Lab1

Ba12-104 Nguyen Ngoc Lan

November 30, 2024

1 Protocol Design

The file transfer protocol follows these steps:

1. The server listens on a specific port for incoming connections.
2. The client establishes a connection with the server.
3. The client reads the file in chunks and sends it to the server.
4. The server writes the received chunks to a file.
5. The connection is closed.

2 System Organization

The system comprises two components:

1. **Server:** Accepts incoming connections and writes the received data to a file.
2. **Client:** Connects to the server and sends the file data.

3 Implementation

The file transfer is implemented using Python sockets. Below are the key snippets:

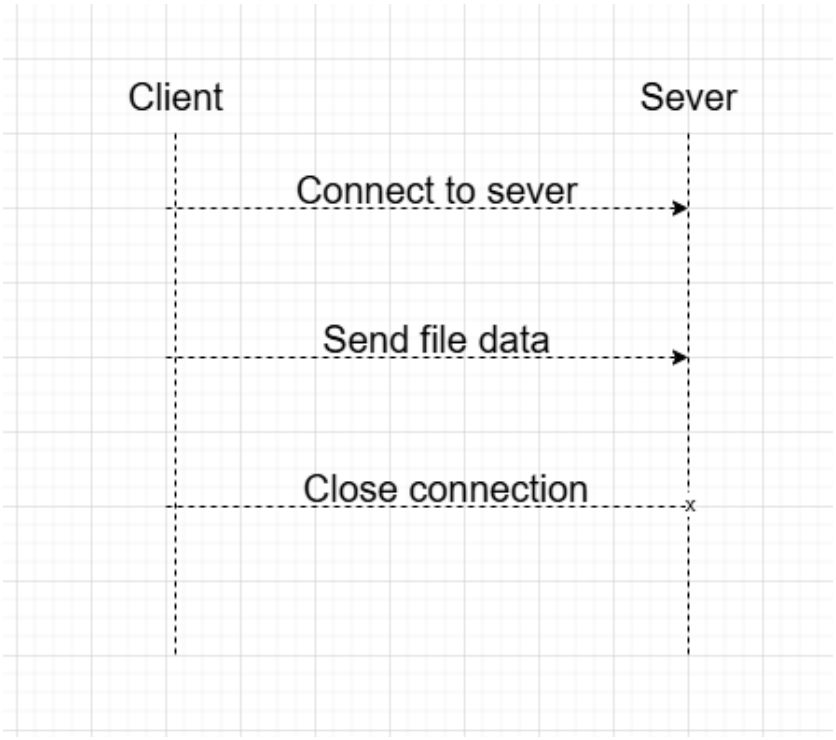

Figure 1: Protocol Design Flow

3.1 Client Code

Listing 1: Client code: send_file()

```
import socket

def send_file(server_host, server_port, file_path):
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server_host, server_port))
    print(f"Connected to {server_port}")

    with open(file_path, 'rb') as file:
        while chunk := file.read(1024):
            client_socket.send(chunk)

    print("File sent successfully!")
```

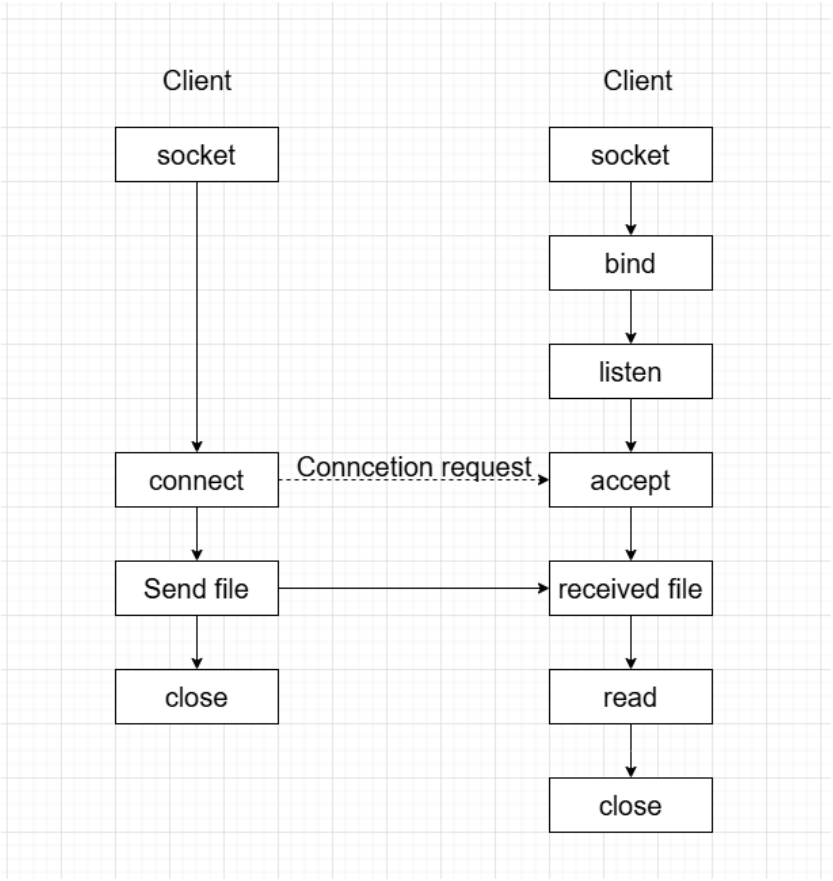

Figure 2: Client-Server Architecture

```
    client_socket.close()

if __name__ == "__main__":
    send_file("192.168.12.253", 6969, "transferfile.txt")
```

3.2 Server Code

Listing 2: Server code: start_server()

```
import socket

def start_server(host='0.0.0.0', port=6969):
```

```
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen(1)
    print(f"Server listening on {port}...")

    conn, addr = server_socket.accept()
    print(f"Connected by {addr}")

    with open("received_file.txt", 'wb') as file:
        while chunk := conn.recv(1024):
            file.write(chunk)

    print("File received successfully!")
    conn.close()
    server_socket.close()

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