01.tcp.file.transfer.tex

Nguyen Khanh Duy

November 29, 2024

1 Protocol Design

The file transfer system is designed using a TCP-based protocol to ensure reliable and ordered delivery of data between the client and server. The communication between the sender and receiver is handled through a socket connection, where the sender transmits the file in chunks, and the receiver writes them to a file. The protocol design is illustrated in Figure 1.

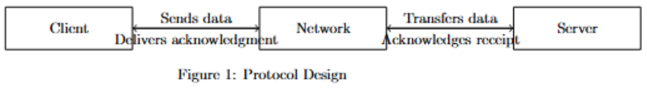

Figure 1: Protocol Design

2 System Organization

The system is organized into two main components: the sender (client) and the receiver (server). The client initiates the connection and sends the file, while the server listens for incoming connections, accepts the file, and stores it. The organization of the system is depicted in Figure 2.

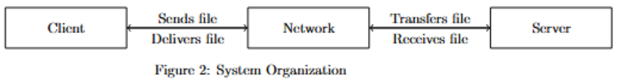

Figure 2: System Organization

3 File Transfer Implementation

The implementation of the file transfer utilizes Python's socket library to create both server-side and client-side programs. The server listens on a specific port, receives the file data in chunks, and writes it to disk. The client reads the file to be sent, connects to the server, and transmits the data. Below is the implementation code:

3.1 Server-Side Code

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

void send_file() {
    const char *server_ip = "192.168.110.5";
    int server_port = 21003;
    const char *filename = "2.txt";

    // open file for read data
    FILE *file = fopen(filename, "rb");
    if (file == NULL) {
        perror("Failed to open file");
        return;
    }

    // create socket
    int client_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (client_socket < 0) {
        perror("Socket creation failed");
        fclose(file);
        return;
    }

    // setup address server
    struct sockaddr_in server_addr;
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(server_port);
    if (inet_pton(AF_INET, server_ip, &server_addr.sin_addr) <= 0) {
        perror("Invalid address or Address not supported");
        close(client_socket);
        fclose(file);
        return;
    }

    // connect to server
    if (connect(client_socket, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("Connection failed");
        close(client_socket);
        fclose(file);
        return;
    }

    // read data from file and send by socket
    char buffer[1024];
    size_t bytes_read;
    while ((bytes_read = fread(buffer, 1, sizeof(buffer), file)) > 0) {
        if (send(client_socket, buffer, bytes_read, 0) < 0) {
            perror("Failed to send file");
            break;
        }
    }

    printf("File '%s' has been sent\n", filename);

    // close connection and file
    close(client_socket);
    fclose(file);
}

int main() {
    send_file();
    return 0;
}
```

3.2 Client-Side Code

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

void send_file() {
    char *server_ip = "192.168.110.5";
    int server_port = 8080;
    char *filename = "1.txt";
```

```
// create socket client
int client_socket = socket(AF_INET, SOCK_STREAM, 0);
if (client_socket < 0) {
    perror("Socket creation failed");
    exit(EXIT_FAILURE);
}

// setup address server
struct sockaddr_in server_addr;
memset(&server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(server_port);

// Convert IP address from string to network format
if (inet_pton(AF_INET, server_ip, &server_addr.sin_addr) <= 0) {
    perror("Invalid address or address not supported");
    exit(EXIT_FAILURE);
}

// connect to server
if (connect(client_socket, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
    perror("Connection failed");
    exit(EXIT_FAILURE);
}

// open file for read data
FILE *file = fopen(filename, "rb");
if (file == NULL) {
    perror("File not found");
    close(client_socket);
    exit(EXIT_FAILURE);
}

// read file and send data by socket
char buffer[1024];
size_t bytes_read;
while ((bytes_read = fread(buffer, 1, sizeof(buffer), file)) > 0) {
    if (send(client_socket, buffer, bytes_read, 0) == -1) {
        perror("Send failed");
        fclose(file);
        close(client_socket);
        exit(EXIT_FAILURE);
    }
}

printf("File '%s' has been sent\n", filename);
```

```
    // close file and  socket
    fclose(file);
    close(client_socket);
}

int main() {
    send_file();
    return 0;
}
```

4 Roles and Responsibilities

The roles and responsibilities in the file transfer process are as follows:

- **Sender (Client)**: The client initiates the connection to the server, reads the file to be transferred, and sends it in chunks.
- **Receiver (Server)**: The server listens for incoming connections, receives the file in chunks, and writes the data to disk.
- **Network**: The network layer ensures that the data is transferred reliably using TCP connections.