Department of Computer Science and Engineering,
Anna University, Chennai- 600025
CS23502- NETWORKS AND DATA COMMUNICATION
Practical
July- Oct 2026
Year/Sem/Batch : III/V/ N

Exercise: 7	TCP Congestion Control	18- 08- 2026

Observation questions:

1. What is congestion control in TCP, and why is it needed? Differentiate congestion control from flow control.

2. Explain the Slow Start algorithm of TCP with a neat cwnd-vs-RTT graph.

3. What is AIMD (Additive Increase, Multiplicative Decrease)? Explain the Congestion Avoidance phase.

4. Explain the Fast Retransmit and Fast Recovery mechanisms. How do they avoid an unnecessary return to Slow Start?

5.Compare TCP Tahoe and TCP Reno with respect to their reaction to packet loss.

Execution Questions

For each question, write a Java client–server program (Server.java as receiver, Client.java as sender) using TCP sockets. The client should print a round-wise table of cwnd, ssthresh, phase, and the triggering event for each RTT. Compile with javac and run the Server first, then the Client, on localhost.

1. Slow Start: Write a client–server program in which the client's congestion window (cwnd) starts at 1 and doubles every round (RTT) while cwnd < ssthresh (take ssthresh = 16), and increases by 1 every round once cwnd ≥ ssthresh. Assume no packet loss. Print the round, cwnd, ssthresh and phase (Slow Start / Congestion Avoidance) for each round, and observe the shape of the cwnd-vs-round curve.
2. Fast Retransmit: Modify Question 2 so that, instead of a timeout, the server simulates the receipt of 3 duplicate ACKs at pre-decided rounds. On this event the client must retransmit immediately (without waiting for a timeout), setting ssthresh = max(cwnd/2, 2) and cwnd = ssthresh (i.e. cwnd must NOT be reset to 1). Compare the resulting cwnd table with that of Question 2.
3. Fast Recovery: Extend Question 3 so that after the 3-duplicate-ACK event the client enters a Fast Recovery state: ssthresh = max(cwnd/2, 2), then cwnd = ssthresh + 3 (window inflation). Any further duplicate ACKs received while still recovering should increase cwnd by 1 each. When a new ACK is finally received, the client should exit Fast Recovery by setting cwnd = ssthresh (deflation) and resume normal Congestion Avoidance. Compare the smoothness of this recovery with Question 3.