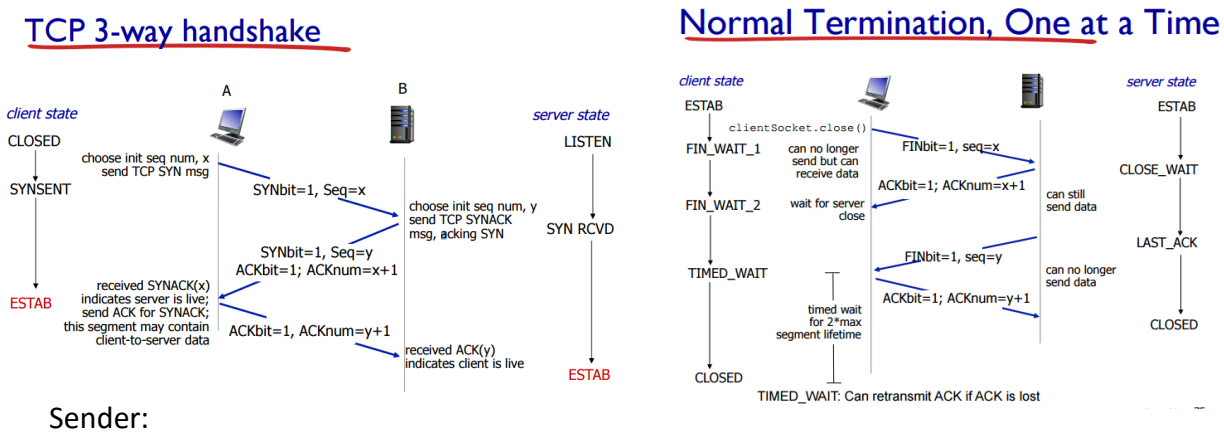
Comp3331 Assignment 1 Report z3492998 Yu Ting Lam

This assignment are written in python version: 2.7.3

1) This STP protocol is designed base on the TCP protocol, including three-way handshake, four-segment connection termination, fast retransmit, timeout retransmit. Also, there is a PLD to model the packet loss condition for standard part. Both sender and receiver file has been log as required and the logs are stored into the specific file. (All the component in standard part is working)

   HowToImplement

   Handshake and the termination is just following the lecture notes that can be explained in the following diagram.



   Sender:

   The standard part of the sender consists of the PLD function when a packet is sent. It is just an if statement to ignore the send to function when the rand number is smaller than the pdrop.

   Re- retransmit in timeout and duplicated ack is pretty similar in making a new packet and send it out though PLD again. Except the timeout would always the very first packet in the window, duplicated ack would packet the packet base on the received ack and computed the one it need (usually should be the first one in window as well)

   Single timeout: a base time would be set at the beginning of data transmit and it would be updated if the window move or timeout retransmit happen. If the different between current time and base time is greater than the timeout input value, the timeout event would be trigger and retransmit the first element in the window. // origin timout

   Duplicated ack would be calculated with a counter which would increase by 1 if the ack is same as the pervious received ack.

   Window array store the whole packet in this assignment. It makes calculating the seq and excepted ack very convenient. However, for large packet size, ***it would be better to just store the seq and len(data) in the window array to save the memory.***

Receiver:

After the handshake has been done, receiver can handle data packet.

Those data packets can arrived not in order, the receiver would always return the smallest continues ack back. i.e if the packet is lost in the middle, the ack would indicated that is missing to the sender as it keeps sending the duplicate ack for the pervious segment. It is same as what TCP will do.

Receiver also has a buffer to store those packets that arrive earlier (not in order). When the missing packet arrived, receive will check on the buffer and return the proper ack back to indicate that it received those packets. This way, the output file for showing the text transfer would be same as the input they have in the sender.

2) The packet is stored in a string array in python.

packet = [seq, ack_num, flag, data]

The packet contain the following element, sequence number, ack number, a flag that contain 3 infrmation (syn, ack_bit and fin) and data at the end.

Packet:

Seq
ack_num
Flag (FLAG_SYN = 0b001 ,FLAG_ACK = 0b010, FLAG_FIN = 0b100)
data

3)

a) Timeout value choose to be 2.5ms

The following record is generated by setting different timeout while keeping other attribute the same

filename = "test1.txt"

mws = 500

mss = 50

pdrop = 0.1

seed = 300

The output is come from the sender log. (last ack)

* Design issue: the Ack num at the end is come from random number. It would change every time

timeout = 1					
snd	89.9150390625	A	0	0	9
timeout = 1.5					
snd	88.5051269531	A	0	0	8
timeout = 2					
snd	73.5322265625	A	0	0	2

timeout = 2.5 snd 53.6059570312 A 0 0 4
timeout = 3 snd 67.9790039062 A 0 0 9
timeout = 4 snd 81.5390625 A 0 0 9

In timeout = 2.5 ms

Pdrop = 0.1	Pdrop = 0.3
snd 269.019042969 A 0 0 2 Amount of data transferred: 2143 bytes Number of data segments sent: 47 Number of packets dropped: 4 Number of retransmitted segments: 15 Number of duplicate acknowledgements: 29	snd 3063.37304688 A 0 0 2 Amount of data transferred: 1743 bytes Number of data segments sent: 48 Number of packets dropped: 13 Number of retransmitted segments: 13 Number of duplicate acknowledgements: 25

Result discuss:

When drop rate increase, the duplicate ack decrease and the time require for the data transmit increase. This is because the fast retransmitted packet has a higher chance to loss. In this case, those lost packet can only wait until it timeout to get resent. Therefore, it would take a longer time to transmit the data.

b)

Timeout\	Total packet transfer	Total time
2.5	62	153.202880859
4*2.5	62	109.207763672
2.5/4	69	107.899902344

*Total segment is calculated by every time PLD function is called.

Discuss:

Case :

Total_time(Tcurrent)> total_time(4* Tcurrent):

Using (4* Tcurrent) can wait for more ack to come back before timeout retransmit. i.e fast re-transmit are more likely to happen.

Therefore, total time is reduce.

Total_packet(Tcurrent)> total_packet(Tcurrent/4):

Timeout would happen more likely as the timeout is shorter.

If the timeout is too short (2.5/4):

Cons: premature timeout, unnecessary retransmissions

If the timeout is too long (4*2.5):

Pros: in this case, packet need to resend due to fast transmit lost is not many, so total time is smaller.

Cons: slow reaction to segment loss and connection has lower throughput.

Appendix

3a)

Timeout = 2.5

pdrop = 0.1	pdrop = 0.3
-bash-4.1$ python receiver.py	-bash-4.1$ python receiver.py
arrive_seq : 5	arrive_seq : 9
arrive_seq : 0	arrive_seq : 0
arrive_seq : 0	arrive_seq : 0
ack_send back : 50	ack_send back : 50
arrive_seq : 50	arrive_seq : 50
ack_send back : 100	ack_send back : 100
arrive_seq : 150	arrive_seq : 150
ack_send back : 100	ack_send back : 100
arrive_seq : 200	arrive_seq : 200
ack_send back : 100	ack_send back : 100
arrive_seq : 250	arrive_seq : 250
ack_send back : 100	ack_send back : 100
arrive_seq : 300	arrive_seq : 300
ack_send back : 100	ack_send back : 100
arrive_seq : 350	arrive_seq : 350
ack_send back : 100	ack_send back : 100
arrive_seq : 400	arrive_seq : 400
ack_send back : 100	ack_send back : 100
arrive_seq : 450	arrive_seq : 100
ack_send back : 100	ack_send back : 450
arrive_seq : 500	arrive_seq : 100
ack_send back : 100	ack_send back : 450
arrive_seq : 550	arrive_seq : 100
ack_send back : 100	ack_send back : 450
arrive_seq : 100	arrive_seq : 600
ack_send back : 600	ack_send back : 450
arrive_seq : 100	arrive_seq : 650
ack_send back : 600	ack_send back : 450
arrive_seq : 100	arrive_seq : 750
ack_send back : 600	ack_send back : 450
arrive_seq : 600	arrive_seq : 800
ack_send back : 650	ack_send back : 450
arrive_seq : 600	arrive_seq : 850

ack_send back : 650	ack_send back : 450
arrive_seq : 650	arrive_seq : 450
ack_send back : 700	ack_send back : 500
arrive_seq : 700	arrive_seq : 450
ack_send back : 750	ack_send back : 500
arrive_seq : 750	arrive_seq : 950
ack_send back : 800	ack_send back : 500
arrive_seq : 800	arrive_seq : 500
ack_send back : 850	ack_send back : 550
arrive_seq : 850	arrive_seq : 1000
ack_send back : 900	ack_send back : 550
arrive_seq : 900	arrive_seq : 550
ack_send back : 950	ack_send back : 700
arrive_seq : 1000	arrive_seq : 1050
ack_send back : 950	ack_send back : 700
arrive_seq : 1050	arrive_seq : 1100
ack_send back : 950	ack_send back : 700
arrive_seq : 600	arrive_seq : 700
ack_send back : 950	ack_send back : 900
arrive_seq : 1100	arrive_seq : 1200
ack_send back : 950	ack_send back : 900
arrive_seq : 650	arrive_seq : 1250
ack_send back : 950	ack_send back : 900
arrive_seq : 1150	arrive_seq : 1300
ack_send back : 950	ack_send back : 900
arrive_seq : 1200	arrive_seq : 1350
ack_send back : 950	ack_send back : 900
arrive_seq : 1300	arrive_seq : 900
ack_send back : 950	ack_send back : 1150
arrive_seq : 1350	arrive_seq : 1400
ack_send back : 950	ack_send back : 1150
arrive_seq : 1400	arrive_seq : 1450
ack_send back : 950	ack_send back : 1150
arrive_seq : 950	arrive_seq : 1550
ack_send back : 1250	ack_send back : 1150
arrive_seq : 950	arrive_seq : 1150
ack_send back : 1250	ack_send back : 1500
arrive_seq : 950	arrive_seq : 1500

ack_send back : 1250 arrive_seq : 950 ack_send back : 1250 arrive_seq : 950 ack_send back : 1250 arrive_seq : 1450 ack_send back : 1250 arrive_seq : 1500 ack_send back : 1250 arrive_seq : 1550 ack_send back : 1250 arrive_seq : 1250 ack_send back : 1593 arrive_seq : 1250 ack_send back : 1593 arrive_seq : 1250 ack_send back : 1593 arrive_seq : 5 arrive_seq : 0	ack_send back : 1593 arrive_seq : 5 arrive_seq : 0

timeout = 3

pdrop = 0.1	pdrop = 0.3
-bash-4.1$ python receiver.py arrive_seq : 4 arrive_seq : 0 arrive_seq : 0 ack_send back : 50 arrive_seq : 50 ack_send back : 100 arrive_seq : 150 ack_send back : 100 arrive_seq : 200 ack_send back : 100 arrive_seq : 250 ack_send back : 100 arrive_seq : 300 ack_send back : 100	-bash-4.1$ python receiver.py arrive_seq : 1 arrive_seq : 0 arrive_seq : 0 ack_send back : 50 arrive_seq : 50 ack_send back : 100 arrive_seq : 150 ack_send back : 100 arrive_seq : 200 ack_send back : 100 arrive_seq : 250 ack_send back : 100 arrive_seq : 300 ack_send back : 100

arrive_seq : 350	arrive_seq : 400
ack_send back : 100	ack_send back : 100
arrive_seq : 400	arrive_seq : 450
ack_send back : 100	ack_send back : 100
arrive_seq : 450	arrive_seq : 100
ack_send back : 100	ack_send back : 350
arrive_seq : 500	arrive_seq : 100
ack_send back : 100	ack_send back : 350
arrive_seq : 550	arrive_seq : 600
ack_send back : 100	ack_send back : 350
arrive_seq : 100	arrive_seq : 650
ack_send back : 600	ack_send back : 350
arrive_seq : 100	arrive_seq : 700
ack_send back : 600	ack_send back : 350
arrive_seq : 100	arrive_seq : 800
ack_send back : 600	ack_send back : 350
arrive_seq : 600	arrive_seq : 350
ack_send back : 650	ack_send back : 500
arrive_seq : 600	arrive_seq : 350
ack_send back : 650	ack_send back : 500
arrive_seq : 650	arrive_seq : 900
ack_send back : 700	ack_send back : 500
arrive_seq : 700	arrive_seq : 500
ack_send back : 750	ack_send back : 550
arrive_seq : 750	arrive_seq : 1000
ack_send back : 800	ack_send back : 550
arrive_seq : 800	arrive_seq : 550
ack_send back : 850	ack_send back : 750
arrive_seq : 850	arrive_seq : 1050
ack_send back : 900	ack_send back : 750
arrive_seq : 900	arrive_seq : 1100
ack_send back : 950	ack_send back : 750
arrive_seq : 1000	arrive_seq : 1150
ack_send back : 950	ack_send back : 750
arrive_seq : 1050	arrive_seq : 1200
ack_send back : 950	ack_send back : 750
arrive_seq : 600	arrive_seq : 750
ack_send back : 950	ack_send back : 850

arrive_seq : 1100	arrive_seq : 1250
ack_send back : 950	ack_send back : 850
arrive_seq : 650	arrive_seq : 1300
ack_send back : 950	ack_send back : 850
arrive_seq : 1150	arrive_seq : 850
ack_send back : 950	ack_send back : 950
arrive_seq : 1200	arrive_seq : 1350
ack_send back : 950	ack_send back : 950
arrive_seq : 1300	arrive_seq : 950
ack_send back : 950	ack_send back : 1400
arrive_seq : 1350	arrive_seq : 1450
ack_send back : 950	ack_send back : 1400
arrive_seq : 1400	arrive_seq : 1500
ack_send back : 950	ack_send back : 1400
arrive_seq : 950	arrive_seq : 1600
ack_send back : 1250	ack_send back : 1400
arrive_seq : 950	arrive_seq : 1650
ack_send back : 1250	ack_send back : 1400
arrive_seq : 950	arrive_seq : 1750
ack_send back : 1250	ack_send back : 1400
arrive_seq : 950	arrive_seq : 1800
ack_send back : 1250	ack_send back : 1400
arrive_seq : 950	arrive_seq : 1400
ack_send back : 1250	ack_send back : 1550
arrive_seq : 1450	arrive_seq : 1400
ack_send back : 1250	ack_send back : 1550
arrive_seq : 1500	arrive_seq : 1400
ack_send back : 1250	ack_send back : 1550
arrive_seq : 1550	arrive_seq : 1900
ack_send back : 1250	ack_send back : 1550
arrive_seq : 1600	arrive_seq : 1950
ack_send back : 1250	ack_send back : 1550
arrive_seq : 1650	arrive_seq : 1550
ack_send back : 1250	ack_send back : 1700
arrive_seq : 1700	arrive_seq : 1700
ack_send back : 1250	ack_send back : 1850
arrive_seq : 1250	arrive_seq : 1850
ack_send back : 1750	ack_send back : 1954

arrive_seq : 1250 ack_send back : 1750 arrive_seq : 1250 ack_send back : 1750 arrive_seq : 1250 ack_send back : 1750 arrive_seq : 1250 ack_send back : 1750 arrive_seq : 1750 ack_send back : 1800 arrive_seq : 1750 ack_send back : 1800 arrive_seq : 1800 ack_send back : 1850 arrive_seq : 1900 ack_send back : 1850 arrive_seq : 1950 ack_send back : 1850 arrive_seq : 1750 ack_send back : 1850 arrive_seq : 1750 ack_send back : 1850 arrive_seq : 1850 ack_send back : 1954 arrive_seq : 2 arrive_seq : 0	arrive_seq : 2 arrive_seq : 0