University of Science and Technology of Ha Noi

Distributed System

Report Labwork 1

Dao Ngoc Tung - BA12-185

November 2024

Contents

1 Introduction

The target of the first practical lab work is transferring file from a client side to a server side over TCP/IP in CLI(Command Line Interface). This assignment plays an important role in understanding how two networks communicate through TCP protocol.

1.1 TCP and File Transfer over Sockets

TCP-Transmission Control Protocol-guarantees reliable, ordered, and error-checked delivery of data. File transfer over sockets involves the following:

- Establish a connection between the client and the server.
- Data transfer from the client and server.
- Properly handling connection closure and EOF (End of File).

2 Protocol Design

Below is the interaction diagram showing how the client and server communicate:

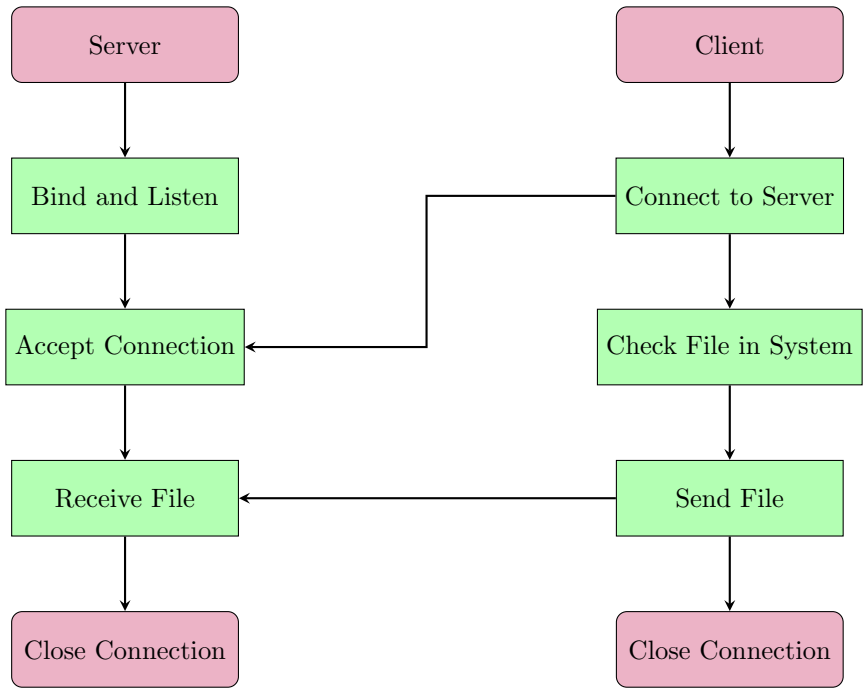

Figure 1: Client-Server Interaction for TCP File Transfer

2.1 Logic of the Protocol

1. The server is initiated by binding to an IP address and port, then waits for connections from clients.
2. The client starts the process by initiating a connection to the server.
3. Once connected, the server waits for a file, the client checks the file name in the system and sends the entered file to the server
4. After the file is fully transferred, both the client and server close the connection properly.

3 System Organization

3.1 System Architecture

The server and the client are 2 main components of the system. Their roles are described in the following:

- **Server:** Handles incoming connections and receive file from clients.
- **Client:** Connects to the server and sends files.

3.2 Architecture Diagram

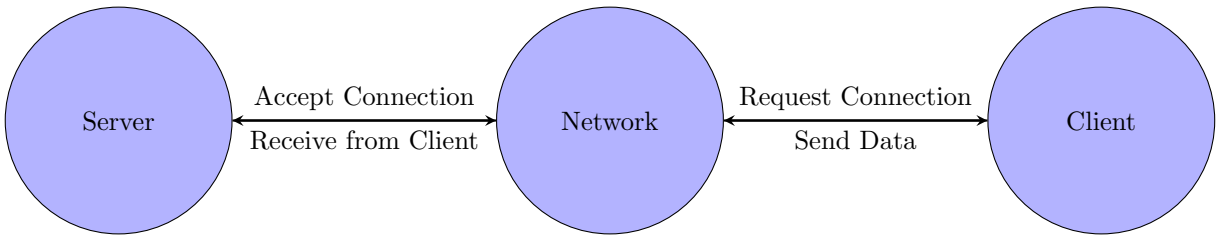

Figure 2: System Architecture for TCP File Transfer

4 Implementation

4.1 Key Code Snippets

4.1.1 Server Code

The server code is responsible for listening to connections and sending files:

```python
import socket
def server(host='127.0.0.1', port=13112):
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen(1)
    print(f"Server Listening On {host}:{port}")
    while True:
        conn, addr = server_socket.accept()
        print(f"Connected By Address: {addr}")
        namefile = conn.recv(512).decode()
        print(f"Client Requested File: {namefile}")
        try:
            with open(namefile, 'rb') as file:
                while (chunk := file.read(1024)):
                    conn.sendall(chunk)
            print("File Sent Completed!")
        except FileNotFoundError:
            conn.sendall(b"ERROR: File Not Found.")
            print("Not File Sent To Client")
        conn.close()

if __name__ == "__main__":
    server()
```

Listing 1: Server Code

4.1.2 Client Code

The client code connects to the server and receives the file:

```python
import socket
def client(server_host='127.0.0.1', server_port=13112, namefile='ngoctung.txt'):
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server_host, server_port))
    print(f"Connected To Server {server_host}:{server_port}")
    client_socket.sendall(namefile.encode())
    with open(f"received_{namefile}", 'wb') as file:
        while (chunk := client_socket.recv(1024)):
            if b"ERROR" in chunk:
                print(chunk.decode())
                break
            file.write(chunk)
    print(f"File Transfer Complete: received_{namefile}")
    client_socket.close()

if __name__ == "__main__":
    client(namefile='ngoctung.txt')
```

Listing 2: Client Code

5 Lab work execution

The lab work was executed by me. The server side was running at a PC using Window 10 and client side was executed using laptop running Ubuntu 24.04.

- **Window:** Run the server file and receive testwindow file from Ubuntu.
- **Ubuntu:** Run the client file and send the testwindow file to server

```
daongoctung@MacBook-Pro-cua-ao DS % python3 sev.py
Server Listening On 127.0.0.1:13112
Connected By Address: ('127.0.0.1', 57729)
Client Requested File: ngoctung.txt
File Sent Completed!
```

Figure 3: The server initiates connection and receive a file from the client

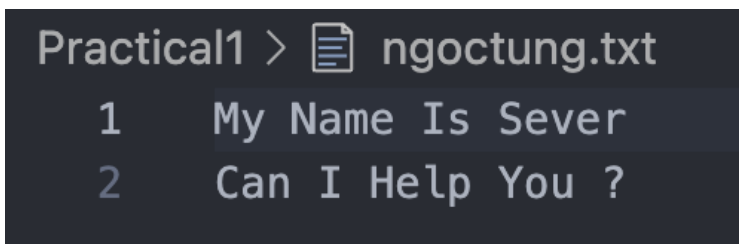

Figure 4: The received file is the same as the file in client side

6 Conclusion

This labwork demonstrates the implementation of client and server connection over TCP/IP by Python which ensures correctness of the data content.