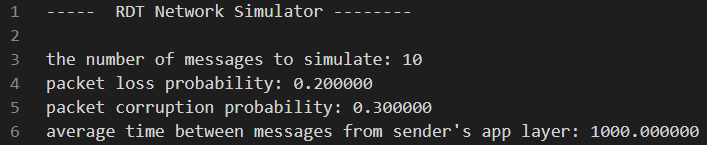
|  |  |  |  |
| --- | --- | --- | --- |
| Name | Youan Cong | Student number | 1005251184 |
| Name | Wentao Zhou | Student number | 1005308490 |

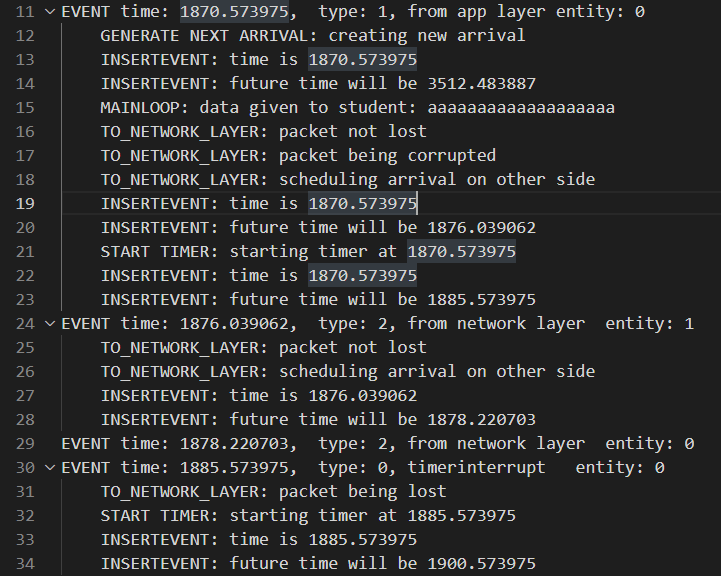
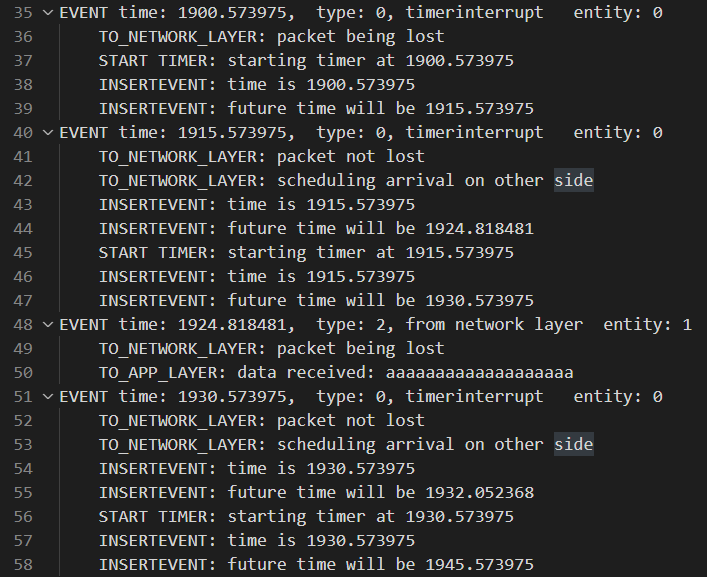
**Description:**

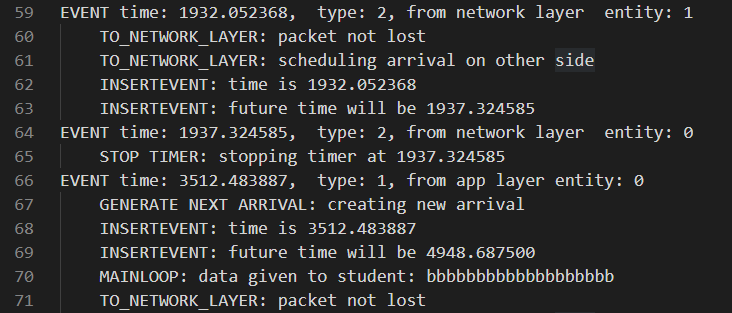
In this assignment, we successfully completed all the required 2 different protocols, and successfully passed the test. We used 15 time unit for timer because this is a round trip so average will take 10 time unit. The rest 5 time unit is for flexibility and code operation.

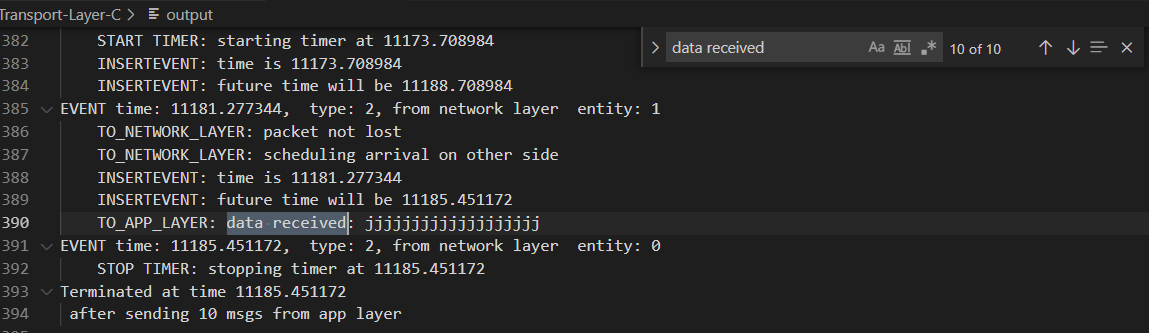
**Example 1: saw.c**

**./a.out 10 0.2 0.3 1000 (Redirect the strerr to output.txt)**









**Implementation details & Test case explanation**

In function A\_send, we set up the packet with the data, current seq num and generate the checksum for it. Then make a copy of the packet for later re-sending, send the pack to B and start the timer.

In line 24, we can see the packet arrives but it’s corrupted (No received and send to application). Our if checking catches this case with the checksum checking and send the opposite acknum to A\_recv.

In line 29, A\_recv receives the wrong acknum or corrupted packet and our code just returns and let the timer runs out

Starting line 30, timer keeps running out, re-send the copied packet and restart the timer (because of the packet loss from A to B).

In line 48, B receives the packet and decides the data and seq num are both correct. Then send the acknum to A\_recv, reverse the seq num and send the data to application.

In line 51, packet loss between B to A and A\_timout again resends the packet.

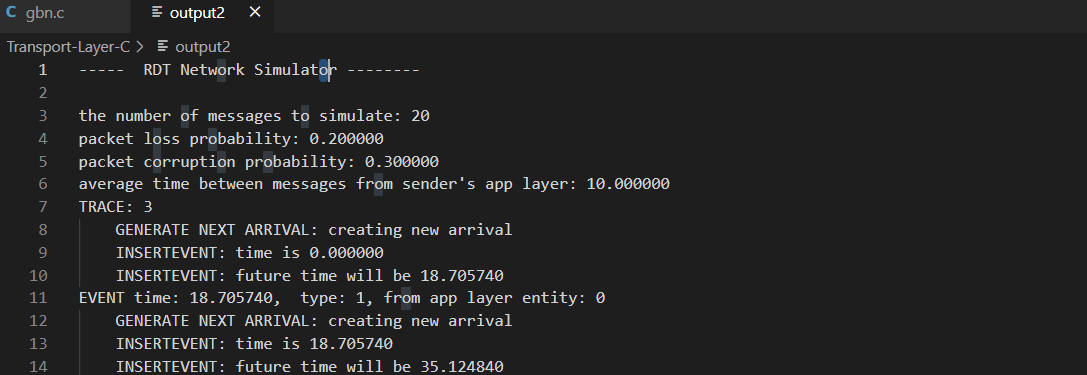
In line 64, B\_recv receives the old packet with wrong seq num. It just sends the acknum right back.

In line 66, A\_recv receives the not corrupted and right acknum. It reverses the seq num for next data, set the received variable to 1 and stop the timer.

Obviously, from the picture, the last received data is exactly the 10th letter, “j”, and by searching it can be seen that all 10 data have been completely received.

**Example 2: gbn.c**

**./a.out 20 0.2 0.3 10 (Redirect the strerr to output.txt)**



**Implementation details & Test case explanation**

For GBN, since some details of the basic operations are similar with saw.c, only the worth noting implementations are explained here.

Once we received a packet, we set up the packet with the data and checksum in A\_send, and then add it to the packet list. Then we send the packet with expected num from the packet list within the range of window. The packet list can be both used as the storage for packet within the window and the buffer for packet out of the window.

Similarly, our GBN can handle the corrupt packet like what we did in exmaple 1 using checksum. B\_rev will check it and the seqnum, if the seqnum of the packet cannot match B’s expected num, a ACK of the most recent received packet will be sent.

For A\_rec(), if the packet passes the checksum validation, then the base will be updated to the seq num of the most recent ACKed packet, and it will control timer when all packets in the window aer sent but aren’t ACKed.

A\_timeout() will resend packets from base to nextnumseq-1(which are those sent but not yet ACKed packets in the window) using the packet array.

