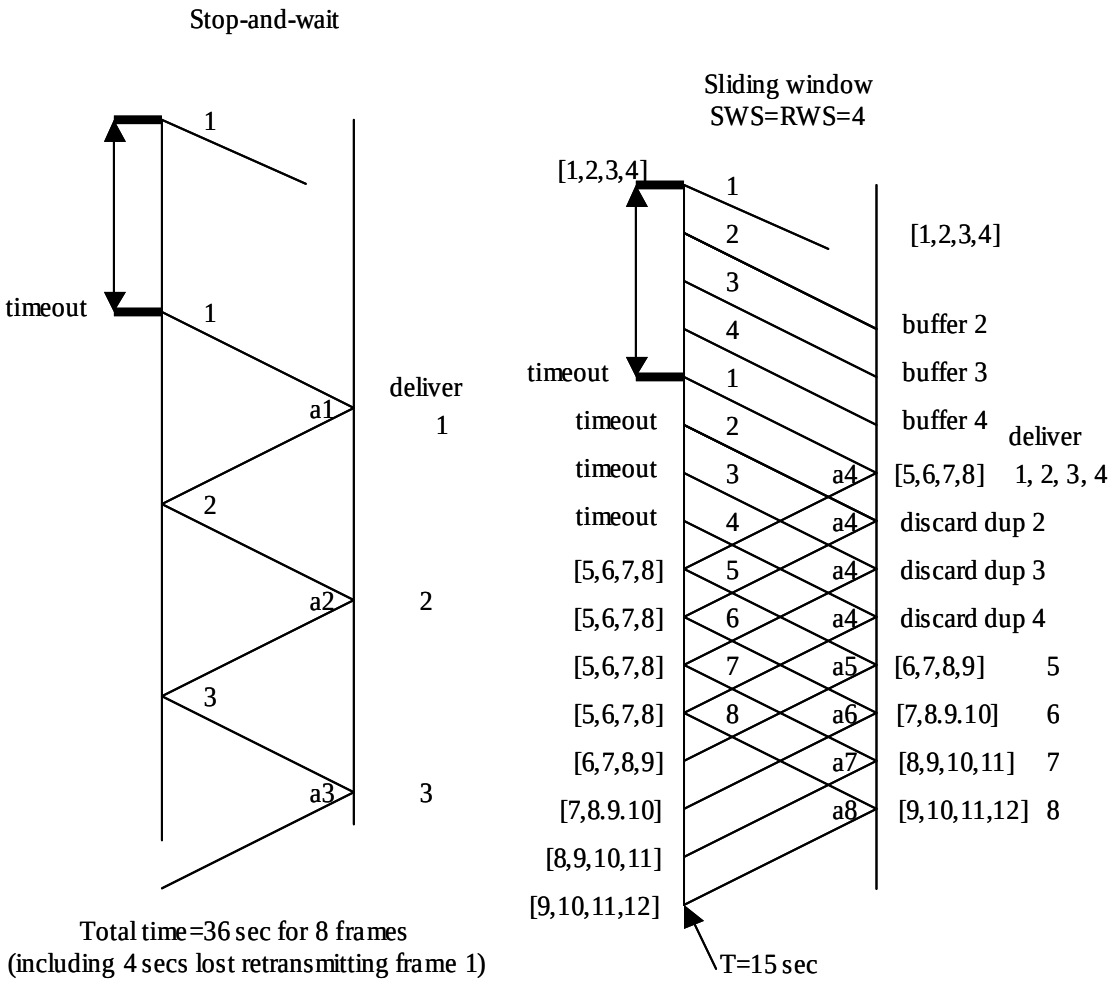
(b) For the link between host B and C, the propagation delay = 500 miles × 10 μs per mile = 5 ms (one-way). For stop-and-wait only one frame can be sent per cycle, and the cycle time is equal to 2 × prop delay + tx time = 2 × 5 ms + 1000 bits / *R* bits per sec, where *R* bps is the required bandwidth for the link between hosts B and C. Hence, the throughput for the stop-and-wait link is 1000 bits / cycle time = 1000 bits / (0.01 + 1000 / *R* seconds). We found the throughput for the A-B link (using a sliding window) to be 60 Kbps. To avoid buffer overflow we need the throughput on the two links to be equal, or

$$60{,}000 = 1000 / (0.01 + 1000/R)$$

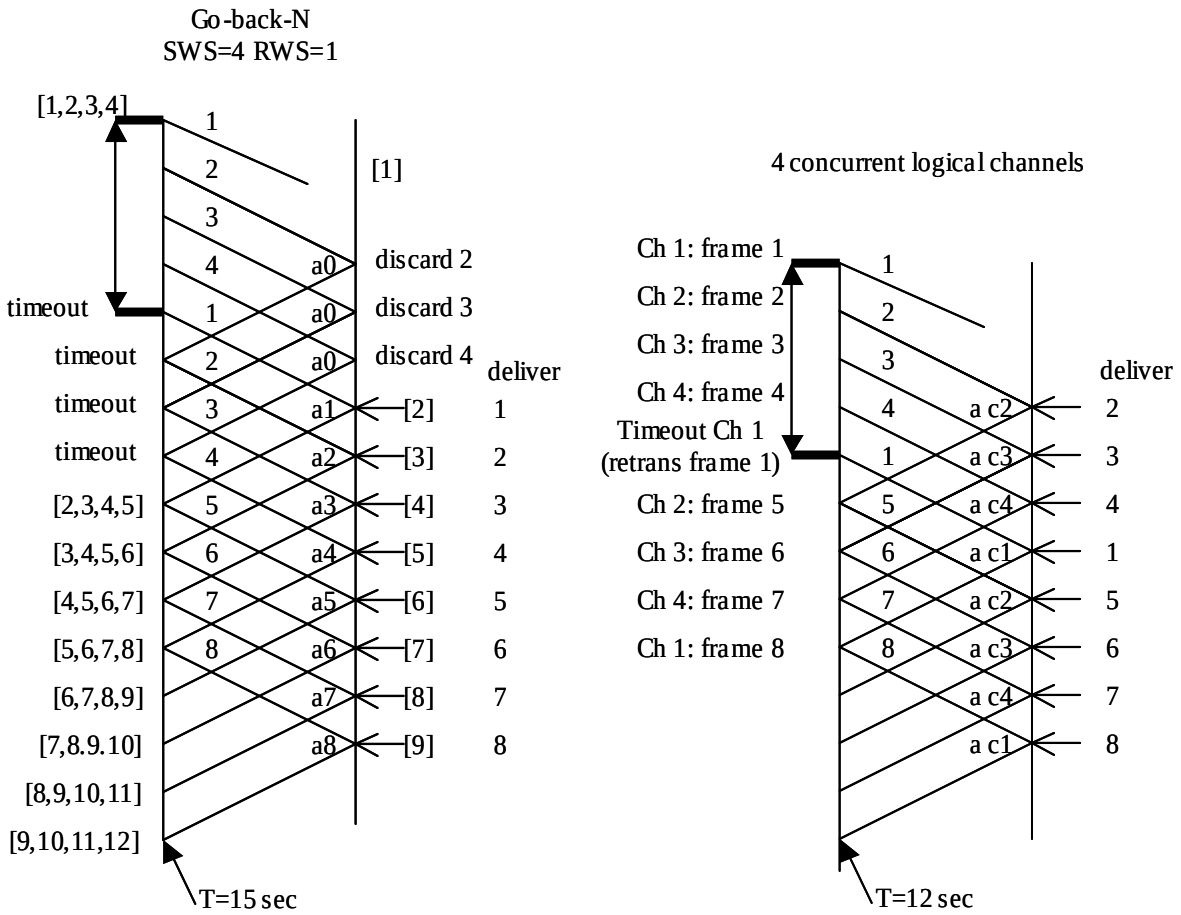
Solving for *R*, we obtain *R* = 150,000 bits per sec.

(c) For the A-B link, the pipe is kept full by continuously transmitting during the 50 ms required to transfer one frame. In 50 ms host A can transmit 100 Kbps × 50 ms = 5000 bits. Thus, the window size should be 5000 bits / 1000 bits per frame = 5 frames. Note that in this case, the throughput on the link A-B is at its maximum of 100 Kbps. The difficulty occurs in the link from B to C. We found that the throughput for this link (using stop-and-wait) is 1000 bits / (0.01 + 1000 / *R* seconds). However, when we set this equal to 100 Kbps we find that a solution does not exist for *R*. In other words, no matter how large the bandwidth, the throughput cannot reach 100 Kbps using stop-and-wait on the link B to C (because of the propagation delay).

4. **Sliding Window Algorithms and concurrent logical channels**.
Figures below show the sequence of packet transmissions for the four schemes. I have ignored sequence number wrap around in these figures. While the sliding window and go-back-N approaches take the same amount of time in this example, notice the difference in how the acknowledgments are generated and the windows are handled.

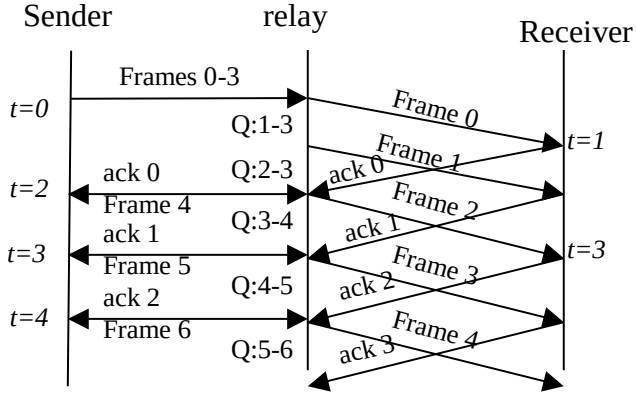


Note that the figure for the sliding window protocol assumes that acknowledgments are not sent for frames that are received in the window but out of order. This is "approach 2" as discussed in class.



In this scenario the sliding window and go-back-N schemes take the same time. The transfer times are the same because we can perfectly set the timeout to equal to the round trip time. If there is some variability in the round trip time, we would need to make a more conservative choice for the timeout (to avoid unnecessary retransmissions). In this example, if the timeout is set to 8 seconds but the round trip time is still 4 seconds, then we could set the SWS=8. Repeating the above scenario (but with 12 packets total) you find that the sliding window protocol is faster than go-back-N because it can avoid retransmitting some frames.

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



t=0 A sends frames 0-3. Frame 0 starts from relay to receiver. Frames 1-3 queued at relay

t=1 Frame 0 arrives at receiver, Acknowledgment 0 starts back. Frame 1 leaves relay. Frames 2-3 queued at relay

t=2 ack 0 arrives at sender, then frame 4 goes to relay. Frame 2 leaves relay for receiver, frames 3-4 queued at relay. Frame 1 arrives at receiver, Ack 1 leaves.

t=3 ack 1 arrives at sender, then frame 5 goes to relay. Frame 3 leaves relay for receiver, frames 4-5 queued at relay. Frame 2 arrives at receiver, ack 2 leaves

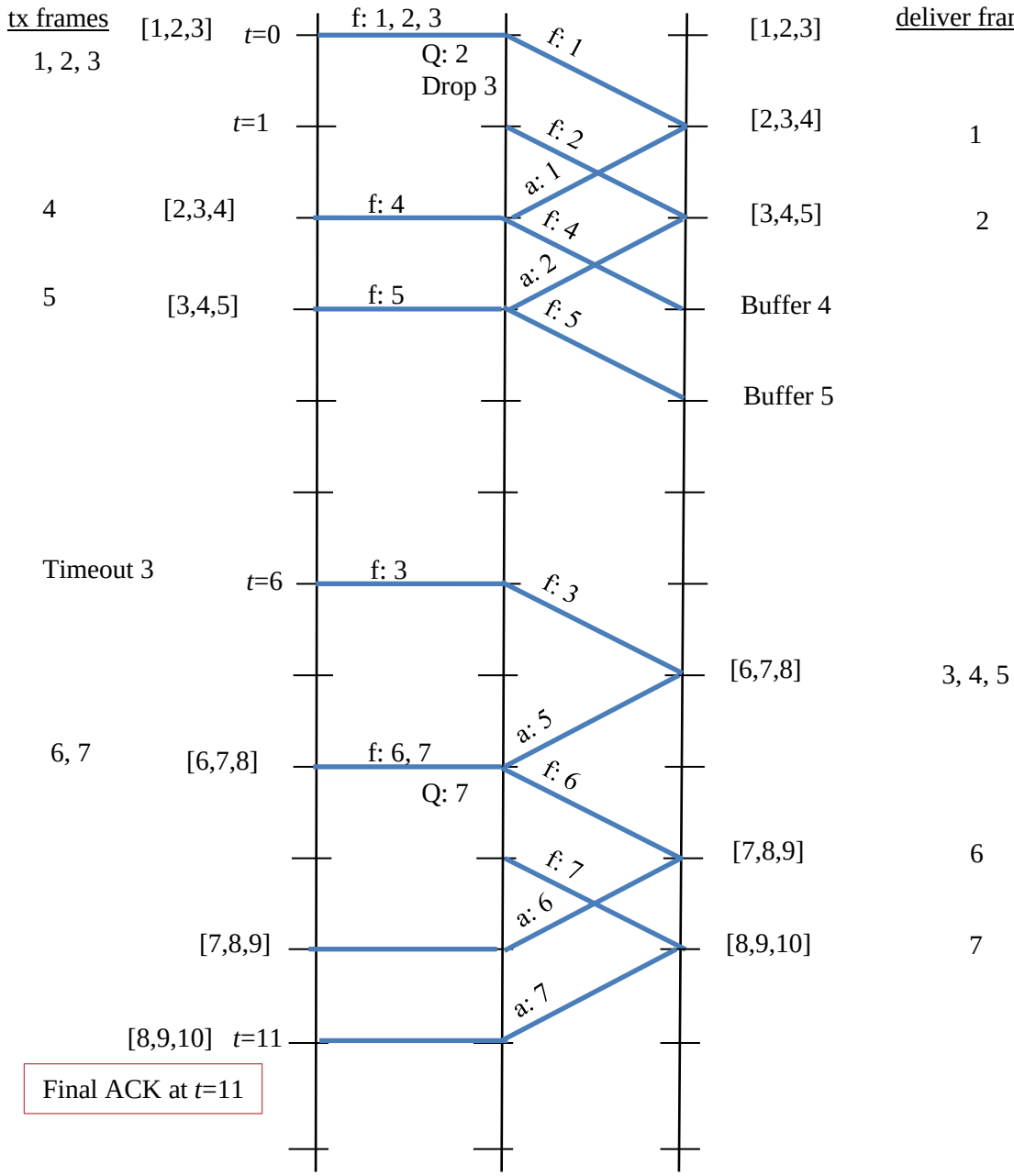
t=4 ack 2 arrives at sender, then frame 6 goes to relay. Frame 4 leaves relay for receiver, frames 5-6 queued at relay. Frame 3 arrives at receiver, ack 3 leaves

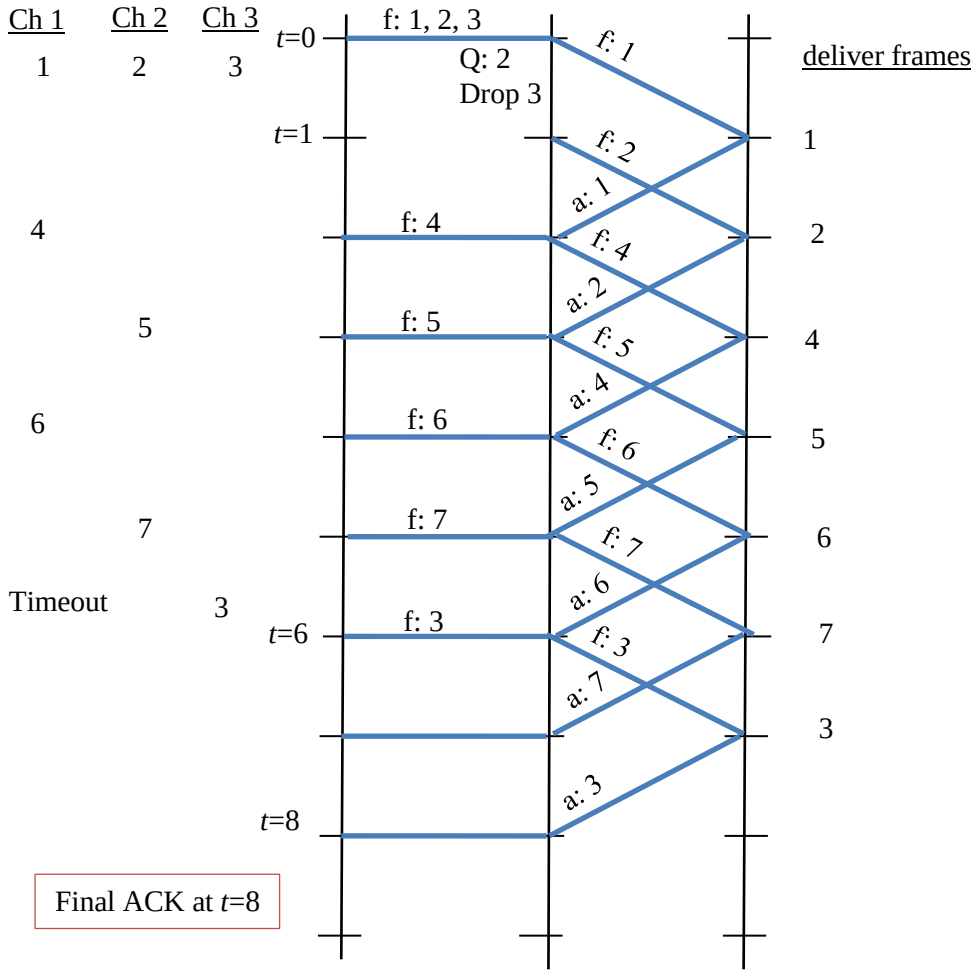
In steady-state the queue size at the relay is two frames, though there is a peak requirement to buffer three frames at time 0.

Notice that once the system settles (after time 2), the sender is allowed to transmit one frame per time unit. There are no retransmissions or timeouts. This keeps the pipe between the relay and the receiver full. Also, notice each frame waits 2 time units in the queue (starting with frame 4).

6. **ARQ**.

An important point to notice about this problem is related to the sender's window size. The reason there are some retransmissions is because the sender's window is too large. This allows the queue at the relay to overflow. If the sender's window size is 2 instead of 3, then the queue will not overflow and the pipe between the relay and receiver will be kept full. This problem demonstrates a scenario in which a window size that is too large can lead to underutilization of the network.


tx frames
deliver frames
1, 2, 3
[1,2,3]
t=0
f: 1, 2, 3
Q: 2
Drop 3
f: 1
[1,2,3]
t=1
f: 2
[2,3,4]
1
4
[2,3,4]
f: 4
a: 1
f: 4
[3,4,5]
2
5
[3,4,5]
f: 5
a: 2
f: 5
Buffer 4
Buffer 5
Timeout 3
t=6
f: 3
f: 3
[6,7,8]
3, 4, 5
6, 7
[6,7,8]
f: 6, 7
Q: 7
a: 5
f: 6
[7,8,9]
6
f: 7
a: 6
[7,8,9]
[8,9,10]
7
a: 7
[8,9,10]
t=11
Final ACK at t=11



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?

The receiver is expecting frame 17, and its window contains [17, 18, 19, 20, 21]. The sender's window can start no earlier than [12, 13, 14, 15, 16], and this scenario occurs only if the receiver correctly decoded frames 12 through 16, but all acknowledgements were lost, requiring retransmissions of frames 12 through 16. However, for the receiver's window to advance to 17, frame 10 (and frame 11) must have been successfully acknowledged. Frames 10 and 11 cannot be in the sender's window.

(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?

The number of sequence numbers is simply SWS + RWS, and the number of bits required in the header is equal to $\lceil \log_2(SWS + RWS) \rceil$

(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?

Nine sequence numbers are required, since the receiver's window size is one.