COMP9331 Assignment report

Name: Xinyu Wang
zid: z5143329

Python version: 3.6

Question1:

I made two classes called Sender and Receiver as well as a STP packet, it contains all the information that is needed for sap process.

Then I start with simple udp transfer protocol, by implementing simple error-free file transfer on my localhost.

Since file we handle is pdf file, so I use ‘wb’ and ‘rb’ as well as ‘a+’ to read and write data to newly created file.

And then I move to the very first feature: 3-way handshake and FIN-ACK FIN-ACK feature in TCP, To establish a reliable connection between I start transferring actual data. I tested many times to ensure it work without bugs.

The next step is to implement a stop-and-wait protocol based on the connection I have established. In order to sent more than one packets to receiver, I wrote function to split the whole app_data into segments that are size of MSS(Max segment size), and transfer them one by one. During this process, I implement appropriate logic for updating sequence number and ack number. Basically the logic is: if segments have been transferred successfully, the sequence should be like this: seq_num += len(data that has been successfully transferred)

The next step I take is to create two method called write_to_sender_log() and write_to_receiver_log(), they will update sender_log.txt and receiver_log.txt when packets are transferred between sender and receiver.

The next step I take is to create PLD module, it is basically a function which take pdrop and seed as argument, then it generate random value between 0 and 1 and compare it with pdrop, to decide if the packet to be dropped or not.

The next feature I implement is drop packets. For a simple stop-and-wait protocol, the way my program use to handle dropped packets is: when sender drop a packet, it won’t know if the packet has reached the receiver successfully, so it will simply wait for the right ack back from receiver, but it is dropped and sender will never get the ack it wants, so sender will start a timer when it send a packet, and wait till the timeout of the socket, and then it will know that packet has been dropped, and it then retransmit the dropped packet. In this case I tested many values of pdrop to ensure that it can tolerant various situations.

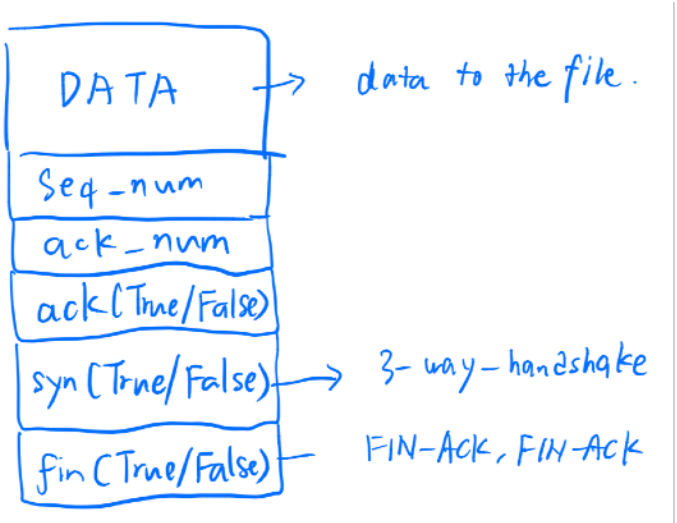
After this, I started implementing pipeline protocol, normally I need to fill in the window (MWS/MSS) segments before receive ack. Here is how my method looks like:

- initially reset num_seg_in_window to 0
- Keep sending segments before window is full
- Now window is full, receive ack from receiver, window free 1 segment at the meantime.
- If a segment is dropped, receiver keep sending ack of the non_received segments, so sender will finally find the segments is dropped, and retransmit the segment. **Receiver can receive all the data that after the dropped segments but store those into a buffer**, after it get the dropped segment from sender, it will write the whole buffer into file, in an appropriate sequence (after the dropped segment). And things can keep going.

And I have also created some variables like data_position, base_size, num_in_window to help implementing this assignment.

Implemented Features	Not Implemented
Both: Three-way-handshake (SYN SYNACK ACK)	Duplicate, corrupt, delay, order, maxorder,
Both: Four-segment cnnt termination (FIN ACK FIN ACK)	
Sender: Single-timer for timeout operation	
Sender: Simple timeout retransmit	
Receiver: Immediate acknowledgement / ACKs	
Both: Sequence Numbers, Acknowledgement Numbers	
Sender: Maximum Segment Size (MSS)	
Sender: Packet Loss and Delay (PLD)	
Sender: pipeline protocol	
Sender: Fast retransmit	

Question 2



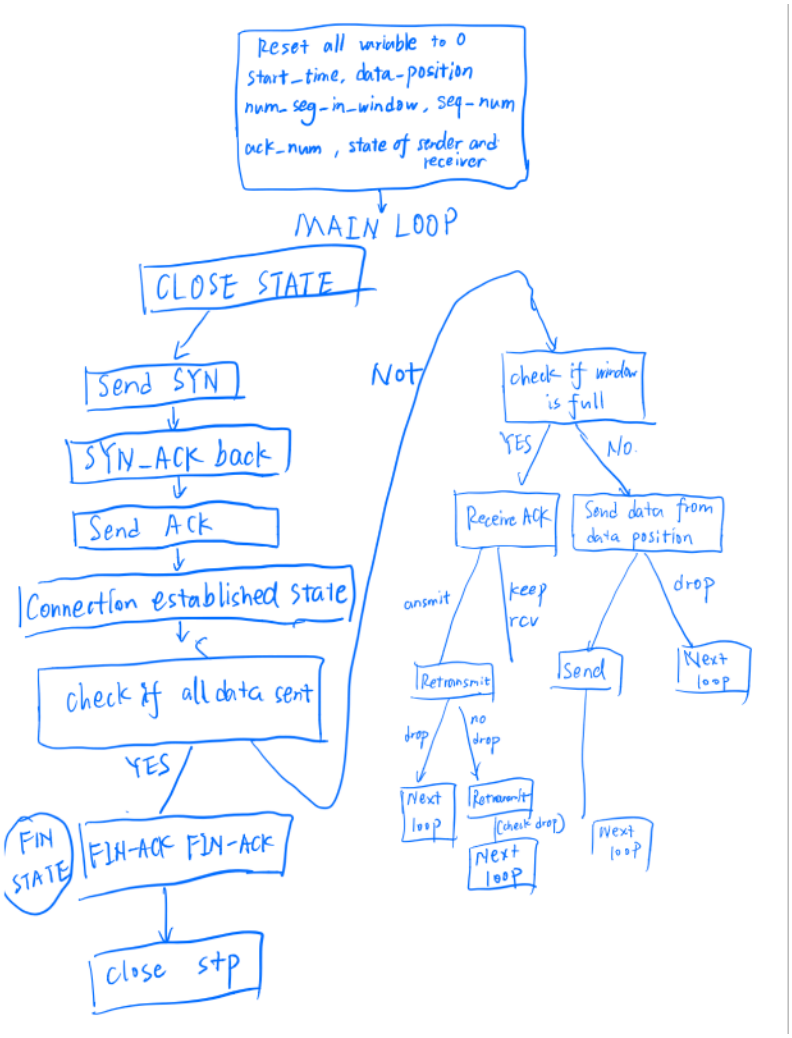
Question 3

(Some thoughts about pDuplicate: I think the method to implement this is to let the receiver to check if seq_num equal to ack, if duplicated, then it sends a signal to sender, and receiver just drop the duplicated segments. I think it is doable, but I already late for one day, I really struggled to do this assignment…)

My program cannot handle the situation after one packet is dropped, and before the dropped packet is retransmitted, there is another packet dropped. I cannot solve this. So when pdrop is high, program will sometimes crash. But it works when MWS/MSS is 1, which is stop-and-wait protocol, in this case it can deal all situations.

The structure of my code is a little bit too complicated, which makes it hard for me to debug. So I will have to set a overall structure before implementing the code.

Question a:



The sequence is as the draw I attached, argument is :
MWS is 600, MSS is 150, seed is 5, pdrop is 0.1.

snd	0.00	S	0	0	0
rcv	0.00	SA	0	0	1
snd	0.01	A	1	0	1
snd	0.01	D	1	150	1
snd	0.01	D	151	150	1
snd	0.01	D	301	150	1
snd	0.01	D	451	150	1

rcv	0.01	A	1	0	151
snd	0.01	D	601	150	1
rcv	0.01	A	1	0	301
snd	0.01	D	751	150	1
rcv	0.01	A	1	0	451
drop	0.01	D	901	150	1
rcv	0.01	A	1	0	601
snd	0.01	D	1051	150	1
rcv	0.01	A	1	0	751
snd	0.01	D	1201	150	1
rcv	0.01	A	1	0	901
snd	0.01	D	1351	150	1
rcv/DA	0.01	A	1	0	901
rcv/DA	0.01	A	1	0	901
rcv/DA	0.01	A	1	0	901
snd/RXT	0.01	D	901	150	1
snd	0.01	D	1501	150	1
snd	0.01	D	1651	150	1
snd	0.01	D	1801	150	1
rcv	0.01	A	1	0	1501
snd	0.01	D	1951	150	1
rcv	0.01	A	1	0	1651
snd	0.01	D	2101	150	1
rcv	0.02	A	1	0	1801
drop	0.02	D	2251	150	1
rcv	0.02	A	1	0	1951
snd	0.02	D	2401	150	1
rcv	0.02	A	1	0	2101
snd	0.02	D	2551	150	1
rcv	0.02	A	1	0	2251
snd	0.02	D	2701	150	1
rcv/DA	0.02	A	1	0	2251
rcv/DA	0.02	A	1	0	2251
rcv/DA	0.02	A	1	0	2251
snd/RXT	0.02	D	2251	150	1
snd	0.02	D	2851	150	1
snd	0.02	D	3001	28	1
rcv	0.02	A	1	0	2851
rcv	0.02	A	1	0	3001
rcv	0.02	A	1	0	3029
snd	0.02	F	3029	0	1
rcv	0.02	A	1	0	3030
rcv	0.02	F	3030	0	1
snd	0.02	A	3029	0	2

==========

Size of the file (in Bytes)	3028
Segments transmitted (including drop & RXT)	27
Number of segments handled by PLD	23
Number of segments dropped	2
Number of segments Corrupted	0
Number of segments Re-ordered	0
Number of segments Duplicated	0
Number of segments Delayed	0
Number of Retransmissions due to TIMEOUT	0
FAST RETRANSMISSION	2
Number of DUP ACKS received	6

==========