University of Science and Technology of Ha Noi

Distributed System

Report Labwork 1

Tran Minh Phuong - 22BI13366

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:

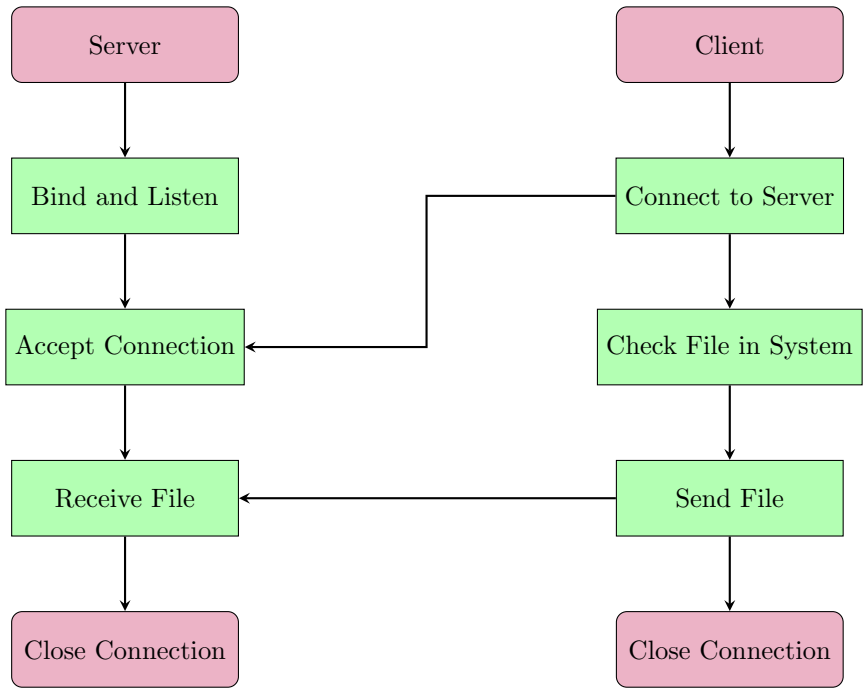

Figure 1: Client-Server Interaction for TCP File Transfer

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

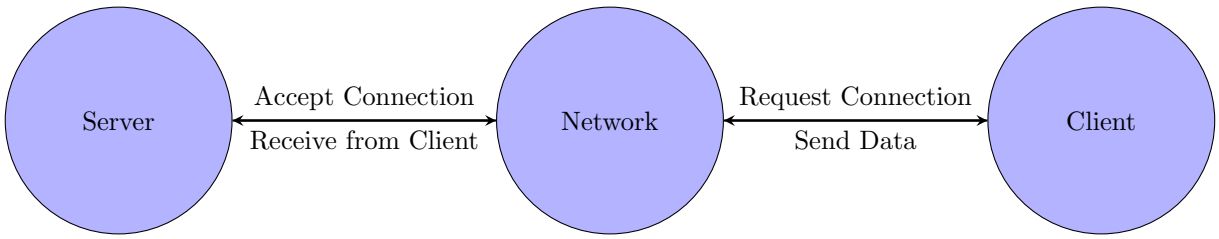

Figure 2: System Architecture for TCP File Transfer

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

HOST = input("Your server ip address?\n") #enter your host ip address
PORT = 2109  # port

def start_server():
    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"Connection established with {addr}")


        filename = b""
        while True:
            byte = conn.recv(1)
            if byte == b'\n':
                break
            filename += byte
        filename = filename.decode('utf-8')
        print(f"Receiving file: {filename}")


        with open(filename, 'wb') as file:
            while True:
                data = conn.recv(1024)
                if not data:
                    break
                file.write(data)

        print(f"File '{filename}' received and saved in the server folder.")
        conn.close()
        print(f"Connection with {addr} closed.")

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

Listing 1: Server Code

4.1.2 Client Code

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

```python
import socket
import os

SERVER_IP = input("Server's IP address you want to connect?\n")
PORT = 2109

def send_file():
```

```
    try:
        client_socket = socket.socket(socket.AF_INET, socket.
            SOCK_STREAM)
        print(f"Attempting to connect to {SERVER_IP}:{PORT}...")
        client_socket.connect((SERVER_IP, PORT))
        print(f"Connected to server at {SERVER_IP}:{PORT}")

        while True:
            filename = input("Enter the full file name (with
                extension): ")
            if os.path.isfile(filename):  # Ensure the file exists
                break
            print("File not found. Please try again.")

        client_socket.send(filename.encode('utf-8') + b'\n')  #
            Send filename with newline delimiter

        with open(filename, 'rb') as file:
            print(f"Sending file '{filename}' to the server...")
            while chunk := file.read(1024):  # Read in chunks of 1
                KB
                client_socket.send(chunk)

        print(f"File '{filename}' sent successfully to the server."
            )
        client_socket.close()

    except ConnectionRefusedError:
        print("Connection failed: Ensure the server is running and
            reachable.")
    except socket.timeout:
        print("Connection timed out: Server is taking too long to
            respond.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

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

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

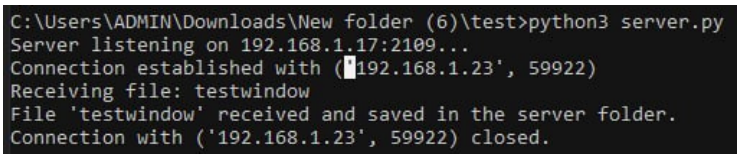

Figure 3: The server initiates connection and receive a file from the client

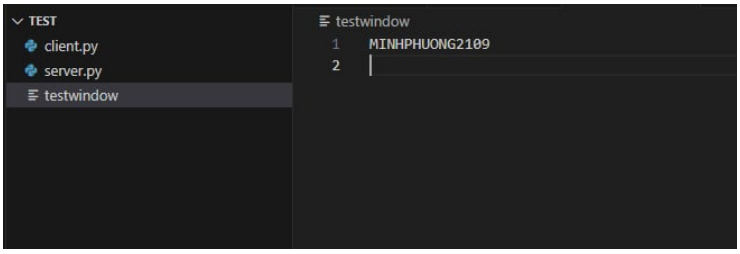

Figure 4: The received file is the same as the file in client side

6 Conclusion

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