CSC358 A2 report
Name: xuwencai, Student Number: 1004750209, utorid: xuwencai
Name: Xuan Du, Student Number: 1004079356, utorid: duxuan1

Overall

For Stop and Wait, everything works successfully. Messages can be all sent successfully even when P(loss) and P(corrupt) approaches 1.

For Go back N protocol, everything works successfully. When I gradually increase the loss probability, and corrupt probability until 1, the program will never encounter an infinity loop.

Both programs will run longer when the corrupt probability and loss probability increase, which makes sense since we need to resend more messages. When the loss probability or corrupt probability be 1, the program will go to infinity loop, because in this case B will never receive a valid packet or never receive any packet.

Stop And Wait protocol

Stop And Wait protocol is very straightforward that when the sender doesn't receive the correct ACK from the receiver and timeout, a resend will be operated.

```
/* called when A's timer goes off (timeout) */
void A_timeout(void) {
    printf("    A_timeout: resend packet: %s.\n", A_side.prev_pck.payload);
    to_network_layer(A, A_side.prev_pck);
    starttimer(A, A_side.increment);
}
```

Also both sides can handle different situations in all states for rdt3.0 FSM. Below is the printout of Network Simulator, which terminated successfully.
Application ends up receiving 9 out of 10 messages, the reason for 10 messages being sent but only 9 received is that on an average interval of 1000, a very short interval (about 10) interrupts the sending which is almost unlikely to happen (but it happens).
So overall it is the correct implementation for Stop and Wait.

/saw 10 0.2 0.3 1000

```
-----  RDT Network Simulator --------

the number of messages to simulate: 10
packet loss probability: 0.200000
packet corruption probability: 0.300000
average time between messages from sender's app layer: 1000.000000
TRACE: 3
      GENERATE NEXT ARRIVAL: creating new arrival
      INSERTEVENT: time is 0.000000
      INSERTEVENT: future time will be 1870.573975
EVENT time: 1870.573975,  type: 1, from app layer entity: 0
      GENERATE NEXT ARRIVAL: creating new arrival
      INSERTEVENT: time is 1870.573975
      INSERTEVENT: future time will be 3512.483887
      MAINLOOP: data given to student: aaaaaaaaaaaaaaaaa
   A_send: send packet: aaaaaaaaaaaaaaaaa.
      TO_NETWORK_LAYER: packet not lost
      TO_NETWORK_LAYER: packet being corrupted
      TO_NETWORK_LAYER: scheduling arrival on other side
      INSERTEVENT: time is 1870.573975
      INSERTEVENT: future time will be 1876.039062
      START TIMER: starting timer at 1870.573975
```

INSERTEVENT: time is 1870.573975
INSERTEVENT: future time will be 2020.573975
EVENT time: 1876.039062, type: 2, from network layer entity: 1
B_recv: packet corrupted.
TO_NETWORK_LAYER: packet not lost
TO_NETWORK_LAYER: scheduling arrival on other side
INSERTEVENT: time is 1876.039062

There's more going on but I skipped it for the page limit.

INSERTEVENT: future time will be 13167.271484
EVENT time: 13167.271484, type: 2, from network layer entity: 0
A_recv: packet corrupted.
EVENT time: 13313.125977, type: 0, timerinterrupt entity: 0
A_timeout: resend packet: jjjjjjjjjjjjjjjj.
TO_NETWORK_LAYER: packet not lost
TO_NETWORK_LAYER: scheduling arrival on other side
INSERTEVENT: time is 13313.125977
INSERTEVENT: future time will be 13320.694336
START TIMER: starting timer at 13313.125977
INSERTEVENT: time is 13313.125977
INSERTEVENT: future time will be 13463.125977
EVENT time: 13320.694336, type: 2, from network layer entity: 1
B_recv: recv message: jjjjjjjjjjjjjjjj
B_recv: send ACK.
TO_NETWORK_LAYER: packet not lost
TO_NETWORK_LAYER: scheduling arrival on other side
INSERTEVENT: time is 13320.694336
INSERTEVENT: future time will be 13324.868164
TO_APP_LAYER: data received: jjjjjjjjjjjjjjjj
EVENT time: 13324.868164, type: 2, from network layer entity: 0
A_recv: ACK received.
STOP TIMER: stopping timer at 13324.868164
Terminated at time 13324.868164
after sending 10 msgs from app layer

Go-back-N protocol

The stderr output is too long and hard to explain so I am using output from printf statement added by myself to analyze.

./a.out 20 0.2 0.3 10

B_recv: packet corrupted
A_recv: valid acknowledgement, but sequence number greater than the base.
timeout, resend packets between sequence number 1 and sequence number 4
B_recv: packet corrupted
A_recv: packet corrupted.
B_recv: valid packet but seqnum is not expected seqnum
A_recv: packet corrupted.
timeout, resend packets between sequence number 1 and sequence number 6
B_recv: valid packet but seqnum is not expected seqnum
B_recv: valid packet but seqnum is not expected seqnum
A_recv: packet corrupted.
B_recv: packet corrupted
A_recv: packet corrupted.
B_recv: valid packet but seqnum is not expected seqnum
B_recv: valid packet but seqnum is not expected seqnum
B_recv: packet corrupted
A_recv: packet corrupted.
timeout, resend packets between sequence number 1 and sequence number 7
A_recv: valid acknowledgement, but sequence number greater than the base.
B_recv: packet corrupted
B_recv: valid packet but seqnum is not expected seqnum
A_recv: packet corrupted.
B_recv: valid packet but seqnum is not expected seqnum
B_recv: valid packet but seqnum is not expected seqnum
A_recv: packet corrupted.
timeout, resend packets between sequence number 1 and sequence number 8
B_recv: valid packet but seqnum is not expected seqnum
A_recv: valid acknowledgement, but sequence number greater than the base.
B_recv: valid packet
A_recv: valid acknowledgement, but sequence number greater than the base.
A refuse data from application layer
B_recv: valid packet
B_recv: packet corrupted
timeout, resend packets between sequence number 1 and sequence number 9
A_recv: valid acknowledgement, but sequence number greater than the base.
A refuse data from application layer
B_recv: packet corrupted
A_recv: packet corrupted.
A_recv: valid acknowledgement packet.
B_recv: packet corrupted
A_recv: valid acknowledgement, but sequence number greater than the base.
B_recv: packet corrupted
B_recv: packet corrupted
B_recv: packet corrupted
A_recv: valid acknowledgement, but sequence number greater than the base.
B_recv: valid packet but seqnum is not expected seqnum
timeout, resend packets between sequence number 3 and sequence number 11
B_recv: valid packet
A_recv: valid acknowledgement, but sequence number greater than the base.
A refuse data from application layer
A refuse data from application layer
A_recv: packet corrupted.
B_recv: valid packet
A refuse data from application layer
A_recv: packet corrupted.
B_recv: valid packet
A_recv: valid acknowledgement packet.
B_recv: valid packet
B_recv: packet corrupted
B_recv: packet corrupted
B_recv: valid packet but seqnum is not expected seqnum
timeout, resend packets between sequence number 6 and sequence number 12
B_recv: valid packet but seqnum is not expected seqnum
B_recv: packet corrupted
A_recv: valid acknowledgement packet.
B_recv: packet corrupted
A_recv: valid acknowledgement, but sequence number greater than the base.
A_recv: packet corrupted.
B_recv: packet corrupted
B_recv: valid packet but seqnum is not expected seqnum
A refuse data from application layer
B_recv: valid packet
B_recv: valid packet
timeout, resend packets between sequence number 7 and sequence number 15
A_recv: valid acknowledgement, but sequence number greater than the base.
B_recv: valid packet
A_recv: packet corrupted.
B_recv: packet corrupted

```
A_recv: valid acknowledgement packet.
A_recv: valid acknowledgement packet.
B_recv: valid packet but seqnum is not expected seqnum
B_recv: valid packet but seqnum is not expected seqnum
A_recv: valid acknowledgement, but sequence number greater than the base.
B_recv: packet corrupted
A_recv: valid acknowledgement, but sequence number greater than the base.
B_recv: valid packet
A_recv: valid acknowledgement, but sequence number greater than the base.
timeout, resend packets between sequence number 10 and sequence number 15
B_recv: valid packet
B_recv: valid packet
A_recv: valid acknowledgement packet.
B_recv: valid packet
A_recv: valid acknowledgement packet.
B_recv: packet corrupted
A_recv: packet corrupted.
B_recv: valid packet but seqnum is not expected seqnum
B_recv: valid packet but seqnum is not expected seqnum
B_recv: packet corrupted
A_recv: valid acknowledgement, but sequence number greater than the base.
B_recv: valid packet but seqnum is not expected seqnum
timeout, resend packets between sequence number 14 and sequence number 15
A_recv: valid acknowledgement, but sequence number greater than the base.
B_recv: packet corrupted
B_recv: valid packet
A_recv: valid acknowledgement, but sequence number greater than the base.
B_recv: packet corrupted
A_recv: valid acknowledgement, but sequence number greater than the base.
timeout, resend packets between sequence number 14 and sequence number 15
B_recv: valid packet but seqnum is not expected seqnum
A_recv: packet corrupted.
B_recv: valid packet but seqnum is not expected seqnum
B_recv: valid packet but seqnum is not expected seqnum
A_recv: valid acknowledgement packet.
B_recv: packet corrupted
A_recv: valid acknowledgement, but sequence number greater than the base.
B_recv: valid packet but seqnum is not expected seqnum
A_recv: packet corrupted.
A_recv: valid acknowledgement, but sequence number greater than the base.
Terminated at time 341.334229
 after sending 20 msgs from app layer
```

Firstly, whenever a timeout happens, I resend packets in the window that hasn't received an acknowledgement. Note from above print statements the number of packets I resend whenever a timeout happens are never greater than 8, since the window size is 8 in this assignment.
Secondly, the corrupt probability is 0.3, and packet corrupted lines above are nearly 30 percent of the total lines.
Also, there are 20 messages being sent, but we need to resend a lot of messages, so that's why numbers of B_recv lines are much greater than 20, which also makes sense.
The loss probability is 0.2, which means there will be time that the clock timeout in A. I check from above timeout happens 9 times (the number of print statements for timeout is 11), and that is because sometimes A does not receive a packet for a long time, or A receive a corrupt acknowledgement, 20 * 0.2 + 20 * 0.3 = 10. Timeout 11 times is a fair number.
Both A and B will receive a corrupt packet, which is what I expected, because the checksum includes acknum, in order to check the corrupt packet that B sends.
Sometimes B will receive a valid packet, but the sequence number is not the expected sequence number. This is the case when A sends a packet too quickly, the packet seqnum is not B expected.