Practical Work 1: TCP File Transfer

Vo Trong Giang - BI12-130

November 28, 2025

1 Introduction

In this practical work, I developed a Command Line Interface (CLI) application to transfer files between a Client and a Server using C sockets over TCP/IP. The system ensures reliability and follows a specific application-level protocol.

2 Protocol Design

Below is the text-based diagram illustrating the interaction between the Client and the Server. The protocol uses a handshake mechanism ("OK" signal) to ensure the server is ready before the file content is streamed.

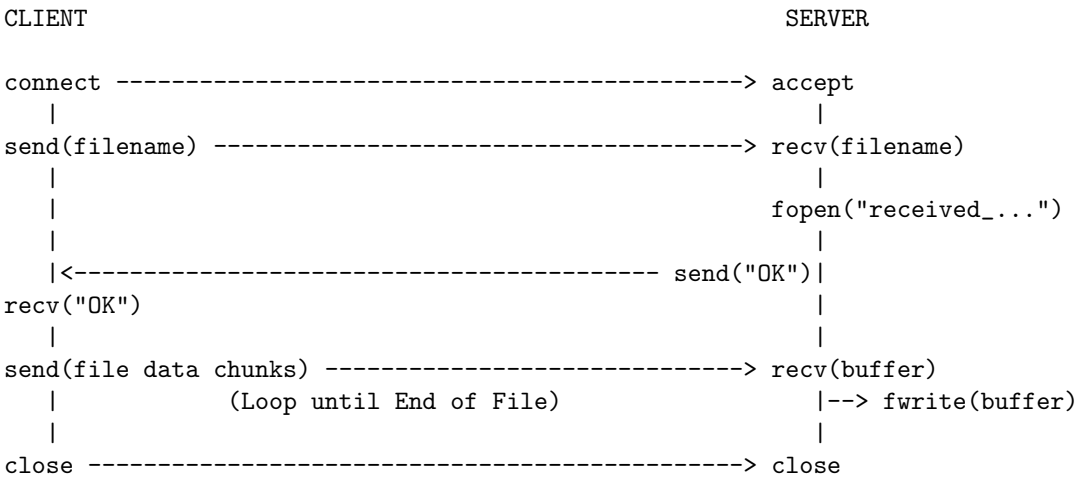

Figure 1: Protocol Sequence Diagram

3 System Organization

The system architecture follows the Client-Server model:

- **Server:** Binds to port 8080 and listens for connections. It saves received files with a "received_" prefix.
- **Client:** Connects to the Server's IP (127.0.0.1) and streams the file data.

4 Implementation Details

Below are the core code snippets demonstrating the transfer logic.

4.1 Server Side (Receiver)

```
// 1. Receive Filename
read(new_socket, buffer, BUFFER_SIZE);
sprintf(filename, "received_%s", buffer);

// 2. Send Acknowledgement
send(new_socket, "OK", 2, 0);

// 3. Receive Data Loop
FILE *fp = fopen(filename, "wb");
while ((valread = recv(new_socket, buffer, BUFFER_SIZE, 0)) > 0) {
    fwrite(buffer, 1, valread, fp);
}
fclose(fp);
```

4.2 Client Side (Sender)

```
// 1. Send Filename
send(sock, filename, strlen(filename), 0);

// 2. Wait for Server Readiness
read(sock, buffer, BUFFER_SIZE);
if (strcmp(buffer, "OK") == 0) {
    // 3. Send Data Loop
    while ((bytes_read = fread(buffer, 1, BUFFER_SIZE, fp)) > 0) {
        send(sock, buffer, bytes_read, 0);
    }
}
```

5 Conclusion

The implementation successfully transfers both text and binary files (like images). The use of the "OK" acknowledgment ensures the Server is ready before data is sent.