UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI
DEPARTMENT OF INFORMATION COMMUNICATION TECHNOLOGY

DISTRIBUTED SYSTEM REPORT

Practical Work 1: TCP File transfer

- - -

Le Vu Hoang Linh - BI12-243 - Cyber Security

March 28, 2024

Contents

1 Introduction

This report presents the outcomes of Practical Work 1 for the Distributed System course, focusing on the implementation of file transfer over TCP/IP within a CLI environment. In this work, a chat system that facilitates client-server communication through sockets for transmitting files was built using the C programming language.

2 Protocol design

TCP/IP consists of two primary protocols: the Transmission Control Protocol, which manages data flow between two endpoints, and the Internet Protocol, which handles the addressing and routing of packets.

TCP is a connection-based protocol that requires a stable connection before data transfer can occur. This connection is established through a three-way handshake process. Initially, a computer sends a SYN (synchronize) bit to the server to initiate a connection. The server acknowledges this by replying with a packet that contains both an SYN bit and an ACK (acknowledgment) bit. To complete the handshake, the initiating computer sends an ACK bit back to the server confirming that the connection has been setup successfully. With this process completed, data can be transmitted reliably. If data is lost or corrupted during transmission, it is re-sent, creating a connection that is effectively lossless (See Figure 1).

3 System organization

This practical work includes 2 files:

1. **client.c:**
    - Open listening Socket
    - Reads a text file
    - Sends the file's data to the Server
2. **server.c:**
    - Open listening Socket
    - Receives the data sent by the Client
    - Saves the data to a new text file

The process[1] is demonstrated in Figure 2.

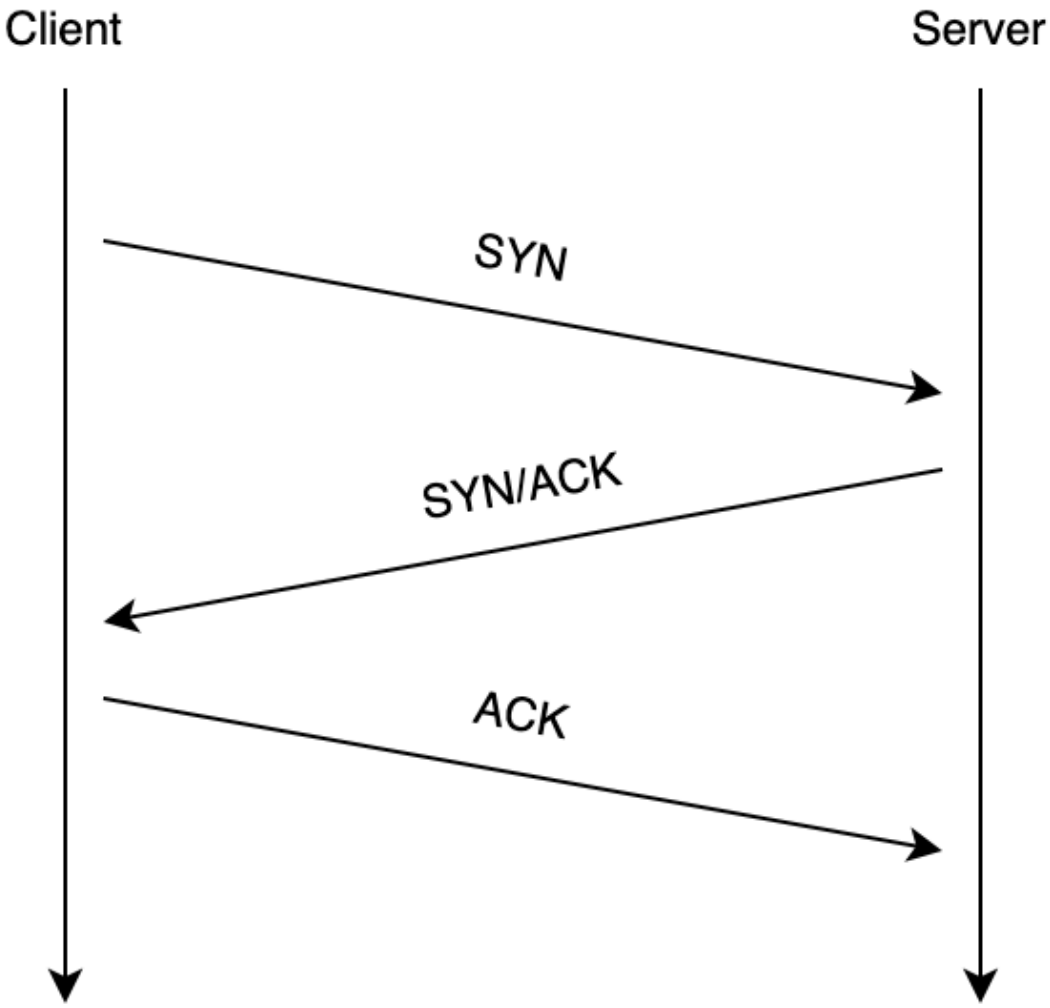

Figure 1: TCP Three-Way Handshake

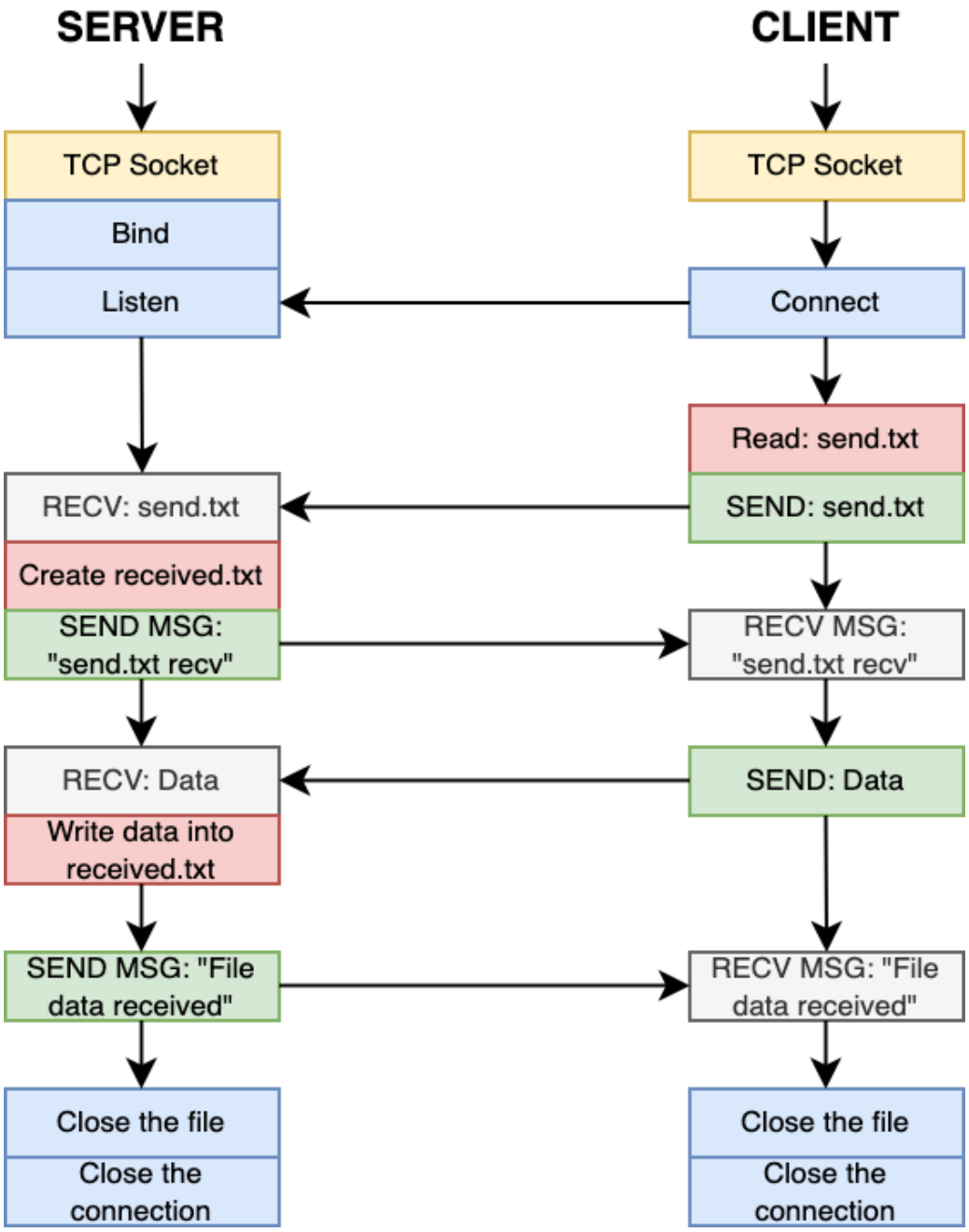

Figure 2: Overall procedure for the TCP file transfer

3.1 Server Setup

- Initializes a TCP socket.
- Binds the socket to an IP address and port (in this case localhost and port 8080).
- Listens for incoming client connections.

3.2 Client Setup

- Initializes a TCP socket.
- Connects to the server's IP address and port (in this case localhost and port 8080).

3.3 File Transfer Initiation

- The client reads the file `send.txt`.
- The client sends the file to the server.
- The server receives the filename `send.txt` and creates a new file `received.txt` to store the incoming data.

3.4 Confirmation and Data Transfer

- The server sends a confirmation message back to the client, indicating that the `send.txt` file has been received.
- The client starts sending the data to the server.
- The server receives the data and writes it into `received.txt`.

3.5 Transfer Completion

- Once the file data is fully transferred, the server sends a message "File data received" to the client.
- The client then closes the connection, completing the file transfer process.

4 File transfer implementation

4.1 Socket Creation

Both the client and the server start by creating a TCP socket using the `socket()` function. This socket acts as an endpoint for communication.

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

4.2 Connection Setup (Client) / Binding and Listening (Server)

4.2.1 Connection Setup (Client)

The client uses the `connect()` function to establish a connection to the server

```
e = connect(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr));
```

4.2.2 Binding and Listening (Server)

The server uses `bind()` to bind the socket to an IP address and port, and listen() to start listening for incoming connections.

```
e = bind(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr));
listen(sockfd, 10);
```

4.3 Accepting Connections (Server)

The server accepts an incoming connection using `accept()`, which returns a new socket file descriptor for the established connection.

```
int new_sock = accept(sockfd, (struct sockaddr*)&new_addr, &addr_size);
```

4.4 File Transfer (Client)

Before sending the file, the client opens the file using `fopen()` and then sends the file's data in chunks using a while loop and the `send()` function.

```
fp = fopen(filename, "r");
send_file(fp, sockfd);
```

4.5 File Reception and Writing (Server)

The server receives the data in chunks using a while loop and the `recv()` function, writing each chunk to the file using `fprintf()`.

```
write_file(new_sock);
```

4.6 Closing the Connection

Finally, after the file transfer is complete, both the client close the respective socket descriptors with `close()`.

5 Conclusion

The provided code demonstrates a straightforward implementation of a TCP-based file transfer between a client and a server in C (Figure 3), showcasing the core functions of socket programming for reliable communication.

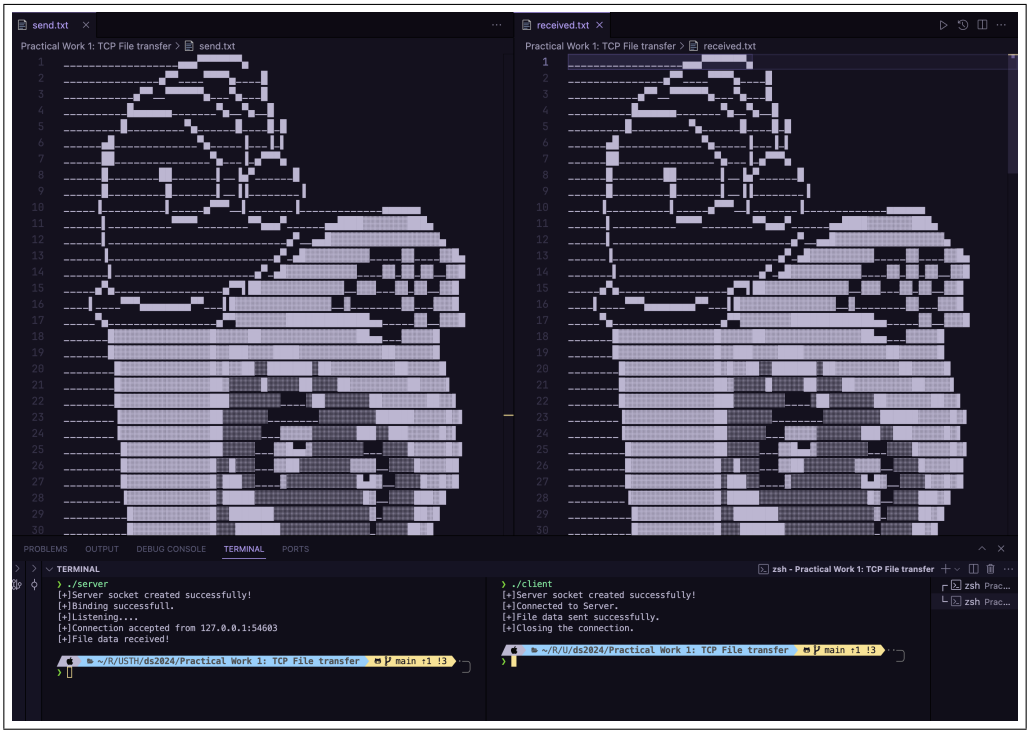

Figure 3: Results

References

[1] Nikhil RoxTomar. File Transfer Using TCP Socket in Python3. https://nikhilroxtomar.medium.com/file-transfer-using-tcp-socket-in-python3-idiot-developer-c5cf3899819c, 2021.