Lab 8 – TCP Simulation

Consider a scenario where a sender (S) has '**n**' bytes of data to be sent to a receiver (R) which maintains a buffer of size '**m**' from which the data is read at a rate of 1 byte for every '**k**' bytes received successfully. S sends the data as multiple packets, as the amount of data that can be sent at a time is limited by the maximum segment size (**MSS**). S uses sliding window protocol to determine the number of packets that can be sent at a time before receiving an acknowledgement from R. The size of the window is determined based on the advertised window sent by R and the congestion window maintained by S as follows:

maximum_window = minimum(congestion window, advertised window)

effective_window = maximum_window – {last_byte_sent – last_byte_acknowledged}

Initially, the congestion window starts with 1 MSS and doubles for every successful receipt of acknowledgement for a window of packets transmitted. In the case of non-receipt of acknowledgement, the congestion window is halved and later increases linearly on successful receipt of acknowledgement with the following increment:

increment = MSS * MSS / congestion window

Also, when the connection is initiated, the entire buffer in R is assumed to be free and hence the initial advertised window would be equal to 'm'. Eventually, as R receives more data from S, the size of the buffer decreases and hence the advertised window.

The **i^{th}** packet received by R is erroneous and hence acknowledgement is not sent by R for the packet.

Simulate this scenario using socket programming with S sending messages carrying the sequence number (starting with an initial sequence number (**ISN**) for the first packet transmitted) and the byte range of data to be sent in the packet. In return, R sends acknowledgements carrying the next expected byte and the advertised window for each received packet. On receiving an erroneous packet, R acknowledges the subsequent packets with the next expected byte as that sent for the previous packet. These duplicate acknowledgements can be taken as a hint to end the simulation. Print the messages received on either side. In addition, print the congestion window and effective window every time they are updated in S.

As a follow-up, simulate fast retransmission on receiving 3 duplicate acknowledgements and complete the simulation till the advertised window becomes 0 or till the entire data is transmitted, whichever occurs earlier.

Sample:

Input on S: n = 2500, MSS = 100, ISN = 0

Input on R: m = 2000, k = 500, i = 10

Simulation:

R sends ACK = -1, AW = 2000

1. S calculates effective_window = 100 and sends SEQ = 0, DATA = 0-99
1. R sends ACK = 100, AW = 1900
2. S calculates effective_window = 200 and sends SEQ = 100, DATA = 100-199

3. S sends SEQ = 200, DATA = 200-299
2. R sends ACK = 200, AW = 1800
3. R sends ACK = 300, AW = 1700
4. S calculates effective_window = 400 and sends SEQ = 300, DATA = 300-399
5. S sends SEQ = 400, DATA = 400-499
6. S sends SEQ = 500, DATA = 500-599
7. S sends SEQ = 600, DATA = 600-699
4. R sends ACK = 400, AW = 1600
5. R sends ACK = 500, AW = 1501
6. R sends ACK = 600, AW = 1401
7. R sends ACK = 700, AW = 1301
8. S calculates effective_window = 800 and sends SEQ = 700, DATA = 700-799
9. S sends SEQ = 800, DATA = 800-899
10. S sends SEQ = 900, DATA = 900-999
11. S sends SEQ = 1000, DATA = 1000-1099
12. S sends SEQ = 1100, DATA = 1100-1199
13. S sends SEQ = 1200, DATA = 1200-1299
14. S sends SEQ = 1300, DATA = 1300-1399
15. S sends SEQ = 1400, DATA = 1400-1499
8. R sends ACK = 800, AW = 1201
9. R sends ACK = 900, AW = 1101
11. R sends ACK = 900, AW = 1102
12. R sends ACK = 900, AW = 1102
13. R sends ACK = 900, AW = 1102
14. R sends ACK = 900, AW = 1102
15. R sends ACK = 900, AW = 1102