Design Laboratory (CS69202)
Spring Semester 2025

Topic : Client-Server (Socket) **Date: Feb 5, 2025**

Important Instructions:

1. **Using Linux**
    a. This assignment is based on Network programming and will be beneficial for the students to do this assignment in a Linux OS.
    b. If you don't have linux, use virtualbox + linux image, both free. We suggest Ubuntu, but feel free to use any linux version.If you decide to use cygwin or wsl, it's up to you, just remember that most of your batch mates (and the instructors) will use Linux (and ubuntu), so, naturally, we might not be able to provide support for other systems.
2. **Programming language**
    a. This assignment is using the C programming language. You can also use cpp (without STL) if you want, as the important calls remain mostly the same.
3. **Error handling**
    a. A proper error handling (e.g., when the syscall or the library calls fail) is expected in this Assignment.

In Class Assignment

Section A (Basic TCP Client-Server)

You are tasked with implementing a simplified Client-Server messaging system using Transmission Control Protocol (TCP) in the C programming language. The objective is to enable the client to send messages to the server, with the server responding by sending acknowledgments. Your implementation should include both client and server components.

Part - I (Server Implementation)

Implement a Server that can handle incoming message requests from a single client using Transmission Control Protocol (TCP). The server should respond to each message from the client with an acknowledgment, ensuring reliable communication between the client and the server.

Requirements:

1. Create a TCP server that listens on a specific port and accepts a single client connection at a time.
2. Receive messages from the client, print them on the server side, and send an acknowledgment back to the client.
3. Close the connection gracefully when the client sends an exit command.

Part - II (Client Implementation)

Implement a client that sends messages to the server and receives acknowledgement from the server.

Requirements:

1. Connect to the server using the server's IP and port.
2. Allow the user to input messages and send them to the server.
3. Display the acknowledgment received from the server.
4. Implement a command (exit) to terminate the connection.

Section B (Extended TCP Client-Server)

You are tasked with implementing a simplified Client-Server messaging system using Transmission Control Protocol (TCP) in the C programming language. The objective is to enable multiple clients to send messages to the server concurrently, with the server responding to each message by sending an acknowledgment and log the client messages with IP addresses and timestamps.

Part - I (Server Implementation)

Implement a Server that can handle multiple incoming message requests from clients simultaneously using Transmission Control Protocol (TCP). The server should respond to each client's message with an acknowledgment.

Requirements:

1. Modify the server to handle multiple clients simultaneously.
2. Implement proper synchronization to protect shared resources, such as writing to a log file (log.txt) that stores client messages along with IP addresses and timestamps.

Part - II (Client Implementation)

Implement a client that sends messages to the server and receives acknowledgement from the server.

Requirements:

1. Use the existing client from Section A with minimal changes.
2. Test the client by connecting multiple instances to the server concurrently.

Take Home Assignment
Section - A (Ping command)

You are assigned to implement a simplified version of the ping command using Transmission Control Protocol (TCP) in the C programming language. The goal is to measure the round-trip time (RTT) between a client and a server. Your implementation should include both a client and a server.

Part - I (Server Implementation)

Implement a Server that can handle multiple incoming ping requests from clients. The server should respond to each client's ping request with an acknowledgment. The server should be able to handle concurrent ping requests from different clients.

Requirements:

4. Upon receiving a ping request, the server should respond with an acknowledgment containing the same payload included in the ping.
5. Implement a mechanism to handle packet loss or timeouts (e.g., retransmission of acknowledgment).
6. The server should log each incoming ping request, including the source IP address, port, and RTT in a file.
7. The server should be able to handle multiple ping requests simultaneously. You need to protect the logfile, so that multiple ping handler processes are not overwriting the file.

Part - II (Client Implementation)

Implement a client that sends ping requests to the server and measures the round-trip time. The client should be capable of sending multiple ping requests to the server.

Requirements:

1. Implement a mechanism to send ping requests to the server.
2. The client should measure the round-trip time (RTT)---the time difference between sending a ping request and receiving the acknowledgment.
3. Display the acknowledgment from the server along with the RTT.
4. Your ping client should allow users (from the command line) to specify the server ip, the number of ping requests, the interval between requests and the port to send the ping.

Section - B (FTP requests)

You are tasked with implementing a File Transfer Protocol (FTP) server and client in the C programming language using Transmission Control Protocol (TCP). The FTP system should support basic file transfer operations, including put, get, close, cd (change directory), and ls (list directory contents) from the client to the server as follows:

1. **put <file_name>**: Upload a file from the client to the server in the current directory.
2. **get <file_name>**: Download a file from the server to the client in the current directory.
3. **close**: Close the connection with the client.
4. **cd <directory_name>**: Change the current directory on the server.
5. **ls**: List all the contents of the current directory on the server available to the client.

Part - I (Server Implementation)

Implement an FTP server that can handle file transfer requests from multiple clients *concurrently*. The server should support the commands

Requirements:
1. The server should bind to a specific port to listen for incoming FTP commands.
2. The server should handle multiple concurrent connections.

Part - II (Client Implementation)

Implement an FTP client that allows users to connect to the FTP server and perform file transfer operations. The client should support the commands

Requirements:
1. Implement a mechanism to establish a connection with the server (using the known port).
2. Implement a simple user interface prompt (**ftp_client>**) for the client to enter FTP commands.
3. Display appropriate messages for success or failure of file operations.