Rahil Agrawal
z5165505
COMP3331/9331 Computer Networks and Applications
Assignment for Session 2, 2018

Section 1

Introduction

This assignment has been implemented in Java. The two main files are Sender.java and Receiver.java. These files interact with each other with the use of SenderThread.java and ReceiverThread.java to enable concurrency and sending and receiving packets at each side. This ensures use of the same socket to receive and send packets.

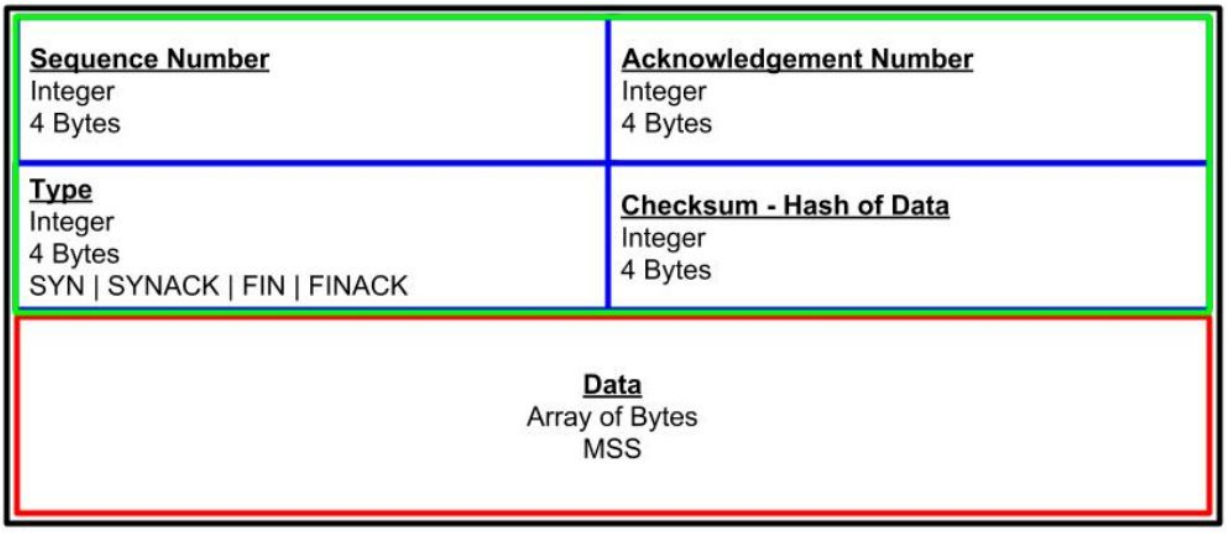
Major features implemented

1. Sender creates packets (data size = MSS) and forwards them to PLD Module to send the to the Receiver.
2. PLD Module uses pseudo-random number to drop, duplicate corrupt or reorder a packet or send it normally (or Retransmit).
3. Sender does the above until it reaches MWS and then checks all the ACKs it has received meanwhile.
4. Receiver receives a packet, sends back a cumulative ACK. (Corrupted packets are discarded, proper packets are stored for writing to a file later).
5. On receiving ACKs (in the background, done using Threads), the sender stores them for processing once it has sent a certain number of packets (based on MWS). After this, acks are process for Triple Duplicate Acks, lastByteSent and lastByteAcked are recorded, and appropriate packets are sent to PLD yet again..
6. Timeout has been implemented using formulas discussed in lecture slides. *However, the maximum value of timeout was made 5 seconds (instead of 1 minute) to run programs faster. (Packet sequence was observed to be similar or same)*

Unfortunately, the following requirements could **not** be implemented

1. Due to lack of time and urge to submit for 10% bonus marks, delay features of the PLD module could not be implemented.
2. I was unable to debug why the files in 7(b)(i), 7(b)(ii) and 7(c) were taking so long to transfer but I believe it is because of the number of times every packet is retransmitted and acked after a certain while. The MWS reduces to one packet and it takes almost 10 seconds for each packet to be successfully sent because of the number of RXTs and Acks for each packet.

Section 2



Section 3

Trade-offs

1. Some features were not implemented because I found them difficult to incorporate without breaking other parts of the code. Time constraints to submit early also made this harder to work on.
2. Code is not the best in aspects of elegance and security. More emphasis was put on making things work rather than a very neat coding style.

Possible Improvements and Extensions

1. The overall structure and implementation could be improved by planning about concurrency at the start rather than later in the assignment - Better understanding of threads and packets storing and manipulation will be needed for this.
2. Features can be implemented in a much better way using smarter and more elegant data structures such as HashMaps that identify using packets. This will make accuracy of timeout values etc better - More hours need to be invested in development.
3. The program can be extended to incorporate delaying a packet. For this, I don't have a solid understanding. But my approach would be to store the time at which this packet arrives at the PLD and then wait (by using a timer or checking system time before every other packet is sent) until pDelay is reached. Meanwhile, other packets are allowed to be sent. When the timer reaches maxDelay, we can send the stored packet.

Section 4

Following segments were borrowed from the internet

1. Converting a packet object to byte array and vice-versa : https://www.javahelps.com/2015/07/serialization-in-java.html

Section 5

A.

a. pDrop = 0.1
b. pDrop = 0.3

In both cases, when dropping of packets occurs (below), sender continues to send packets until MWS is reached. After this, sender processes the ACKs and realises via 3-dup-ack or via timeout that it has not sent some packets. It then sends the required packets (below). Everytime packet(s) is dropped, this behavior is observed. This is seen more often in (b) because the probability of dropping a packet is higher than that in (a) and the seed for random number generator is the same.

B. Gamma

Gamma	STP Packets from Sender to Receiver	Time taken for Transfer
2	31395	6918.18 seconds
4	-- Takes too long --	-- Takes too long --
6	-- Takes too long --	-- Takes too long --

I believe that my program is not the best when it has to deal with a lot of dropped packets with a huge timeout. This is because it attempts to send the same packet multiple times and drops a lot of times. However, the file is transferred correctly if program is run long enough. Just the number of packets sent and received are a lot.

C. MWS=500 MSS=50 gamma=4 pDrop=0.1 pDuplicate=0.1 pCorrupt=0.1 pOrder=0.1 maxOrder=4 pDelay=0 maxDelay=0 seed=300
   a. File Transferred Successfully : No
   b. How long did the overall transfer take : Could not finish in 3 hours so I stopped the Transfer.
   c. Most critical contributing factor : pDrop - This is because after a few errors in sending, the MWS shrinks and only a couple of packets can be sent. When a packet is dropped, the sender does not receive an ack from the receiver. This causes in the sender to wait for timeout. After this, sender attempts to send the

packet again. This causes quite a lot of delay. Also, when sender continue to send packets after dropping a packet, it will receive an ACK for the dropped packet and it will have to send it again through the PLD Module. Other factors such as pCorr, pDup and pOrder will keep sending other packets and not cause the delay that timeout does. With pCorr and pOrd, the missing packet is retransmitted using 3-dup-Ack which is usually faster than timeout. This causes the packets and MWS to come in sync faster. Whereas with pDrop, it takes a lot longer.

Appendix

Sequence of Packets Received for Section 5(A)(a)

rcv	1.78	S	0	0	0
rcv	1.81	A	1	0	1
rcv	1.82	D	1	100	1
rcv	1.88	D	101	100	1
rcv	2.20	D	201	100	1
rcv	2.20	D	301	100	1
rcv	2.24	D	401	100	1
rcv	2.30	D	501	100	1
rcv	2.37	D	501	100	1
rcv	2.64	D	601	100	1
rcv	2.79	D	701	100	1
rcv	2.90	D	801	100	1
rcv	2.95	D	901	100	1
rcv	2.98	D	1001	100	1
rcv	2.99	D	1001	100	1
rcv	3.00	D	1101	100	1
rcv	3.01	D	1201	100	1
rcv	3.01	D	1301	100	1
rcv	3.02	D	1401	100	1
rcv	3.03	D	1501	100	1
rcv	3.05	D	1601	100	1
rcv	3.06	D	1701	100	1
rcv	3.07	D	1801	100	1
rcv	3.07	D	1901	100	1
rcv	3.08	D	2001	100	1
rcv	3.09	D	2001	100	1
rcv	3.09	D	2101	100	1
rcv	3.10	D	2401	100	1
rcv	3.12	D	2601	100	1
rcv	3.12	D	2701	100	1

rcv	3.12	D	2201	100	1
rcv	3.13	D	2201	100	1
rcv	3.13	D	2801	100	1
rcv	3.14	D	2301	100	1
rcv	3.14	D	2901	100	1
rcv	3.15	D	2501	100	1
rcv	3.15	D	3001	28	1
rcv	3.30	F	3029	0	1
rcv	3.31	A	3030	0	2

==========

Amount of data received (bytes)	3428
Total Segments Received	39
Data segments received	35
Data segments with Bit Errors	0
Duplicate data segments received	4
Duplicate ACKs sent	4

==========

Sequence of Packets Received for Section 5(A)(b)

rcv	1.59	S	0	0	0
rcv	1.63	A	1	0	1
rcv	1.77	D	1	100	1
rcv	1.77	D	101	100	1
rcv	1.77	D	201	100	1
rcv	1.77	D	301	100	1
rcv	1.77	D	401	100	1
rcv	1.79	D	501	100	1
rcv	1.80	D	601	100	1
rcv	1.80	D	801	100	1
rcv	2.27	D	901	100	1
rcv	2.31	D	1001	100	1
rcv	2.31	D	1101	100	1
rcv	2.32	D	1201	100	1
rcv	2.46	D	701	100	1
rcv	2.46	D	701	100	1
rcv	2.46	D	701	100	1
rcv	3.01	D	1301	100	1
rcv	3.01	D	1401	100	1
rcv	3.01	D	1601	100	1
rcv	3.01	D	1701	100	1
rcv	3.02	D	1801	100	1
rcv	3.08	D	2001	100	1

rcv	4.24	D	1501	100	1
rcv	4.24	D	1501	100	1
rcv	4.24	D	1501	100	1
rcv	4.24	D	2201	100	1
rcv	4.24	D	2401	100	1
rcv	4.24	D	1901	100	1
rcv	4.24	D	1901	100	1
rcv	4.24	D	2501	100	1
rcv	4.24	D	2601	100	1
rcv	4.25	D	2101	100	1
rcv	4.26	D	2101	100	1
rcv	4.26	D	2701	100	1
rcv	5.18	D	2301	100	1
rcv	5.18	D	2301	100	1
rcv	5.18	D	2301	100	1
rcv	5.18	D	2301	100	1
rcv	5.18	D	2301	100	1
rcv	5.18	D	2301	100	1
rcv	5.18	D	2301	100	1
rcv	5.18	D	2801	100	1
rcv	5.18	D	2901	100	1
rcv	5.18	D	3001	28	1
rcv	5.18	D	2801	100	1
rcv	5.18	D	2801	100	1
rcv	5.18	D	3001	28	1
rcv	5.79	F	3029	0	1
rcv	5.79	A	3030	0	2

==========

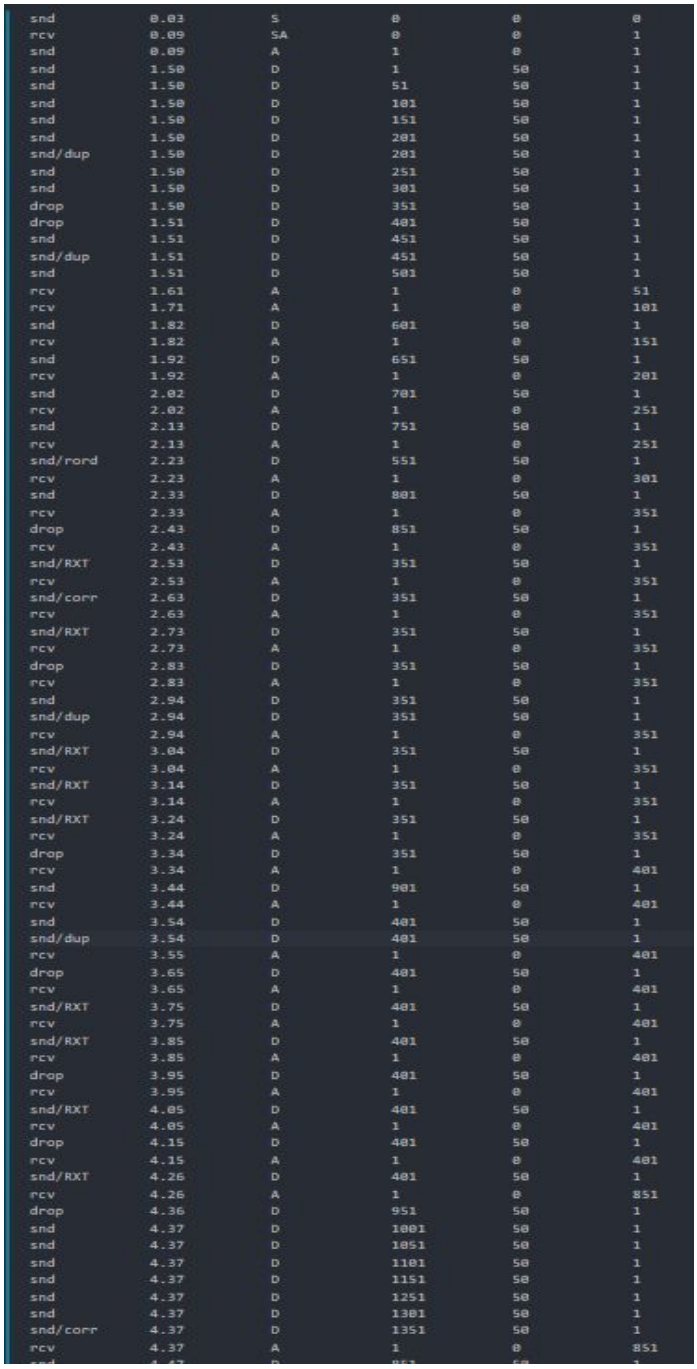
Amount of data received (bytes)	4456
Total Segments Received	50
Data segments received	46
Data segments with Bit Errors	0
Duplicate data segments received	15
Duplicate ACKs sent	15

==========

Summary of Sender for Section 5(B)(i)

Size of the file (in Bytes)	308203
Segments Transmitted (including drop & RXT)	31395
Number of Segments handled by PLD	31391
Number of Segments dropped	15624
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	12716
Number of FAST RETRANSMISSION	3234
Number of DUP ACKS received	12627

Sequence of Packets for Section 5(C)



snd	0.03	S	0	0	0
rcv	0.09	SA	0	0	1
snd	0.09	A	1	0	1
snd	1.50	D	1	50	1
snd	1.50	D	51	50	1
snd	1.50	D	101	50	1
snd	1.50	D	151	50	1
snd	1.50	D	201	50	1
snd/dup	1.50	D	201	50	1
snd	1.50	D	251	50	1
snd	1.50	D	301	50	1
drop	1.50	D	351	50	1
drop	1.51	D	401	50	1
snd	1.51	D	451	50	1
snd/dup	1.51	D	451	50	1
snd	1.51	D	501	50	1
rcv	1.61	A	1	0	51
rcv	1.71	A	1	0	101
snd	1.82	D	601	50	1
rcv	1.82	A	1	0	151
snd	1.92	D	651	50	1
rcv	1.92	A	1	0	201
snd	2.02	D	701	50	1
rcv	2.02	A	1	0	251
snd	2.13	D	751	50	1
rcv	2.13	A	1	0	251
snd/rord	2.23	D	551	50	1
rcv	2.23	A	1	0	301
snd	2.33	D	801	50	1
rcv	2.33	A	1	0	351
drop	2.43	D	851	50	1
rcv	2.43	A	1	0	351
snd/RXT	2.53	D	351	50	1
rcv	2.53	A	1	0	351
snd/corr	2.63	D	351	50	1
rcv	2.63	A	1	0	351
snd/RXT	2.73	D	351	50	1
rcv	2.73	A	1	0	351
drop	2.83	D	351	50	1
rcv	2.83	A	1	0	351
snd	2.94	D	351	50	1
snd/dup	2.94	D	351	50	1
rcv	2.94	A	1	0	351
snd/RXT	3.04	D	351	50	1
rcv	3.04	A	1	0	351
snd/RXT	3.14	D	351	50	1
rcv	3.14	A	1	0	351
snd/RXT	3.24	D	351	50	1
rcv	3.24	A	1	0	351
drop	3.34	D	351	50	1
rcv	3.34	A	1	0	401
snd	3.44	D	901	50	1
rcv	3.44	A	1	0	401
snd	3.54	D	401	50	1
snd/dup	3.54	D	401	50	1
rcv	3.55	A	1	0	401
drop	3.65	D	401	50	1
rcv	3.65	A	1	0	401
snd/RXT	3.75	D	401	50	1
rcv	3.75	A	1	0	401
snd/RXT	3.85	D	401	50	1
rcv	3.85	A	1	0	401
drop	3.95	D	401	50	1
rcv	3.95	A	1	0	401
snd/RXT	4.05	D	401	50	1
rcv	4.05	A	1	0	401
drop	4.15	D	401	50	1
rcv	4.15	A	1	0	401
snd/RXT	4.26	D	401	50	1
rcv	4.26	A	1	0	851
drop	4.36	D	951	50	1
snd	4.37	D	1001	50	1
snd	4.37	D	1051	50	1
snd	4.37	D	1101	50	1
snd	4.37	D	1151	50	1
snd	4.37	D	1251	50	1
snd	4.37	D	1301	50	1
snd/corr	4.37	D	1351	50	1
rcv	4.37	A	1	0	851