COMP 3331 ASSIGNMENT REPORT:

I have written the code in **Python**.

FILE STRUCTURE:

receiver.py = the server runs on this python file which receives the data
sender.py = the sender of the file runs on this python file
receiver_log.txt = the file that keeps live logs of what receiver.py is doing
sender_log.txt = the file that keeps live logs of what sender.py is doing

NOTE: if the provided file to transfer and the provided file to write the data in, are missing an .txt extension then it will be auto added and not if already there.

My codes workflow looks like the following:

1. User runs the code of receiver and sets it on a port, gives it the file which it needs to create and put all the incoming packets in, also give the max window size and sender.
2. Then the user opens new terminal and starts the sender code with sender port, receiver port, the text file the user wants to transfer, the max window size avail for packet transfer, the probability at which retransmissions should occur in case ACK is missed, the probability that a particular packet can be lost during transmission and finally a probability that a ACK can be missed during transmission.
3. Then receiver will create the new empty file as given by the cmd args.
4. The STP packet as stated is 1004 bytes that is 2 bytes for the data type which is an integer then 2 for the sequence number which is again an integer and 1000 bytes for the data that is being transferred.
5. Now sender will create a random ISN between 1 and 2^16.
6. Now sender will try and send a SYN segment and receive an ACK for it until this isn't done it won't move to established part, the receiver will also check if any more SYN segments are being sent for response, then when accepted both move to established.
7. Now sender will start sending packets to the receiver according to the window size requirements.
8. It will send as many as it could in one go and then wait for the ACK for all the sent data if an ACK is not received or was missed or data packet was dropped the sender will resend the data packet again after the timer expires for resending and

then keep resending the data packet in intervals until an ACK is received for the last unacked packet. As soon as the first ACK is received the window will slide and send one more data packet and if more than 1 ACK was received and the window allowed then also more than 1.

9. If the receiver doesn't receive the data packet in seq then it will send ACK and request data send from the sender. dupAckResend
10. The code is using one thread to send all the data packets and one to listen to the ACKS and resend data segments in case timer expires without any ACK for the segment.
11. At the same time sender and receiver are also creating their own personal logs which are being live updated according to whatever is happening.
12. At the end I have provided a timeout to wait so that all the packets can receive an ACK and then function can move to the FIN section
13. Here FIN is sent similar to SYN and tries to get a ACK telling receiver that function has finished here receiver waits 2 seconds after receiving the FIN this is in case the ACK to that FIN is dropped and sender is sending another FIN.
14. The receiver is also using a buffer code to maintain the sequencing of the data packets and set the file in the correct order like how the original one was.

Code LIMITATION and its SOLUTION:

1. In case loss probability is set high, there is a possibility that whenever receiver gets the first FIN and sends the ACK the ACK is missed by the sender hence it will still keep on sending FIN requests but receiver is now under time limitation of 2 seconds and if a FIN ACK handshake is not completed in that time frame then receiver will quit and this is the case where sender can keep running due to no ACK received for FIN.
    a. For this I have now implemented a simple fix that sender will also kill itself after 5 seconds of first FIN call as now receiver is already dead and no point sending more FIN and be stuck in loop (logging this with timeout if this occurs which is extremely rare and mostly only seen when rlp is 95+)

HELP:

The given udp receiver and sender by the course were used to setup the socket in my own code.