LABWORK 1: TCP File Transfer

Student's name: Nguyen Thai Duong

Student's ID: BI12-117

I. Protocol Design

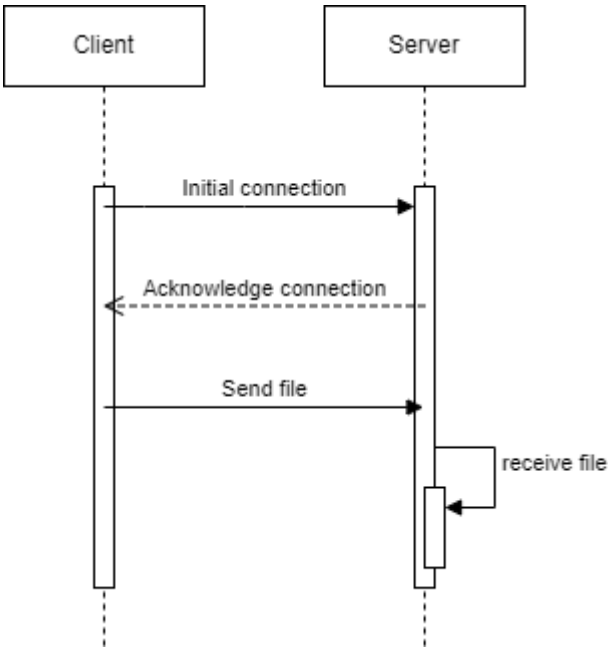

Figure 1.Protocol Design

II. System Organization

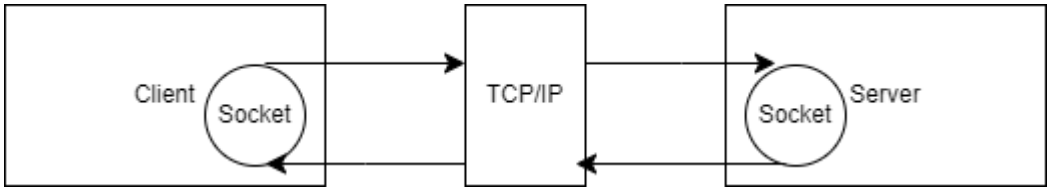

Figure 2.System Architecture

III. File Transfer

The server initializes and creates a file named "receive_file.txt" to save the data received from the client. It utilizes the "recv(new_socket, buffer, BUFFER_SIZE, 0)" function to accept the file from the client via the socket. Subsequently, employing "fwrite(buffer, sizeof(char), bytes_received, received_file)", the received data is stored into the previously generated "receive_file.txt".

```
// Receive data from client and write to file
char buffer[BUFFER_SIZE];
ssize_t bytes_received;
while ((bytes_received = recv(new_socket, buffer, BUFFER_SIZE, 0)) > 0) {
    ssize_t bytes_written = write(file_fd, buffer, bytes_received);
    if (bytes_written != bytes_received) {
        perror("Error writing to file");
        close(file_fd);
        close(new_socket);
        exit(EXIT_FAILURE);
    }
}
if (bytes_received == -1) {
    perror("Error receiving data from client");
    close(file_fd);
    close(new_socket);
    exit(EXIT_FAILURE);
}

printf("[+] Data received and written to 'test2.txt' successfully\n");

// Close file and socket
close(file_fd);
close(new_socket);

return 0;
```

Figure 3.Receive file in server side

Once the server is operational, the client initiates and establishes a connection with the server. Subsequently, it accesses the file to be transmitted, designated as "test.txt", via the "fopen("./test.txt", "rb")" function call. Following this, the content of the file is read, and the byte count is retained in the variable "bytes_read". Upon completion of the read operation, the data is transmitted over the socket (sock) utilizing the "send" function.

```c
// Open file to send
FILE *file_to_send = fopen("file_to_send.txt", "r");
if (file_to_send == NULL) {
    perror("File opening failed");
    exit(EXIT_FAILURE);
}

// Send file data to server
char buffer[BUFFER_SIZE];
ssize_t bytes_read, bytes_sent;
while ((bytes_read = fread(buffer, 1, BUFFER_SIZE, file_to_send)) > 0) {
    bytes_sent = write(client_socket, buffer, bytes_read);
    if (bytes_sent == -1) {
        perror("Data sending failed");
        fclose(file_to_send);
        close(client_socket);
        exit(EXIT_FAILURE);
    } else if (bytes_sent < bytes_read) {
        printf("Data sent partially to server\n");
    } else {
        printf("Data sent successfully to server\n");
    }
}
```

Figure 4.Send file in client side