Computing Laboratory I (CS69011)

Autumn Semester 2023

Assignment 9: Network programming

Assignment date: October 30, 2023

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.
4. **Input/output**
    a. The inputs should be taken from the Terminal and results displayed in the Terminal, **if explicitly not mentioned**.
    b. The IP address and port number will be provided by Command Line Interface (if required).
5. **Documentation**
    a. Include a README file explaining how to compile and run the server and client, any dependencies, and any additional features or improvements you made.
6. **Submission**:
    a. Please upload a single zip file with both the problems and any additional files. Please also include the README
7. **Tutorial**
    a. https://nikhilroxtomar.medium.com/tcp-client-server-implementation-in-c-idiot-developer-52509a6c1f59

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:

1. Upon receiving a ping request, the server should respond with an acknowledgment containing the same payload included in the ping.
2. Implement a mechanism to handle packet loss or timeouts (e.g., retransmission of acknowledgment).
3. The server should log each incoming ping request, including the source IP address, port, and RTT in a file.
4. 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.