01.tcp.file.transfer.tex

inh Vũ Anh

November 2024

1 Introduction

This report describes a custom protocol for transferring files over TCP/IP in CLI, based on the provided chat system:

- One server
- One client
- Using socket

2 Protocol

2.1 Protocol Objectives

- File transfer over TCP/IP in the CLI: + Client is able to send a file to the server + Handle potential error (connection failed, file not found)

2.2 Protocol workflow

Step 1: Connection Establishment - Client connects to the server through a predefined IP and port - Server accepts the connections - A ”Connection established” message will appear on both ends if the connection between server and client is success

Step 2: File Transfer - Client reads the file in chunks and send each chunk to the server - Server receives the data and writes it to a file

Step3: Acknowledgment and Closure - Server and client both return a ”File received successfully” message or a ”Failed to receive file data” error message - Both client and server close their sockets

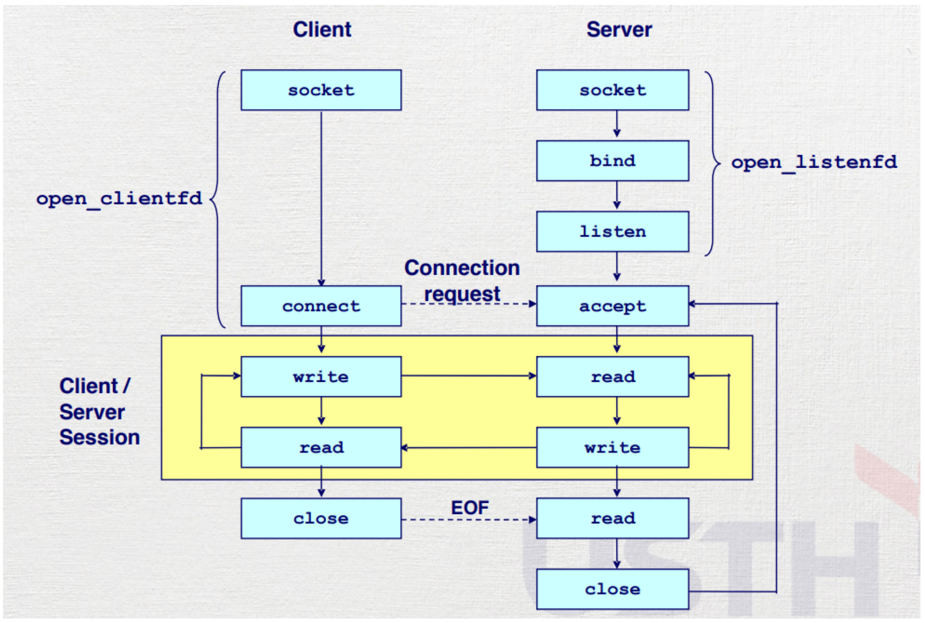

Figure 1: Protocol Diagram

2.3 Protocol Diagram

3 System Organization

3.1 Overview

- The system have two components: + **Server:** - Listen for client connections - Receive message and file data in chunks - Save the file to disk - Acknowledge completion + **Client** - Connect to ther server's IP and port - Read the file in chunks from the local disk and send it - Indicate the end of transfer (EOF) - Acknowledge completion

3.2 System Organization Diagram

4 Implementation

4.1 1. Server

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <unistd.h>
```

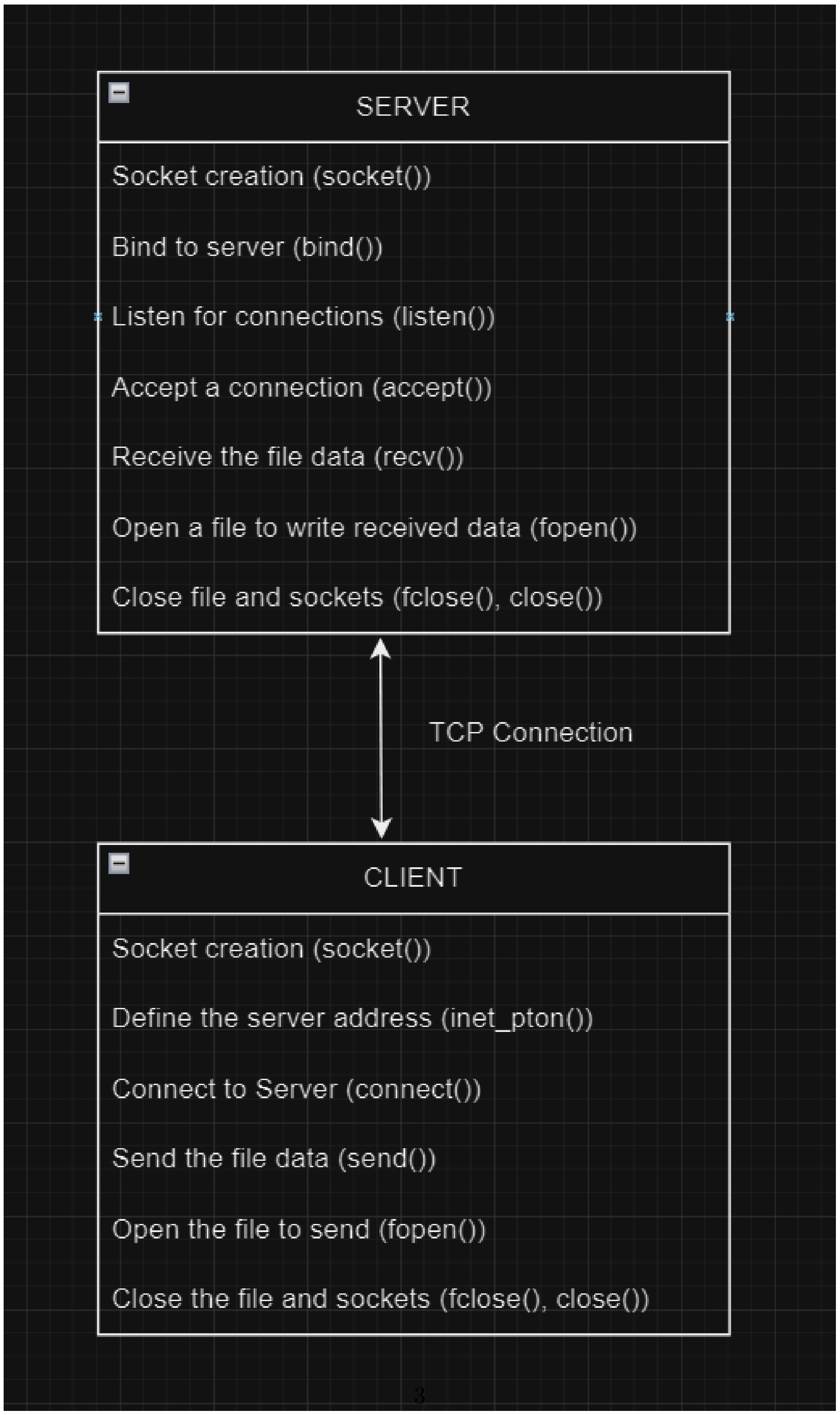
SERVER
Socket creation (socket())
Bind to server (bind())
Listen for connections (listen())
Accept a connection (accept())
Receive the file data (recv())
Open a file to write received data (fopen())
Close file and sockets (fclose(), close())
TCP Connection
CLIENT
Socket creation (socket())
Define the server address (inet_pton())
Connect to Server (connect())
Send the file data (send())
Open the file to send (fopen())
Close the file and sockets (fclose(), close())

Figure 2: System Organization Diagram

```
#define PORT 65432
#define BUFFER_SIZE 1024

int main() {
    int server_fd, client_fd;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_len = sizeof(client_addr);
    char buffer[BUFFER_SIZE];
    FILE *file;

    // Create a socket
    server_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (server_fd == -1) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // Set socket options
    int opt = 1;
    setsockopt(server_fd, SOL_SOCKET,
SO_REUSEADDR, &opt, sizeof(opt));

    // Bind the socket to an address
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr
*)&server_addr, sizeof(server_addr)) == -1) {
        perror("Bind failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    // Listen for incoming connections
    if (listen(server_fd, 1) == -1) {
        perror("Listen failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }
    printf("Server is listening on port %d...\n",
PORT);

    // Accept a connection
```

```
                client_fd = accept(server_fd, (struct
sockaddr *)&client_addr, &addr_len);
                if (client_fd == -1) {
                    perror("Accept failed");
                    close(server_fd);
                    exit(EXIT_FAILURE);
                }
                printf("Connection established with                        clie

                // Open a file to write the received data
                file = fopen("received_file.txt", "wb");
                if (!file) {
                    perror("Failed to create file");
                    close(client_fd);
                    close(server_fd);
                    exit(EXIT_FAILURE);
                }

                // Receive the file data
                ssize_t bytes_received;
                while ((bytes_received = recv(client_fd,
buffer, BUFFER_SIZE, 0)) > 0) {
                    fwrite(buffer, 1, bytes_received, file);
                }

                if (bytes_received == -1) {
                    perror("Failed to receive file data");
                    fclose(file);
                    close(client_fd);
                    close(server_fd);
                    exit(EXIT_FAILURE);
                }

                printf("File received successfully.\n");

                // Close the file and the sockets
                fclose(file);
                close(client_fd);
                close(server_fd);
                return 0;
            }
```

4.2 2. Client

```
            #include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <unistd.h>

#define SERVER_IP "127.0.0.1" // Loopback address
#define PORT 65432
#define BUFFER_SIZE 1024

int main() {
    int client_fd;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE];
    FILE *file;

    // Create a socket
    client_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (client_fd == -1) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // Define the server address
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    if (inet_pton(AF_INET, SERVER_IP,
&server_addr.sin_addr) <= 0) {
        perror("Invalid address or address not
supported");
        close(client_fd);
        exit(EXIT_FAILURE);
    }

    // Connect to the server
    if (connect(client_fd, (struct sockaddr
*)&server_addr, sizeof(server_addr)) == -1) {
        perror("Connection failed");
        close(client_fd);
        exit(EXIT_FAILURE);
    }
    printf("Connected to the server.\n");

    // Open the file to send
    file = fopen("sending_file.txt", "rb");
    if (!file) {
        perror("Failed to open file");
```

```
                close(client_fd);
                exit(EXIT_FAILURE);
            }

            // Send the file data
            size_t bytes_read;
            while ((bytes_read = fread(buffer, 1,
BUFFER_SIZE, file)) > 0) {
                if (send(client_fd, buffer, bytes_read,
0) == -1) {
                    perror("Failed to send file data");
                    fclose(file);
                    close(client_fd);
                    exit(EXIT_FAILURE);
                }
            }
            printf("File sent successfully.\n");

            // Close the file and the socket
            fclose(file);
            close(client_fd);
            return 0;
        }
```