TCP File Transfer

Pham Thai Son (22BI13397) Le Linh Long (22BI13262)
Nguyen Quang Huy (22BI13195)
Nguyen Tuan Khai (22BI13202)
Nguyen Hai Dang (22BI13073)

November 27, 2024

1 Protocol Design

The protocol is designed to enable file transfer between a client and a server over a TCP connection. The client initiates a connection to the server and sends commands to:

- Send a file to the server.
- Request a file from the server.
- End the session.

The server processes these commands, performs the requested operations, and sends responses accordingly. Figure 1 illustrates the protocol flow.

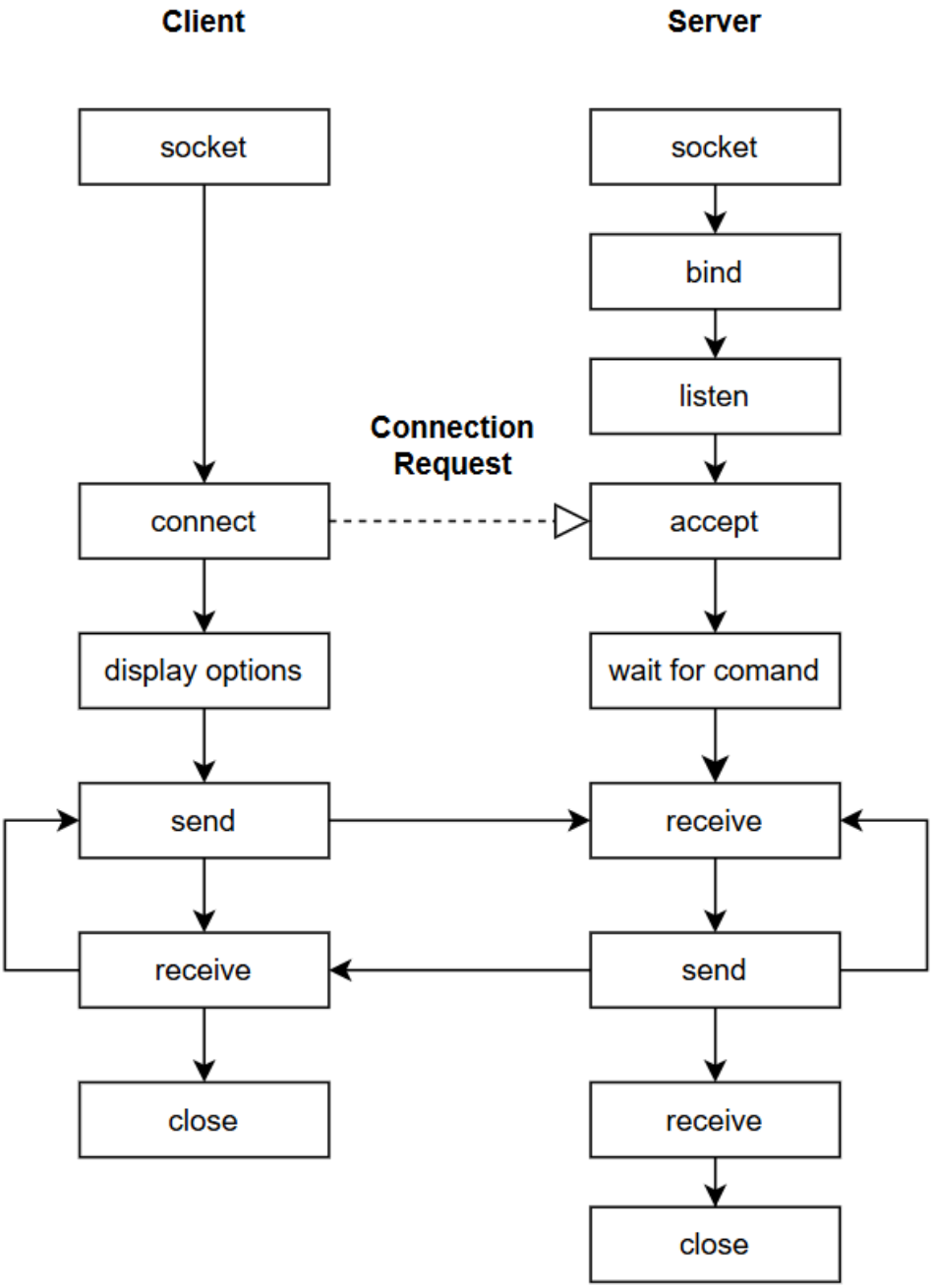

Figure 1: Protocol Design for TCP File Transfer

2 System Organization

The system is organized into two main components: the server and the client. Each has distinct responsibilities and uses specific functions to achieve file transfer via TCP.

2.1 Server-Side Functions

- **socket()**: Creates a TCP socket for the server to communicate with the client.
- **bind()**: Binds the socket to a specific IP address and port (`0.0.0.0:5000` in this case), making it accessible for incoming connections.
- **listen()**: Places the server in listening mode, waiting for clients to connect.
- **accept()**: Accepts an incoming connection from a client, returning a new socket for communication.
- **recv()**: Receives data from the connected client (e.g., commands, file content).
- **send()**: Sends data to the client (e.g., acknowledgments, file content).

2.2 Client-Side Functions

- **socket()**: Creates a TCP socket for the client to communicate with the server.
- **connect()**: Establishes a connection with the server at the specified IP address and port (`127.0.0.1:5000`).
- **send()**: Sends data to the server (e.g., commands, file content).
- **recv()**: Receives data from the server (e.g., responses, file content).

2.3 File Handling Functions

- **open()**: Opens files for reading (`'rb'` mode) or writing (`'wb'` mode) to handle file transfer.
- **os.path.exists()**: Checks if a file exists before attempting to send it.
- **os.path.getsize()**: Retrieves the size of a file for sending metadata to the client.

3 File Transfer Implementation

The file transfer functionality is implemented using Python's socket module. Below is a code snippet demonstrating how files are sent from the server to the client.

Server Code Snippet

```
def send_file(file_name, conn):
    if not os.path.exists(file_name):
        conn.send(f"ERROR: File '{file_name}' not found.". encode(
            'utf-8'))
        return
    conn.send(f"OK:{os.path.getsize(file_name)}".encode('utf-8'))
    with open(file_name, 'rb') as file:
        while (data := file.read(1024)):
            conn.send(data)
    conn.send(b'END')
```

Listing 1: Server File Sending Function

Client Code Snippet

```
def receive_file(file_name, conn):
    with open(file_name, 'wb') as file:
        while True:
            data = conn.recv(1024)
            if data == b'END':
                break
            file.write(data)
```

Listing 2: Client File Receiving Function

4 Team Contribution

The tasks were divided among team members as follows:

- **Pham Thai Son (22BI13397):** Designed the protocol flow and coordinated system integration.
- **Le Linh Long (22BI13262):** Implemented the server-side functions and error handling.
- **Nguyen Quang Huy (22BI13195):** Developed the client-side functions and user interaction menu.

- **Nguyen Tuan Khai (22BI13202):** Created the flowcharts and diagrams for protocol design and system organization.
- **Nguyen Hai Dang (22BI13073):** Conducted debugging, testing, and documentation.

5 Conclusion

This report demonstrates the implementation of a TCP-based file transfer system. The protocol, system organization, and Python implementation ensure reliable communication between the client and server for file operations.