Alvin Nye

CMPS 2301 Introduction to Computer Systems & Networking

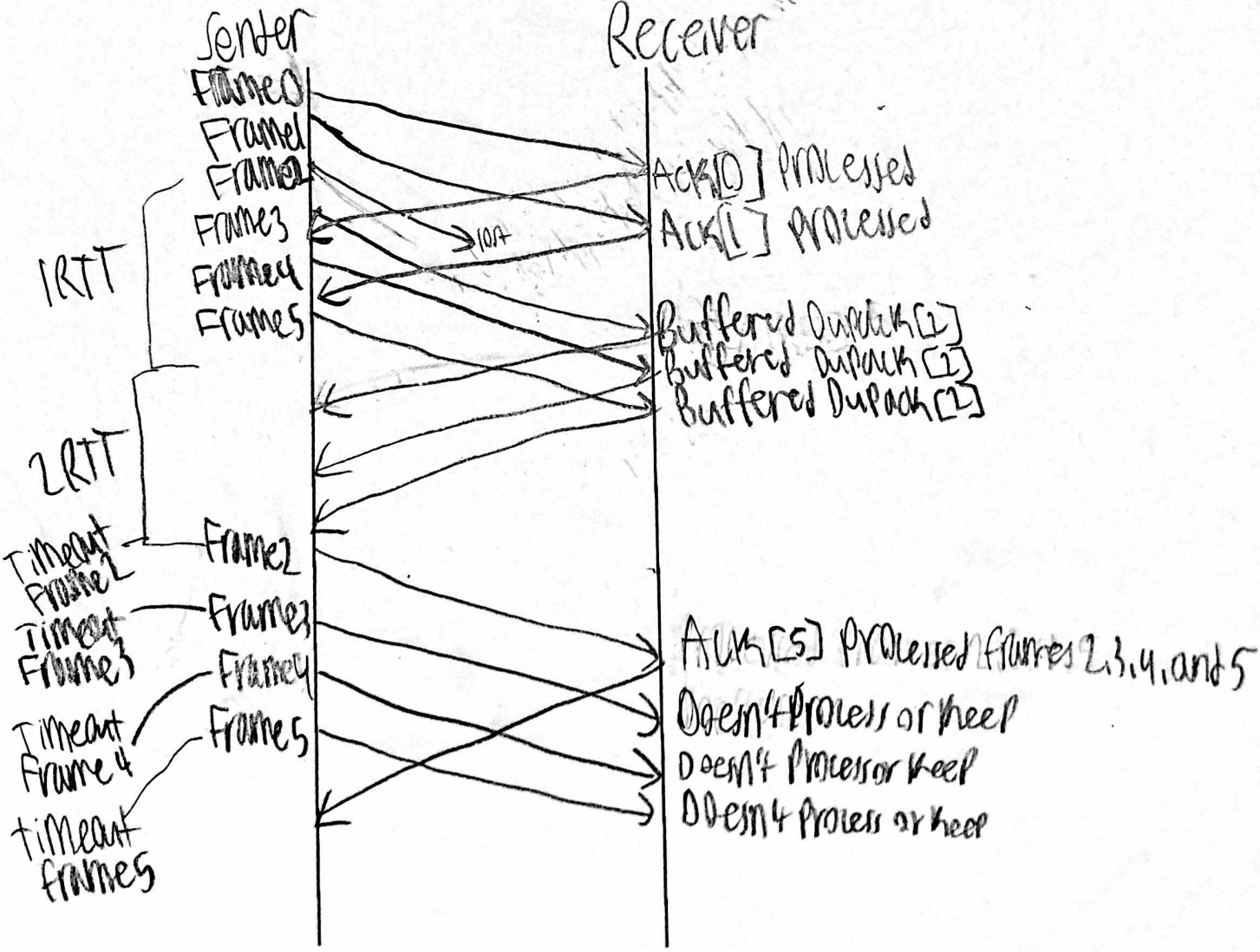
Lab 10: Sliding Window/Bridging

1 Reliable Transmission (12 pts)

1) Consider a variation of the Sliding Window algorithm. Assume the receiver sends a duplicate acknowledgment if it does not receive the expected frame. For example, it sends DUP_ACK[2] when it expects to see Frame[2] but receives Frame[3] instead. Also, the receiver sends a cumulative acknowledgment after it receives all the outstanding frames. For example, it sends ACK[5] when it receives the lost frame Frame[2] after it already received Frame[3], Frame[4], and Frame[5]. Use a timeout interval of about 2 ×RTT.

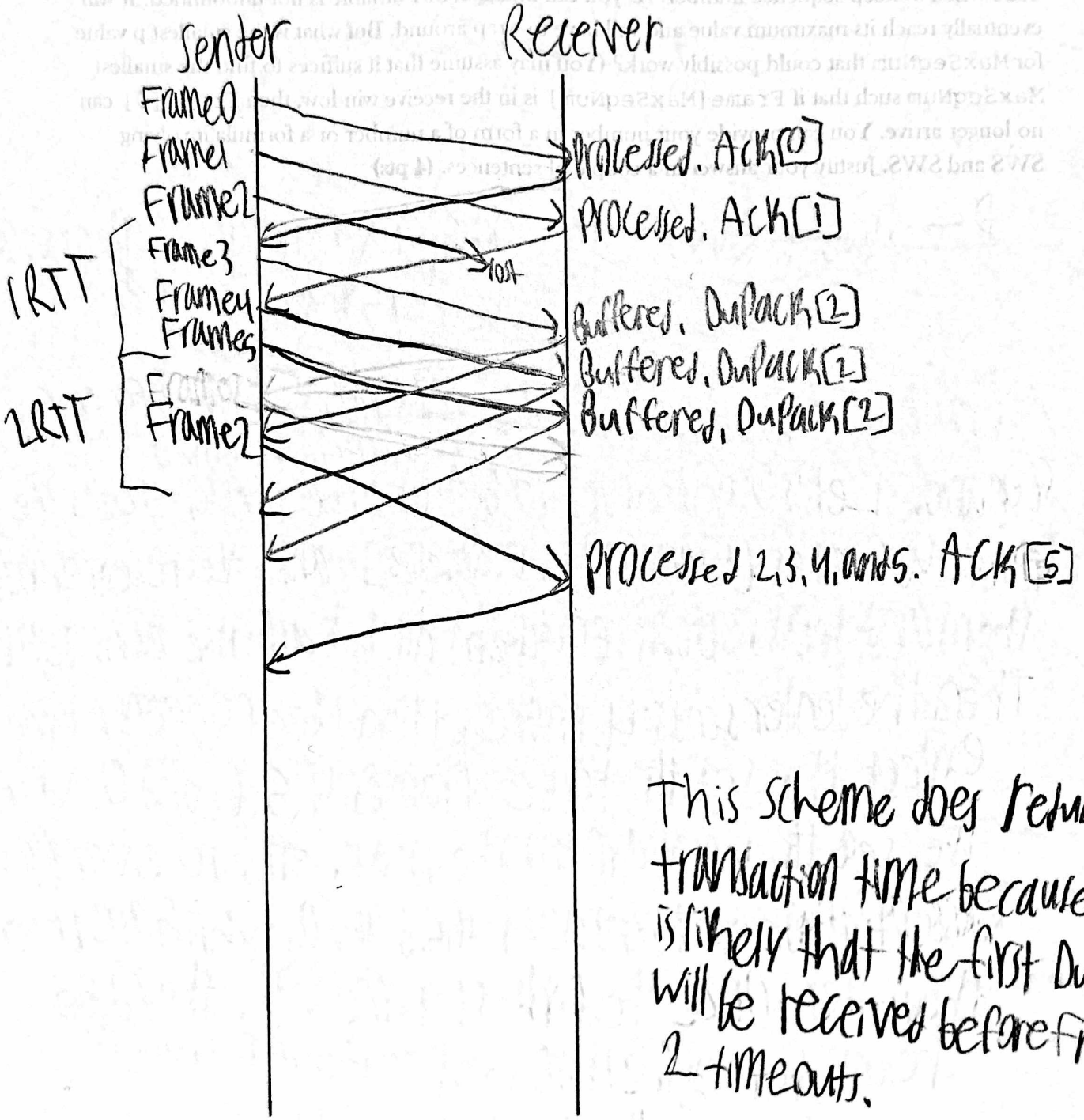
Draw a timeline diagram for the sliding window algorithm with SWS = RWS = 4 frames in the following two situations.

a) Frame 2 is lost. Retransmission takes place upon timeout (as usual). **(4 pts)**



Lab 10: Sliding Window/Bridging

b) Frame 2 is lost. Retransmission takes place either upon receipt of the first DUPACK or upon timeout. Does this scheme reduce the transaction time? (Note that some end-to-end protocols, such as variants of TCP, use similar schemes for fast retransmission.) **(4 pts)**



This scheme does reduce transaction time because it is likely that the first DupACK[2] will be received before frame 2 timeouts.

CMPS 2301 Introduction to Computer Systems & Networking

Lab 10: Sliding Window/Bridging

2) Suppose that we run the sliding window algorithm with `SWS = 5` and `RWS = 3`, and no out-of-order arrivals. As usual, each frame is marked with its sequence number, in order to let the receiver detect undelivered frames, and also order them. The sender maintains a counter variable that is used to keep sequence number. As you can imagine, this variable is not unbounded. It will eventually reach its maximum value and will have to wrap around. But what is the smallest p value for `MaxSeqNum` that could possibly work? (You may assume that it suffices to find the smallest `MaxSeqNum` such that if `Frame[MaxSeqNum]` is in the receive window, then `Frame[0]` can no longer arrive. You can provide your number in a form of a number or a formula involving SWS and SWS. Justify your answer in a couple of sentences. **(4 pts)**

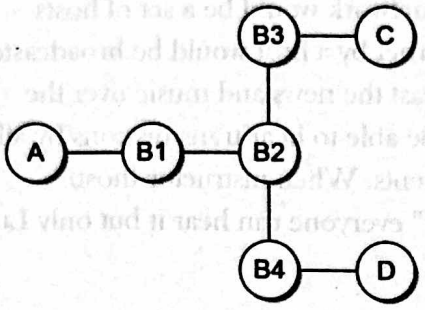
P = SWS + RWS - 1 . So for our example that would be P = 5+3-1 = 7

7 would be the smallest possible MaxSeqNum value for our scenario. Let's say it is set to 6 (which is less than my min) and the sender sends the first 4 frames (Frame[0]... Frame[3] and the receiver accepts them and sends acks for them all, but all the acks get lost. Then the sender sends 4 more, then the receiver would expect the seq #s to be frames 4, 5, 6, and 0 since the seq #s wrap around the max. The receiver would accept these and send acks 4, 5, 6, and 0, which are received by the sender. Once the first 4 time out, they'd be resent, but something would go wrong because the receiver will accept this frame 0, but it's the wrong one. Thus, you'd need 1 more seq number to avoid this issue. Therefore, 7 must be the smallest MaxSeqNum.

Lab 10: Sliding Window/Bridging

table, forward the frame out on all other ports. Strategy works fine if the extended LAN does not have a loop in it. If there is a loop, frames potentially loop through the extended LAN forever.

3) Consider the following arrangement of learning bridges:



Assuming all are initially empty, give the forwarding tables for each of the bridges B1 to B4 after the following transmissions:

```
A sends to C.  C sends to A. D sends to C.
```

Each forwarding table will maintain **<destination_address, interface>** pairs. Mark ports/interfaces with the unique neighbor reached directly from that port; that is, the ports for B1 are to be labeled "A" and "B2" and use these values in the forwarding table. **(4 pts)**

B1 Forwarding Table

Dest Address	Interface
A	A Interface
C	B2 Interface
D	B2 Interface

B2 Forwarding Table

Dest Address	Interface
A	B1 Interface
C	B3 Interface
D	B4 Interface

B3 Forwarding Table

Dest Address	Interface
A	B2 Interface
C	C Interface
D	B2 Interface

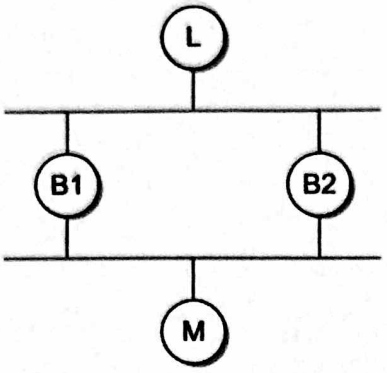
B4 Forwarding Table

Dest Address	Interface
A	B2 Interface
C	B2 Interface
D	D Interface

CMPS 2301 Introduction to Computer Systems & Networking

Lab 10: Sliding Window/Bridging

4) Suppose learning bridges B1 and B2 form a loop:



Also assume that they do **not** implement the loop detection algorithm. Each bridge maintains a single table of **<address, interface>** pairs. Provide a short explanation describing what will happen if M sends to L? **(4 pts)**

When M tries to send to L, both B1 and B2 will pickup the packet and add <M, M interface> to each of their forwarding tables. Then, B1 and B2 will compete to transmit the packet to L at the same time. Let's say B1 gets its packet to L first. Then, L will send this packet to B2. B2 will see this packet, which is a duplicate of one it has already seen, but it will think that this packet had to go through L interface to get to it, so it now has wrong information about where M is since B2 probably doesn't even realize that it was a duplicate. B2 will then send this packet back to M and this process will start all over again.

Submission

Upload your completed version of this lab to canvas.