Total points = 10

Question 1 (1 point)

In our solution, the sender will wait until it receives an ACK for a pair of messages (seqnum and seqnum+1) before moving on to the next pair of messages. Data packets have a data field and carry a two-bit sequence number. That is, the valid sequence numbers are 0, 1, 2, and 3. (Note: you should think about why a 1-bit sequence number space of 0, 1 only would not work in the solution below.) ACK messages carry the sequence number of the data packet they are acknowledging.

The FSMs for the sender and receiver are shown in Figure 1 and Figure 2 respectively. Note that the sender state records whether (i) no ACKs have been received for the current pair, (ii) an ACK for seqnum (only) has been received, or an ACK for seqnum+1 (only) has been received. In this figure, we assume that the seqnum is initially 0, and that the sender has sent the first two data messages (to get things going). A timeline trace for the sender and receiver recovering from a lost packet is shown below:

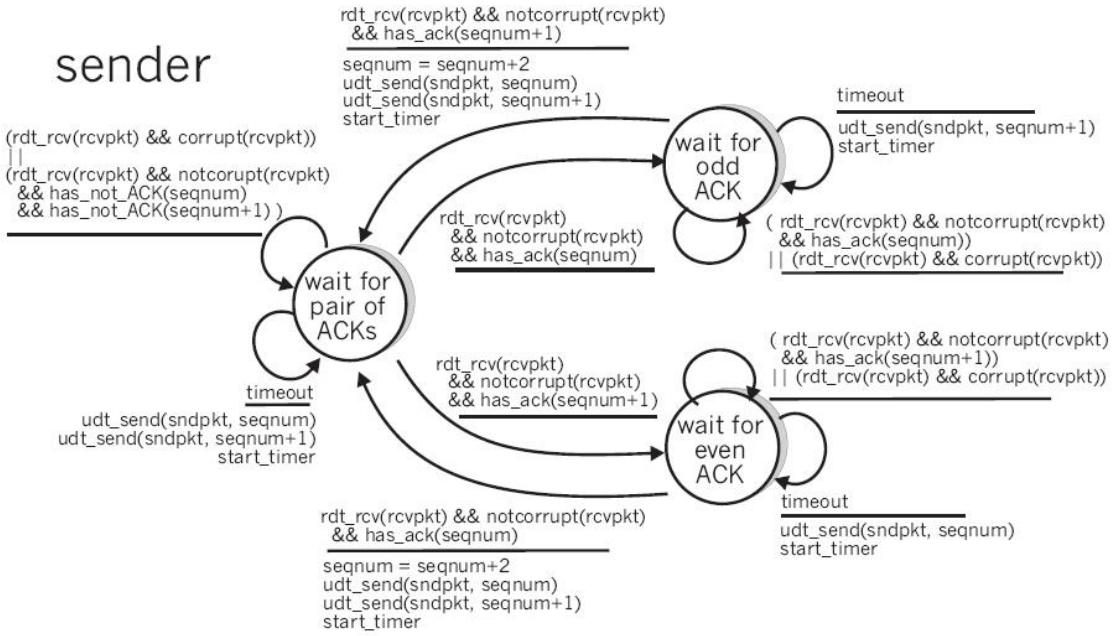

Figure 1: Sender FSM for Question 1

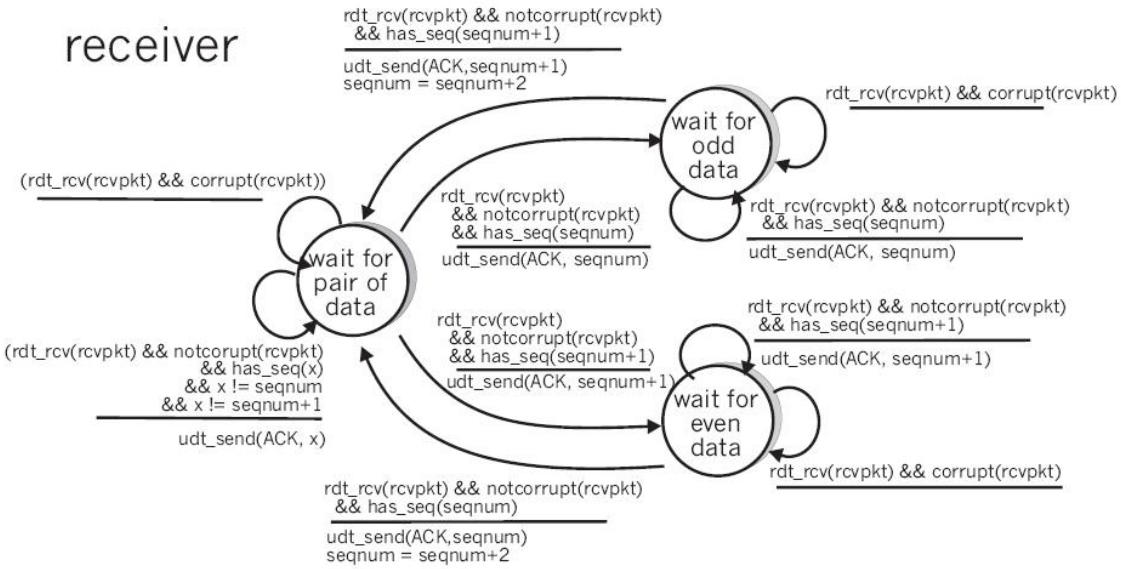

Figure 2: Receiver FSM for Question 1

```
Sender                              Receiver
make pair (0,1)
send packet 0
                    packet 0 drops
send packet 1
                                    receive packet 1
                                    buffer packet 1
                                    send ACK 1
receive ACK 1
(timeout)
resend packet 0
                                    receive packet 0
                                    deliver pair (0,1)
                                    send ACK 0
receive ACK 0
```

Question 2 (1 point)

This problem is a variation on the simple stop and wait protocol (rdt3.0). Because the channel may lose messages and because the sender may resend a message that one of the receivers has already received (either because of a premature timeout or because the other receiver has yet to receive the data correctly), sequence numbers are needed. As in rdt3.0, a 0-bit sequence number will suffice here.

The sender and receiver FSMs are shown in Figure 3 and Figure 4 respectively. In this problem, the sender state indicates whether the sender has received an ACK from B (only), from C (only) or from neither C nor B. The receiver state indicates which sequence number the receiver is waiting for.

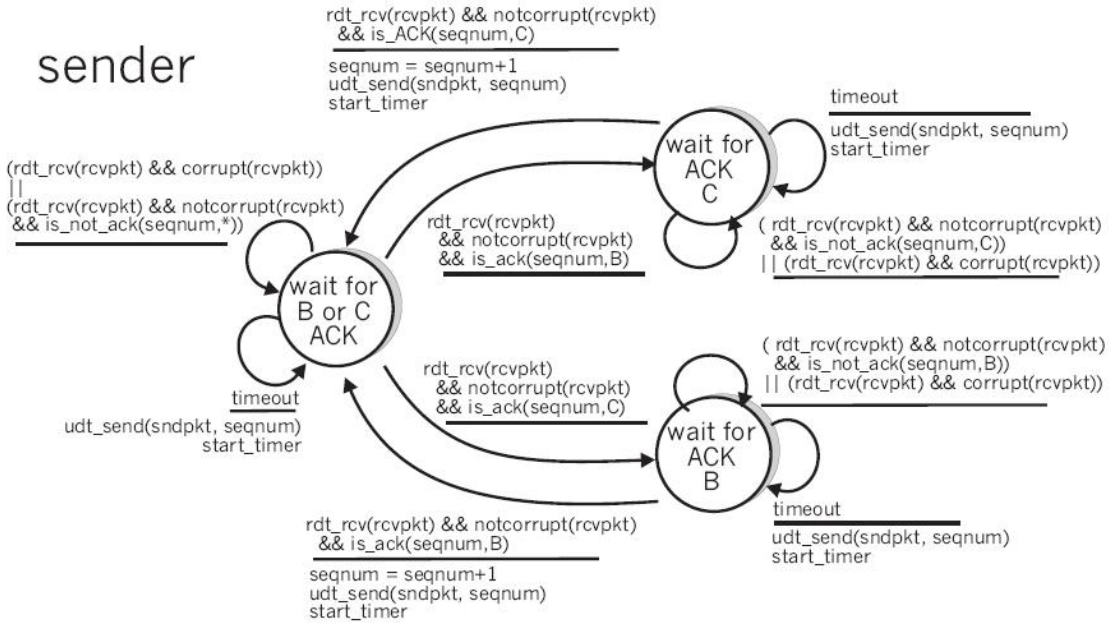

Figure 3: Sender FSM for Question 2

Question 3 (2 points)

Because the A-to-B channel can lose request messages, A will need to timeout and retransmit its request messages (to be able to recover from loss). Because the channel delays are variable and unknown, it is possible that A will send duplicate requests (i.e., resend a request message that has already been received by B). To be able to detect duplicate request messages, the protocol will use sequence numbers. A 1-bit sequence number will suffice for a stop-and-wait type of request/response protocol. A (the requestor) has 4 states:

- "Wait for Request 0 from above". Here the requestor is waiting for a call from above to request a unit of data. When it receives a request from above, it sends a request message, R0, to B, starts a timer and makes a transition to the "Wait for D0" state. When in the "Wait for Request 0 from above" state, A ignores anything it receives from B.
- "Wait for D0". Here the requestor is waiting for a D0 data message from B. A timer is always running in this state. If the timer expires, A sends another R0 message, restarts the timer and remains in this state. If a D0 message is received from B, A stops the time

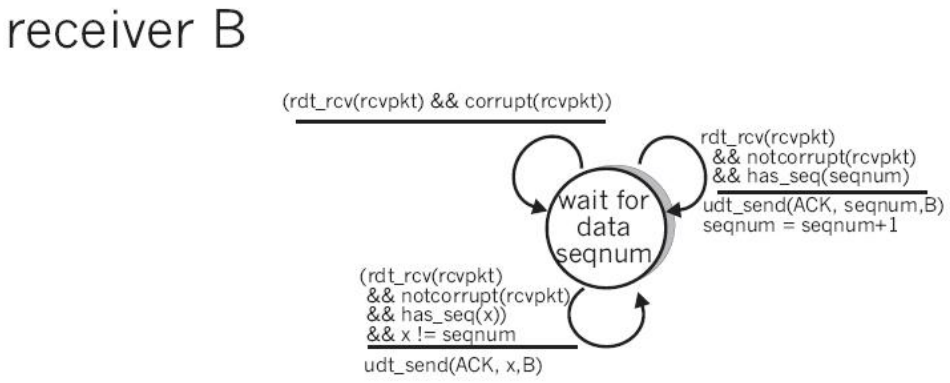

Figure 4: Receiver FSM for Question 2

and transits to the "Wait for Request 1 from above" state. If A receives a D1 data message while in this state, it is ignored.

- "Wait for Request 1 from above". Here the requestor is again waiting for a call from above to request a unit of data. When it receives a request from above, it sends a request message, R1, to B, starts a timer and makes a transition to the
- "Wait for D1" state. When in the "Wait for Request 1 from above" state, A ignores anything it receives from B.
- "Wait for D1". Here the requestor is waiting for a D1 data message from B. A timer is always running in this state. If the timer expires, A sends another R1 message, restarts the timer and remains in this state. If a D1 message is received from B, A stops the timer and transits to the "Wait for Request 0 from above" state. If A receives a D0 data message while in this state, it is ignored.

The data supplier (B) has only two states:

- "Send D0". In this state, B continues to respond to received R0 messages by sending D0, and then remaining in this state. If B receives a R1 message, then it knows its D0 message has been received correctly. It thus discards this D0 data (since it has been received at the other side) and then transits to the "Send D1" state, where it will use D1 to send the next requested piece of data.
- "Send D1". In this state, B continues to respond to received R1 messages by sending D1, and then remaining in this state. If B receives a R1 message, then it knows its D1 message has been received correctly and thus transits to the "Send D1" state.

Question 4 (2 points)

If the arrival rate increases beyond $R/2$ in Figure 3.45(b), then the total arrival rate to the queue exceeds the queue's capacity, resulting in increasing loss as the arrival rate increases.

When the arrival rate equals $R/2$, 1 out of every three packets that leaves the queue is a retransmission. With increased loss, even a larger fraction of the packets leaving the queue will be retransmissions. Given that the maximum departure rate from the queue for one of the sessions is $R/2$, and given that a third or more will be transmissions as the arrival rate increases, the throughput of successfully deliver data can not increase beyond λ_{out}. Following similar reasoning, if half of the packets leaving the queue are retransmissions, and the maximum rate of output packets per session is $R/2$, then the maximum value of λ_{out} is $(R/2)/2$ or $R/4$.

Question 5 (2 points)

a) TCP slowstart is operating in the intervals [1,6] and [23,26]

b) TCP congestion advoidance is operating in the intervals [6,16] and [17,22]

c) After the 16th transmission round, packet loss is recognized by a triple duplicate ACK. If there was a timeout, the congestion window size would have dropped to 1.

d) After the 22nd transmission round, segment loss is detected due to timeout, and hence the congestion window size is set to 1.

e) The threshold is initially 32, since it is at this window size that slowtart stops and congestion avoidance begins.

f) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 16, the congestion windows size is 42. Hence the threshold is 21 during the 18th transmission round.

g) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 16, the congestion windows size is 42. Hence the threshold is 21 during the 18th transmission round.

h) During the 1st transmission round, packet 1 is sent; packet 2-3 are sent in the 2nd transmission round; packets 4-7 are sent in the 3rd transmission round; packets 8- 15 are sent in the 4th transmission round; packets15-31 are sent in the 5th transmission round; packets 32-63 are sent in the 6th transmission round; packets 64 – 96 are sent in the 7th transmission round. Thus packet 70 is sent in the 7th transmission round.

i) The congestion window and threshold will be set to half the current value of the congestion window (8) when the loss occurred. Thus the new values of the threshold and window will be 4.

Question 6 (2 points)

1. Let W_{avg} denote the average window size. Therefore, on average, W_{avg} packets are in flight which implies an average throughput of $\frac{W_{\text{avg}}}{RTT}$. Also, the average window size is $\frac{W + (W/2)}{2}$ since W is the maximum window size (which is why the loss occurs) and $W/2$ is the minimum window size (which corresponds to a loss) and the window size varies linearly between the two sizes. Therefore, $W_{\text{avg}} = 0.75W$, which gives us the average throughput as $\frac{0.75W}{RTT}$.

2. a) The loss rate, L, is the ratio of the number of packets lost over the number of packets sent. In a cycle, 1 packet is lost. The number of packets sent in a cycle is

$$\begin{aligned}
\frac{W}{2} + \left(\frac{W}{2} + 1\right) + \Lambda + W &= \sum_{n=0}^{W/2} \left(\frac{W}{2} + n\right) \\
&= \left(\frac{W}{2} + 1\right)\frac{W}{2} + \sum_{n=0}^{W/2} n \\
&= \left(\frac{W}{2} + 1\right)\frac{W}{2} + \frac{W/2(W/2+1)}{2} \\
&= \frac{W^2}{4} + \frac{W}{2} + \frac{W^2}{8} + \frac{W}{4} \\
&= \frac{3}{8}W^2 + \frac{3}{4}W
\end{aligned}$$

Thus, the loss rate is

$$L = \frac{1}{\frac{3}{8}W^2 + \frac{3}{4}W}.$$

b) For W large, $\frac{3}{8}W^2 \gg \frac{3}{4}W$. Thus, $L \approx \frac{8}{3}W^2$ or $W \approx \sqrt{\frac{8}{3L}}$. From the text, we therefore have

$$\begin{aligned}
\text{average throughput} &= \frac{3}{4}\sqrt{\frac{8}{3L}} \cdot \frac{MSS}{RTT} \\
&= \frac{1.22 \cdot MSS}{RTT \cdot \sqrt{L}}.
\end{aligned}$$