Practical Work 1: TCP File Transfer

Nguyen Tien Ngoc

December 5,

2024

Overview

This practical work aims to implement a 1-to-1 TCP file transfer system using socket programming. The setup includes:

- One server to handle file transfers
- One client to request and receive the file
- Communication over TCP/IP

The following sections explain the protocol design, system organization, implementation, and team responsibilities.

Design Protocol

The file transfer protocol follows these sequential steps:

1. The **server** initializes and binds a socket to an address and port.
2. The server listens for incoming **client connections**.
3. The **client** initiates a connection request to the server.
4. Once connected, the client requests a file, and the server sends the file in chunks.
5. After completion, both the client and server close the

connection. The following diagram illustrates the protocol:

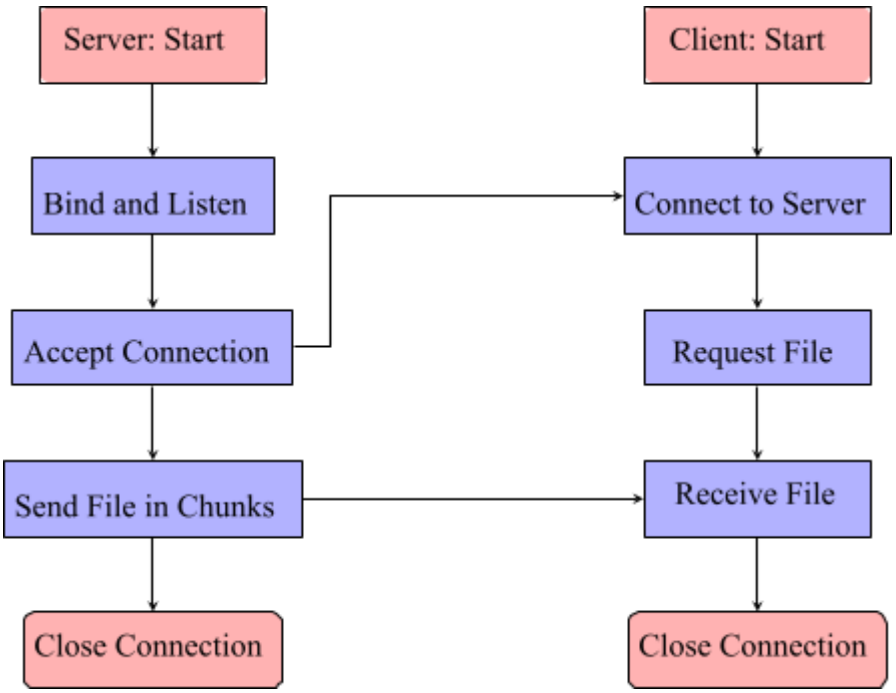

Figure 1: Protocol Design for File Transfer

System Organization

The system is organized using a client-server architecture. The following diagram shows the structure of the system:

Figure 2: System Organization: Client-Server Architecture

Implementation

The file transfer is implemented using C socket programming. Below are snippets for the server and client implementations.

Server Code

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

```
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define PORT 8080
int main() {
    int server_socket, client_socket;
    struct sockaddr_in server_addr, client_addr; socklen_t
    client_len;
    char buffer[1024];
    FILE *file_to_send;

    // Create socket
    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (server_socket < 0) {
        perror("Error opening socket");
        exit(1);
    }

    // Prepare the server address memset(&server_addr, 0,
    sizeof(server_addr)); server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    // Bind the socket
    if (bind(server_socket, (struct sockaddr *)&server_addr, sizeof(server_addr)
        perror("Binding failed");
        exit(1);
    }

    // Listen for incoming connections listen(server_socket,
    1);
    printf("Server listening on port %d...\n", PORT);

    // Accept connection from client
    client_len = sizeof(client_addr);
    client_socket = accept(server_socket, (struct sockaddr *)&client_addr, &clie
```

```
    if (client_socket < 0) { perror("Accept
        failed"); exit(1);
    }

    // Send file to client
    file_to_send = fopen("file.txt", "rb"); if
    (file_to_send == NULL) {
        perror("File not found"); exit(1);
    }

    while (fread(buffer, sizeof(char), sizeof(buffer), file_to_send) > 0) { send(client_socket,
        buffer, sizeof(buffer), 0);
    }

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

    // Close connections
    fclose(file_to_send);
    close(client_socket);
    close(server_socket);

    return 0;
}
```

Client Code

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define PORT 8080


int main() {


    int client_socket;
    struct sockaddr_in server_addr; char
    buffer[1024];
```

```
    FILE *file_to_receive;

    // Create socket
    client_socket = socket(AF_INET, SOCK_STREAM, 0); if
    (client_socket < 0) {
        perror("Error opening socket");
        exit(1);
    }

    // Prepare the server address memset(&server_addr, 0,
    sizeof(server_addr)); server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    // Connect to server
    if (connect(client_socket, (struct sockaddr *)&server_addr, sizeof(server_ad
        perror("Connection failed");
        exit(1);
    }

    // Receive file from server
    file_to_receive = fopen("received_file.txt", "wb"); if
    (file_to_receive == NULL) {
        perror("Unable to open file for writing"); exit(1);
    }

    int bytes_received;
    while ((bytes_received = recv(client_socket, buffer, sizeof(buffer), 0)) > 0 fwrite(buffer,
        sizeof(char), bytes_received, file_to_receive);
    }

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

    // Close connections
    fclose(file_to_receive);
    close(client_socket);

    return 0;
}
```

Roles and Contributions

- **Nguyen Tien Ngoc** **:** Designed the protocol and implemented the server code.
- **Team Member 2:** Developed the client code and tested the system.
- **Team Member 3:** Verified the error handling and documented the project.

Conclusion

The project successfully demonstrates a reliable 1-to-1 TCP file transfer sys- tem using C sockets. The implemented protocol ensures robust error handling and orderly connection management.