Internet Protocols Home Work Assignment 3

1. P1

(a) Possible source and destination port numbers for the segment sent from A to S.

- The ports 1024 and below are used for standard services. However ports b/w 1024 - 65535 can be used for connection by clients.

So, port for server S = 23 A → S

Client A port can be b/w 1024 to 65535 (Eg 1555).

(b) Possible source and destination ports for segments sent from B to S. →

Server port is 23 B → S

Client B port will be b/w 1024 to 65535 (Eg 1556)

(c) Segment sent from S to A.

Server Port 23 S → A.

Client A Port 1555 (1024 to 65535)

(d) The segments sent from S to B.

server Port 23. S → B.

Client B port 1556 (1024 to 65535)

(e) If A and B are different hosts it is possible that the source port number in segments from A to S is same that from B to S.

The TCP connection will have different IP address so they can be easily differentiated.

(f) A Tcp connection is comprised of source IP address, source port number, destination IP address & destination Port number.

So its not possible that, If Client A & Client B are on the same host they will never get assigned the same port by the underlying operating system.

P3 UDP and TCP use 1's complement for their checksums.

0 1 0 1 0 0 1 1
0 1 1 0 0 1 1 0
0 1 1 1 0 1 0 0

1's complement of these 8-bit bytes →

Adding the first 2 8 bit bytes -

0 1 0 1 0 0 1 1
0 1 1 0 0 1 1 0
1 0 1 1 1 0 0 1

Adding this sum with the 3rd

1 0 1 1 1 0 0 1
0 1 1 1 0 1 0 0
1 0 0 1 0 1 1 0 1

0 0 1 0 1 1 0 1
1
0 0 1 0 1 1 1 0

To get 1's complement of this sum invert os & 1's

1 1 0 1 0 0 0 1

Why does UDP take the 1's complement —

Because the reciever adds up the checksum sent with the 8 bit bytes that will be sent. The sum of all this should be 11111111. If there is a zero somewhere in the middle. The reciever gets to know that there was an error.

No a one bits error can never go undetected while sending the data to the reciever.

The sum will reflect the 1 to become 0, or 0 to become 1

* Yes a 2 bit error can go undetected, when 2 bits are changed in such a way that their sum is the same. Eg -

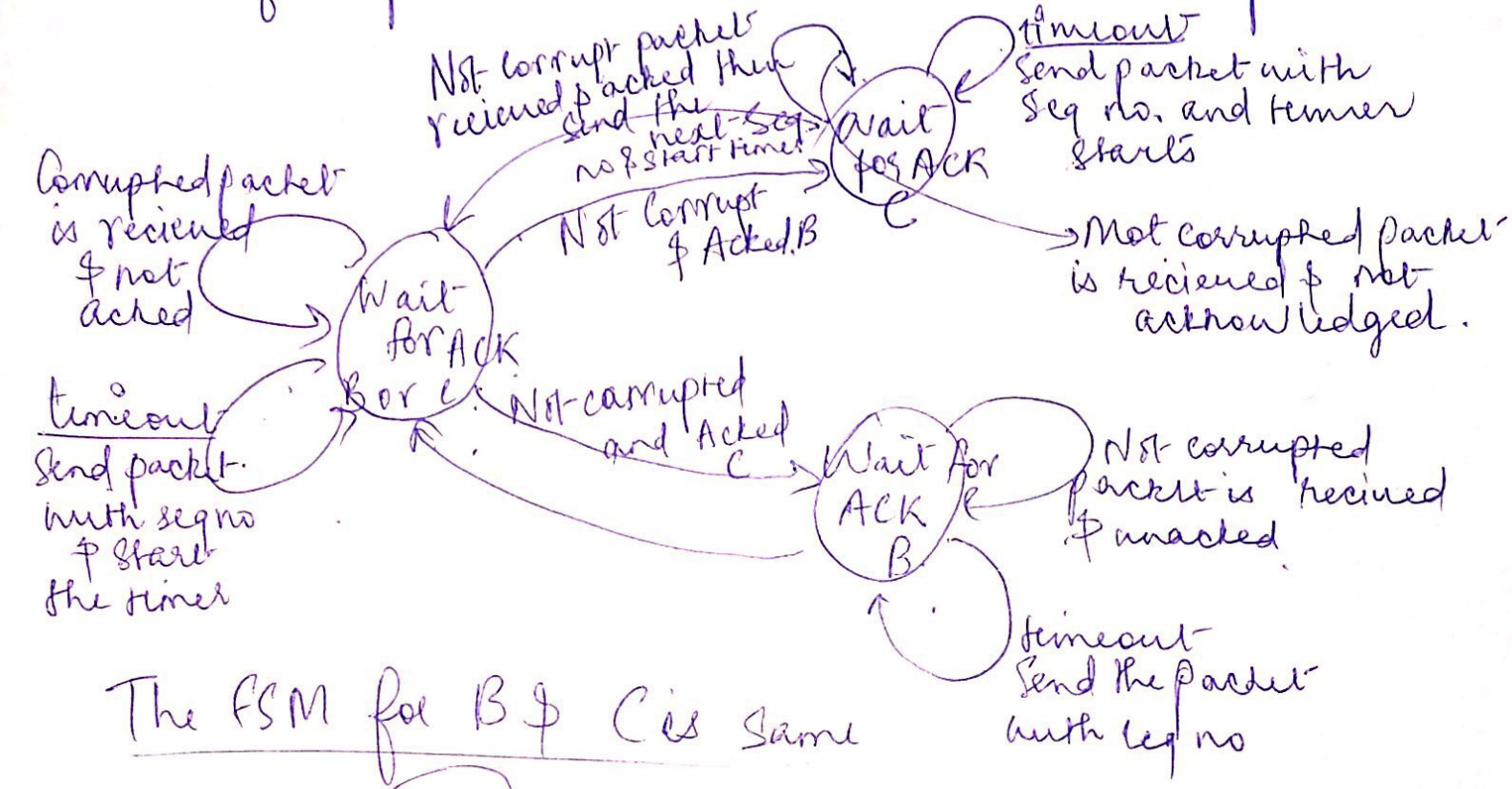
1001		1011	
1011		1001	
0100		0100	
	1		1
0101		0101	

In this case we can see that the bits have changed but the checksum is the same

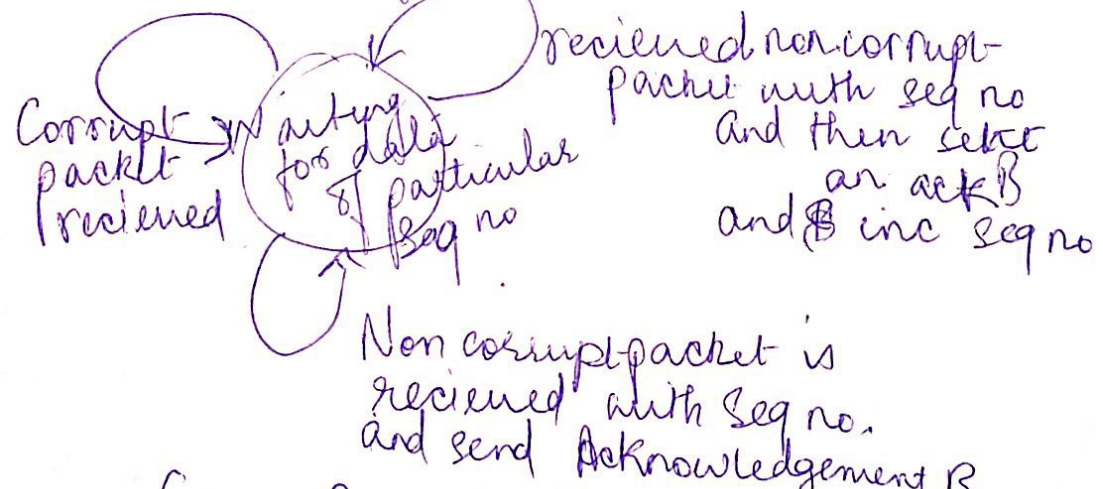
---

P6. Let us consider that the sender is in the state "Wait for call 1 from above" Reciever is in the state "Wait for 1 from below" when the first packet will be sent from sender with seq number 1. The sender will go into "Wait for Ack or Nac 1". When the reciever will recieve the packet it will send an acknowledgement. The reciever after sending the acknowledgement will move into the "Wait 0 from below" state. This means that the reciever is waiting for packet 0 sequence number. Lets say that the acknowledgement which was sent by the reciever was corrupted: The sender on recieving a corrupted acknowledgement will resend the packet with sequence of 1. Now the reciever gets the packet with seq no 1 and sends a NAK. The sender will kelp sending a packet with seq 1 and the reciever will keep sending NAK hence leading to a deadlock.

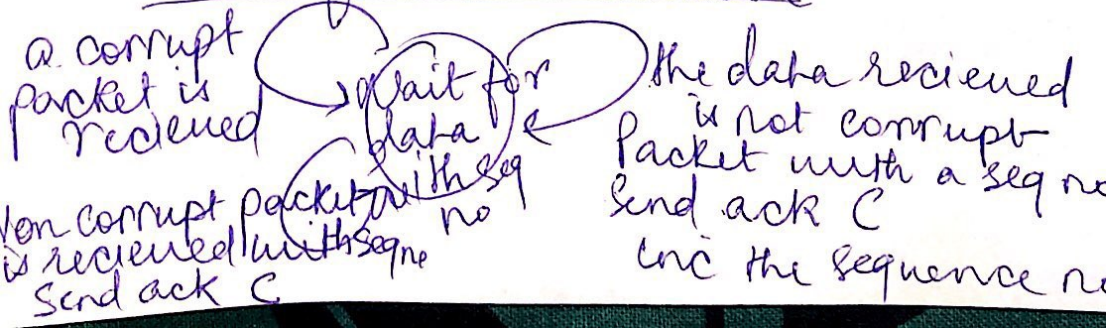
(P19) So when the data is being transmitted over a channel, the messages can be lost in the middle so it is an important thing to use sequence numbers for packets. to retransmit the lost packets.



The FSM for B & C is same



FSM for the Reciever C



P19 FSM explanation →

Sender FSM

- Starts with Seq no 0
- Starts the timer and sends the packet to B & C
- Then wait for ACK
- If there is no ack in a particular time then broadcast the packet to B & C again.

Reciver FSM —

- Wait to recieve an uncorrupted data packet from B & C with the same seq no as the acknowledgement no.
- If a packet's seq no is not = ack no, then loose the acknow and make the packet with ack 1
- If the packets seq no matches with the acknowledgement no. then send the ack no. to the Sender

P22 Consider the GBN protocol
Sender window size = 4
Sequence number range of 1024
at time t, the next inorder packet that reciever is expecting has a sequence number of K

(a) The possible sets of sequence numbers inside the senders window at time t ?

* Lets consider a case when all the acknowledgements have been recieved by the sender. In this case the sender window size will be [K, K+1, K+2, K+3]

* Now lets say in the other case none of the acknowledgements are recieved at the sender.

In this case the sender window will have N packets upto K-1 so it will have in its window

[K-4, K-3, K-2, K-1]

* Hence the sender window size is of length 4 & it has the following possibilities —

[K-4, K-3, K-2, K-1] [K-3, K-2, K-1, K] [K-2, K-1, K, K+1]

[K-1, K, K+1, K+2] [K, K+1, K+2, K+3]

(b) If the reciever is waiting for packet K. Then it has acknowledged K-1, K-2, K-3, K-4

If none of the acknowledgements are recieved by the sender, then the ACKs K-4, K-3, K-2, K-1 might still be propogating.

If its a case that the Sender has sent K-4, K-3, K-2, K-1 then it has recieved an ack for K-5.

Once the reciever has sent an ack for K-5, it will never send an ack less than that.

Hence the possible values of ack that can be sent by the reciever are

K-5, K-4, K-3, K-2, K-1.

P26 Transferring an enoromous file L Bytes from Host A to Host B. MSS = 536 Bytes
TCP seq num feild has 4 Bytes.

(a) What is the max val of L that TCP Seq no.'s are not exhausted?

The possible sequence numbers are 2^{32}.
The maximum file size that can be sent from A to B is simply 2^{32} Bytes as the Seq no increaments by the bytes of data sent..

(b) The number of Segments will be —

$$\left\lceil \frac{2^{32}}{536} \right\rceil = 8012999$$

How long it takes to transmit the file —

- header size getting added = 66 Bytes per segment
  So $66 \times 8012999 = 528857934$
- Size of file = 2^{32} Bytes.

$$\text{time} = \frac{4.824 \times 10^9}{155 \times 10^6} = 249 \text{ seconds}$$

P27

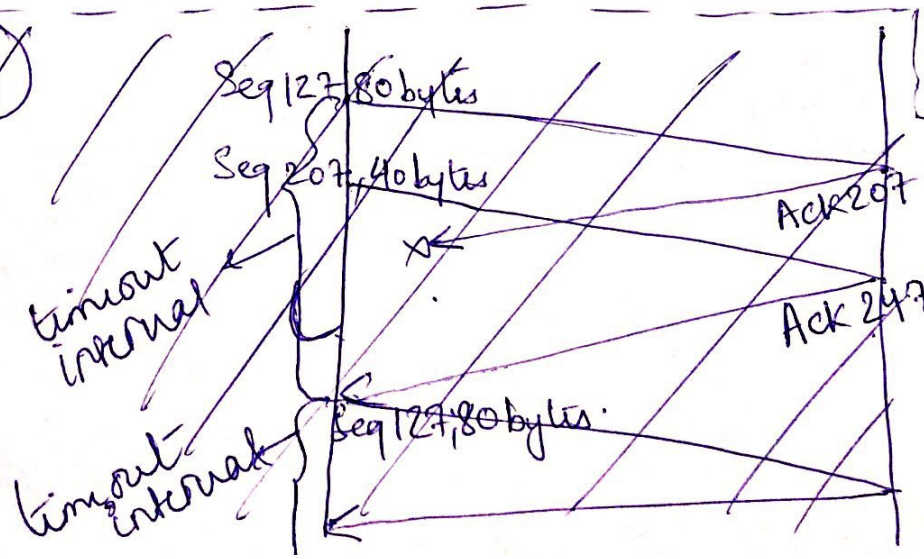
A & B Communicating over a TCP Connection
B has recieved all bytes up to byte 126
1st & 2nd segments have 80 & 40 bytes.
First Seg Seq no 127.
Source Port- 302
Destination Port- 80.

(a) Seq no = 127 + 80 = 207
Source port = 302
Destination port = 80

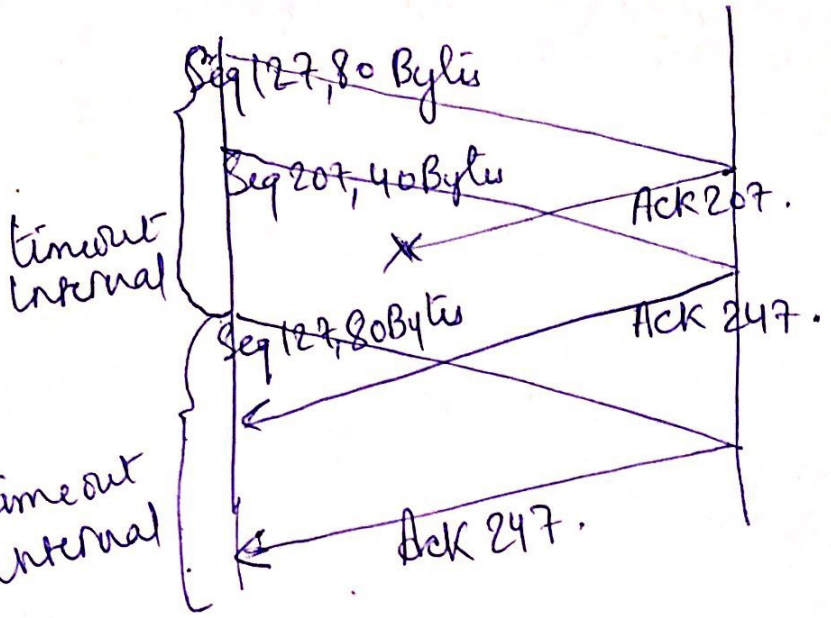
(b) The Acknowledgement no. = 207.
Source Port = 80
Destination Port- 302.

(c) If Second segment arrives before -
Acknowledgement no - 127. (Indicating it still waiting for this no.)

(d)



Please find on next Page.



P32 @ Estimated $RTT^{(1)}$ = Sample RTT_1 $\boxed{\alpha = 0.1}$

Estimated $RTT^{(2)} = x\,SampleRTT_1 + (1-x)\,SampleRTT_2$.

Estimated $RTT^{(3)}$ =

$$x \times SampleRTT_1 + (1-x)\left[x\,SampleRTT_2 + (1-x)\,SampleRTT_3\right]$$

$$= x\,SampleRTT_1 + (1-x)x\,SampleRTT_2 + (1-x)^2\,SampleRTT_3$$

Estimated $RTT^{(4)}$ =

$$x\,SampleRTT_1 + (1-x)\cdot Estimated\ RTT^3$$

$$= x\,SampleRTT_1 + (1-x)x\,SampleRTT_2 + (1-x)^2 x\,SampleRTT_3 + (1-x)^3\,SampleRTT_4$$

(b) $EstimatedRTT^{n} =$

$$\alpha \sum_{j=1}^{n-1} (1-\alpha)^{j} SampleRTT_j$$

(c) $EstimatedRTT^{(\infty)} =$

$$= \frac{\alpha}{1-\alpha} \sum_{j=1}^{\infty} (1-\alpha)^{j} SampleRTT_j$$

$$= \frac{1}{9} \sum_{j=1}^{\infty} (0.9)^{j} SampleRTT_j$$

P4 (a) TCP slow start is operating from intervals [1,6] and [23,26]
The client doubles the amount of packets sent each time till the reciever ~~recieves~~ sends an ACK meaning congestion detected or client's window limit is reached.

(b) Congestion avoidance is from [6,16] & [17,22] as we can see that the window is increasing additively.

(c) After the 16th transmission round, a segment loss is detected by a triple duplicate ack. If in case a timeout occurs the congestion window size is dropped to 1.

(d) After 22nd transmission round, segment loss is detected due to a timeout and in this case as well the congestion window will be set to 1.

(e) The value of threshold is set to half ~~in case~~ the value of congestion window in case packet loss is detected. Hence when loss is detected at 16 congestion window size is 42, which is reduced to 21 at 18

(e) The threshold is initially set to 32. Since it is at this size of window slow start stops and congestion avoidance phase begins.

(g) The threshold is set to half the value of the congestion window when packet loss is determined.

At 22 → Window Size = 26

At 24 → Window Size = 13.

(h)

1st Round.	1 packet
2nd	2, 3
3rd	4, 5, 6, 7
4th	8, 9, 10, 11, 12, 13, 14, 15
5th	16 to 31.
6th	32 to 63
7th	64 to 96.

Thus 70th packet in 7th round.

(i) The cwnd & threshold will be 1/2 the current cwnd (8). So new values of Threshold & cwnd = 4

(j) The cwnd size will be set to 1 first-
Then in 19th round it will increase to 4.

(k) 17 to 22. Packets sent.

17 - 1
18 - 2
19 - 4.
20 - 8.
21 - 16
22 - 21. So it will be total 52 Packets.