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: 1	Applications using Echo Client and Server & Chat	09- 07- 2026

Observation questions:

1. What is the purpose of each of the four core socket system calls used on the server (socket, bind, listen, accept)? Which of these are *not* required on the client?

2. Why is htons() used on the port number before assigning it to sin_port?

3. What is the role of SO_REUSEADDR? What error do you get on re-running the server quickly without it?

4. Why is fork() needed in this program but not in the echo client-server exercise?
5. Explain the roles of the parent and child process after fork(). Which one sends and which one receives?

Execution Questions (Programming Language - C)

1: Echo Client and Echo Server

i. Compile and run the server and client. Record the exact console output on both sides for at least three exchanged messages.

ii. Change the port number in both programs, recompile, and verify the connection still works.

iii. Try connecting two clients to the same running server, one after another. Observe and explain what happens to the second client.

iv. Stop the server while the client is still running, then try sending a message from the client. Record and explain the error.

v. Modify the server so it converts every received message to uppercase before echoing it back. Show the output.

vi. Use netstat -tulnp or ss -tulnp while the server is running to confirm it is listening on the chosen port.
vii. Measure and record the round-trip latency of an echoed message.

2. Chat Application (Client-Server)

i. Compile and run the chat server and client in two terminals. Exchange at least five messages from each side and record the transcripts.
ii. Type exit from the client side and confirm both sides terminate the chat session gracefully. Record the output.
iii. Forcibly kill the server process while chatting, then try sending a message from the client. Observe and explain the resulting error.
iv. Modify the program so every message sent is timestamped before being displayed. Show sample output.
v. Run the client from a different machine on the same network by changing the server IP. Record whether the connection succeeds and any errors encountered.
vi. Use ps -ef while a chat session is active to confirm the parent and child process IDs, verifying the fork actually created two processes.
vii. Attempt to run two instances of the chat client connecting to the same server. Observe and explain the behavior.