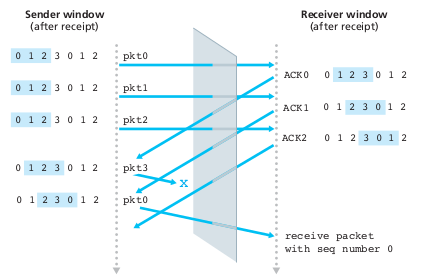
**Q1**

Q22 a) There can be two scenarios at the sender side, the cumulative acknowledgement for (k-1)th packet has been received (an ack value of k) i.e. no packets were lost and their acknowledgements also reached the sender. The second scenario is that the cumulative acknowledgement for (k-1)th packet was not received (an ack value of k) i.e. no packets were lost(as the receiver is expecting a packet with sequence number k) but their acknowledgements didn’t reach the sender. In the first scenario, the sender window is from k to k+3 since the window size is 4. In the second scenario, the sender window is from k-4 to k-1. This is because the sender hasn’t reached the acknowledgements for the last 4 packets it sent (k-4, k-3, k-2 and k-1).

Q22 b) Considering that the receiver is expecting packet number k, it has received and acknowledged packets k-1 and all the packets before that. Considering the scenario described in part a), if the acknowledgements weren’t received by the sender which are k-4 till k-1, these would be in flight. Since k-4 is the smallest packet number whose ACK hasn’t been received by sender, this means that k-5 packet has been received and acknowledged. However, if some packet is lost and an out-of-order packet is sent; GBN sends the ACK of last acknowledged packet again. Using this intuition, we can say that the ACK values currently propagating back to the sender at time t are k-5 to k-1.

Q23)



Considering a scenario where the receiver has a figurative curtain between the sender and the receiver, since the receiver cannot “see” the actions taken by the sender. All the receiver observes is the sequence of messages it receives from the channel and sends into the channel. There is no way of distinguishing the retransmission of the first packet from an original transmission of the fifth packet. Clearly, a window size that is 1 less than the size of the sequence number space won’t work. The window size must be less than or equal to half the

size of the sequence number space for SR protocols.

Q25) a) TCP has a send buffer in which the application puts data, a TCP segment may contain no or multiple application-layer messages because if TCP is used, bytes will be grabbed without surely putting a single application-layer message in the segment. This gives an application less control compared to UDP which will only encapsulate and send whatever the application gives it.

Q25) b) UDP has no flow or congestion control protocols which can add delays in transmittal of segments, TCP has both mechanisms which increases the overhead of sending messages.

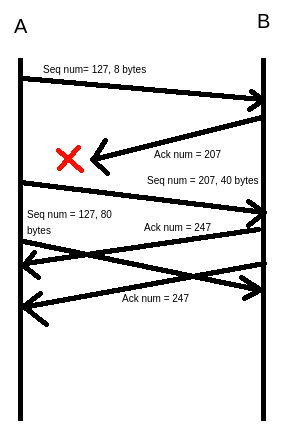
Q26) a) Data sent through TCP is split into byte size portions so sequence number increases by the number of bytes sent over the network. The maximum size L would be the number (of bytes) which can fit in the sequence number field of TCP segment represented by 2^32 which is nearly 4.2 GBs.

Q26) b) Since L = 2^32 and MSS = 536 bytes, the number of segments needed to transmit L is equal to 2^32/536 = 8012999 segments. For each segment, 66 bytes of header is added so for 8012999 segments, 66\*8012999 bytes would be added which is 528857934 bytes. The total number of bytes would be equal to the sum of header bytes and L which is 4823825230 bytes. Dividing this answer with the speed i.e. 155 Mbps gives a time of 248.9 seconds.

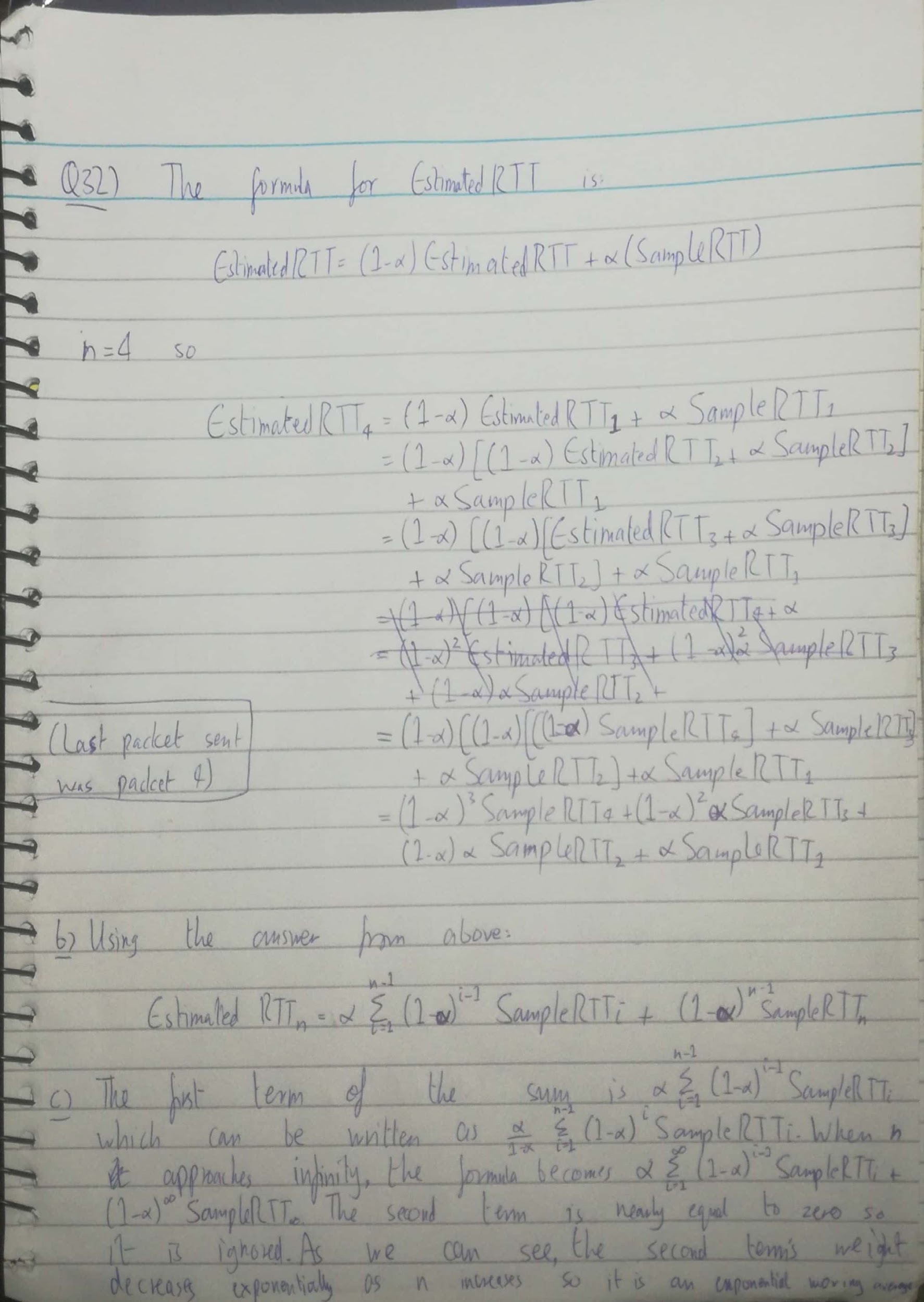
Q27) a) Sequence number = 207, source port number = 302 and destination port number = 80.

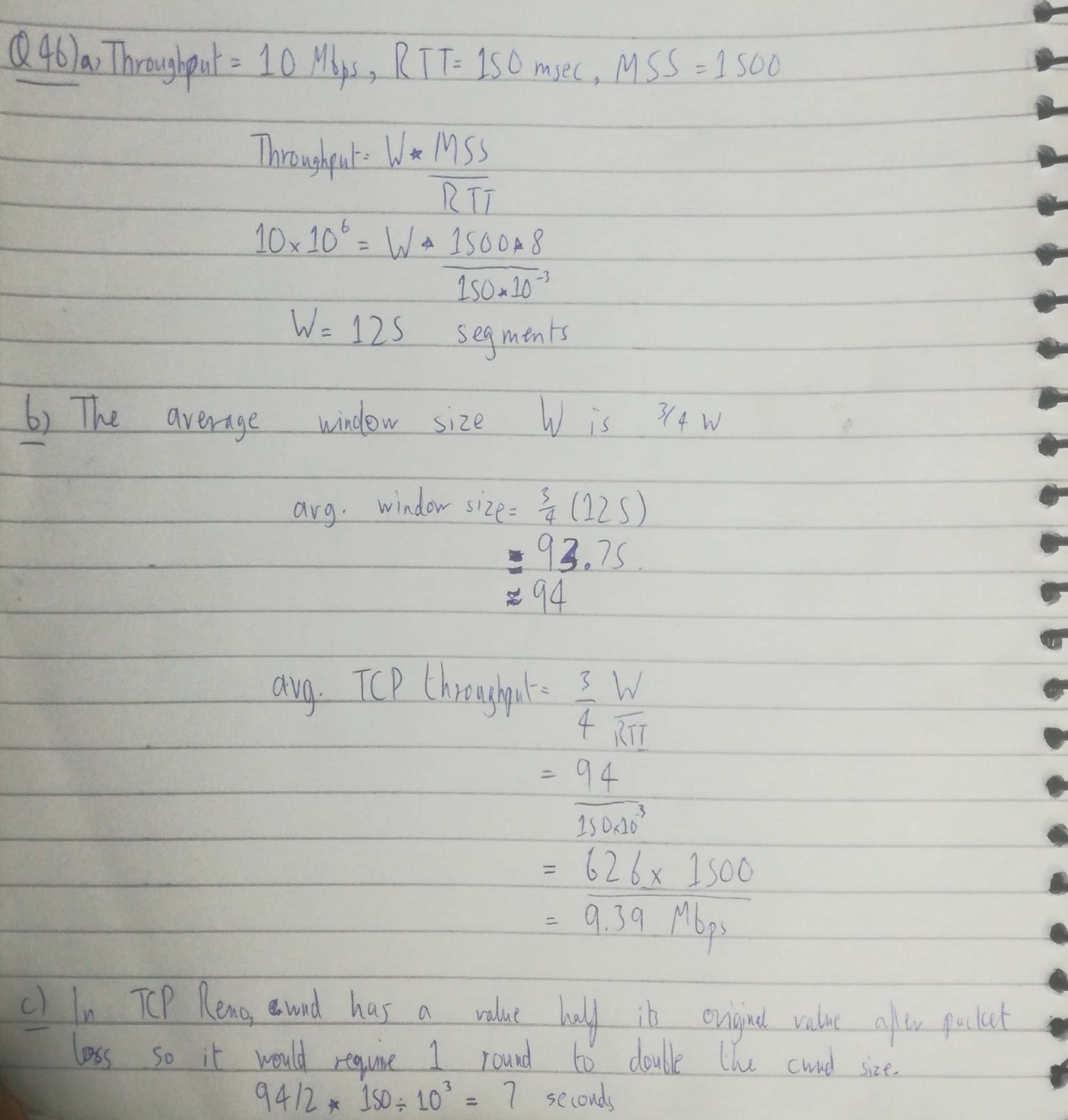
Q27) b) Acknowledgement number = 207, source port number = 80 and destination port number = 302.

Q27) c) Acknowledgement number = 127.

Q27) d) 

Q28) The capacity of the link hinders A’s sending rate which can be at max be 100Mbps. Due to the discrepancies in speed, host A can send data faster than host B can extract it from it’s receive buffer, as a result about 40-45Mbps filling rate is achieved. Now host B will set rcvWindow = 0, when the buffer is full to indicate host A to stop sending data. A will only resume when the value of rcvWindow becomes greater than 0 again. Flow control happens by changing this value which makes A send and stop sending data to B.





**Q2**

Q1) The underlying assumptions of TCP are as below:

1. Short RTT: The time taken for a signal to be sent and acknowledgement received by sender should be short.
2. Abundant energy of node: able to send packet over network
3. Exchange signalling between source and destination: three-way handshake
4. Best-effort service: Packet loss is less, it ensures packet reaches destination and in-order
5. Symmetric data transmission: data speed is the same in both directions
6. Low error rate: packet loss should be low

Q2) With TCP, the transaction is accomplished by connecting to the server (three-way handshake), requesting the file (**GET *file***), then closing the connection (sending a FIN segment). T/TCP would operate by connecting to the server, requesting the document and closing the connection, all in one segment (TAO). It is obvious that bandwidth is saved by this method. (taken from <https://www.linuxjournal.com/article/3075>)

Q3) Forward Error Correction (FEC) is a technique used for controlling errors in data transmission, FEC is accomplished by adding redundancy to the transmitted information using a predetermined algorithm. Part of the data stream is used solely to correct errors in the downlink stream from the satellite. This prevents the picture breaking up. A FEC of 2/3 means that every third bit of data is used to correct errors in the previous two bits. Here signal would be more robust and can be received easier, with less breakup in rain, for example. It is less efficient, with not so much data available to transmit the picture. (taken from <https://www.astra2sat.com/forward-error-correction-fec/>)

Q4) A key feature of TCP is the congestion control algorithm, constructed with the assumption that packet loss is normally very low, and that packet loss therefore is a sign of network congestion. This holds true for wired networks, but for mobile wireless networks non-congestion related packet loss may appear. The varying signal power inherent with mobility and handover between base-stations are two example causes of such packet loss. Another problem is packet reordering where the receiving order of a flow of packets differs from its sending order. Persistent and substantial packet reordering violates the (near) in-order channel assumption. It results in substantial degradation in application throughput and network performance.

**Q3**

Q1)

Q2) Due to the use of round-trip times measurement, the window dynamics of TCP Vegas are much more stable than those of TCP Reno, resulting in a much more efficient utilization of the network resources. In addition, whereas TCP Reno discriminates against users with long propagation delays, TCP Vegas fairly shares the available bandwidth between users, whatever their propagation delays. TCP Vegas enhances the congestion avoidance algorithm of TCP Reno. In essence, TCP Vegas dynamically increases/decreases its sending window size according to observed RTTs (Round Trip Times) of sending packets, whereas TCP Reno continues increasing its window size until packet loss is detected.