QUESTION 3

Part (a): Variation 1

(a) List clearly the actions taken by TCP when an acknowledgement for a segment is received. (5)

[ANS]

a. Window is adjusted as per flow control scheme, data transmitted if allowed
b. A new segment may be transmitted if data is waiting to be sent as per Nagle's algorithm
c. An already sent segment may be retransmitted if it is a duplicate ack
d. Congestion window may change as per congestion control policy (and which phase it is in) used
e. Timeout value is adjusted if it is not the acknowledgement of a retransmitted segment

[Grading remarks: (c) is not considered mostly as almost no one wrote it.]

Part (a): Variation 2

(a) List clearly the actions taken by TCP when a timeout occurs for a segment sent. (5)

[ANS]

a. Retransmission of the segment incurring timeout occurs
b. Congestion window and ss_thres is adjusted as per congestion control policy used, congestion phase adjusted
c. Timeout value is adjusted as per Karn's algorithm

[Grading remarks: 2 each for mentioning the first two, 1 for the last. Also, many of you wrote that RTT is computed for timeout adjustment etc. Lost 0.5 as RTT is computed from segments that get the ack back, it is not something that is computed when a timeout occurs)]

Part (b): Variation 1

(b) Consider that an application using TCP periodically generates (makes a send() call) 200 bytes of data every 1 second for 6 seconds, generating the first 200 bytes at t = 0 and the last 200 bytes at t = 5. The MSS is 550 bytes, and the RTT is 2.7 seconds. Assume that processing times are negligible and there is no loss/timeout. For all TCP segments that are sent from the sender to the receiver, show the time the segment is sent and its size (in bytes, not including header). Give justifications for your answer. For each of the times t=0,

1,2,3,4,5, if no segment is sent, your justification should clearly state why no segment is sent even though data has been generated by the user. No marks will be given without proper justification. (10)

[ANS]
t = 0: 200 bytes generated, Segment 1 sent, size 200 bytes (first segment always sent in Nagle's)
t = 1: 200 bytes generated, nothing sent as ack pending
t = 2: 200 bytes generated, nothing sent as ack pending
t = 2.7: Ack of Segment 1 received, Segment 2 sent, size 400 bytes
t = 3: 200 bytes generated, nothing sent as ack pending
t = 4: 200 bytes generated, nothing sent as ack pending
t = 5: 200 bytes generated, Segment 3 sent with size 550 bytes (as MSS e)
t = 5.4: Ack of segment 2 received, nothing sent as ack of Segment 3 is pending
t = 8.1: Ack of Segment 3 received, Segment 4 sent with size 50 bytes as no pending ack

[Grading remarks: Not all lines are needed, as long as you answered about the segments sent and when not sent upto t=5, it is fine.
The question clearly mentions "no marks" will be given without proper justification. So if you have just drawn a picture, with no mention of why something is sent or not sent at certain times, I didn't give you 0 but you are heavily penalized over and above deductions for mistakes. Don't ask for it back, the question was very clear]

Part (b): Variation 2

Similar

QUESTION 4

Part (a): Variation 1

(a) Consider a TCP connection with MSS = 800 bytes. At the beginning, what would be a good way to choose ss_thres? Justify briefly. (5)

[ANS]
Since no information is available about the network capacity, set it to a very high value so that the sender tries to reach the network capacity very fast with the exponentially growing slow start phase. This is like probing the network to find what is the limit before congestion will happen, you want to do it fast. When a timeout occurs, that indicates the limit, and normal congestion control now kicks in with half the congestion window to slow down much before that from now on.

Part (a): Variation 2

(a) For the 3-duplicate ack scheme, is it better to choose a larger value than 3 (say 10) or smaller value (say 2) to trigger fast retransmit and fast recovery? Justify briefly. (5)

[ANS]
Choosing a lower value may not be able to distinguish well between segments that are lost and segments that are delayed and will arrive out-of-order. This may cause unnecessary retransmissions by fast retransmit. Choosing a much larger value may cause a truly lost packet from being detected as lost much later (when a timeout happens or say 10 duplicate acks), causing fast retransmit to kick in much later to retransmit the packet, the very problem that 2-duplicate acks and fast retransmit was supposed to solve (retransmit a lost packet before it actually times out). 3 is a just a trade-off number that is seen to work well in practice.

[Grading remarks: Ok if you got the essence and argued both way. One way argument will give you 3/5]

Part (b): Variation 1

(b) Consider a TCP connection using cumulative acknowledgements in which a sender A is sending data to a receiver B. The initial sequence no. of A communicated during connection establishment is 13210. Suppose that A has sent 4 segments with 880, 2120, 975, and 800 bytes of data respectively, and B has received all of them. A now sends another 4 segments S1, S2, S3, S4 with 780, 810, 1050, and 975 bytes of data respectively, at times t = 1, 2, 3, 4 seconds respectively. One way delay for a segment or an ack to be received is 1 second. Timeout is set at 3 seconds. S1, S2, and S4 are received in this order by B, but S3 is lost. S3 is retransmitted after timeout and is received by B correctly this time. Show all acknowledgement messages sent by B with the time at which they are sent from B and the sequence number in them. For each ack, justify why it is sent with those values. Assume that processing delay is negligible, no other segments or acks are lost, and any other types of timeouts (i.e. other than retransmission timeout) used in the system is very large. No marks will be given without proper justification. (10)

[ANS]
Total bytes sent in first 4 segments = 4775.
So sequence number of next byte = 13211 + 4775 = 17986 (connection establishment uses up 1 sequence no.)
t = 1: S1 sent with 780 bytes of data
t = 2: S1 reaches B, no ack sent as wait for one more segment (cumulative ack)
t = 2: S2 sent with 810 bytes of data
t = 3: S2 reaches B, ACK (S1-S2, 17986+780+810 = 19576) sent by B (combined ack for S1 and S2)
t = 3: S3 sent by A with 1050 bytes of data

t = 4: ACK (S1-S2, 19576) reaches A
t = 4: S4 sent with 975 bytes of data
t = 5: S4 reaches B, ACK(S4, 19576) sent again immediately as S3 is missing in the middle (duplicate ack)
t = 6: Timeout for S3 at A, also ACK(S4, 19576) reaches A. S3 retransmitted by A
t = 7: S3 reaches B, ACK (S3-S4, 19576 + 1050 + 975 = 21601) sent by B (as S3 completes the sequence from S1 to S4)
t = 8: ACK reaches A, end

[Grading remarks: The S1-S2, S4 in the ACK messages above are just for understanding, no segment id is actually sent. Also, ok if you have shown and argued for just the acks as asked in the question, I wrote the full sequence for understanding.
The question clearly mentions “no marks” will be given without proper justification. So if you have just drawn a picture, with no mention of why an ack is sent at a certain time with certain values, I didn’t give you 0 but you are heavily penalized over and above deductions for mistakes. Don’t ask for it back, the question was very clear]

Part (b): Variation 2

Similar