Algorithm for TCP Client-Server File Transfer

Server Side

1. **Create a Socket**: Initialize a TCP socket using socket().
2. **Bind the Socket**: Assign an IP address and port to the socket using bind().
3. **Listen for Connections**: Put the socket in listening mode using listen().
4. **Accept Client Connection**: Wait for a client to connect using accept().
5. **Receive Filename**: Read the requested filename from the client using recv().
6. **Open & Send File**:
    - If the file exists, read its contents and send it using send().
    - If the file is not found, send an error message.
7. **Close Connection**: Close the client socket after transmission.
8. **Repeat**: Keep the server running to accept new client connections.

Client Side

1. **Create a Socket**: Initialize a TCP socket using socket().
2. **Connect to Server**: Use connect() to establish a connection with the server.
3. **Send Filename**: Input the filename from the user and send it to the server using send().
4. **Receive File Data**: Read the file contents (or error message) from the server using recv().
5. **Display Data**: Print the received file contents to the console.
6. **Close Connection**: Close the socket after receiving the data.

Server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

```
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

void send_file_contents(int client_socket, char *filename) {
    FILE *file = fopen(filename, "r");
    char buffer[BUFFER_SIZE];

    if (file == NULL) {
        strcpy(buffer, "ERROR: File not found.\n");
        send(client_socket, buffer, strlen(buffer), 0);
        return;
    }

    while (fgets(buffer, BUFFER_SIZE, file) != NULL) {
        send(client_socket, buffer, strlen(buffer), 0);
    }

    fclose(file);
}

int main() {
    int server_fd, client_socket;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_size;
    char filename[BUFFER_SIZE];

    // Create socket
    server_fd = socket(AF_INET, SOCK_STREAM, 0);
```

```
if (server_fd == -1) {
    perror("Socket creation failed");
    exit(EXIT_FAILURE);
}

// Set up address
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = INADDR_ANY;
server_addr.sin_port = htons(PORT);

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

// Listen for connections
if (listen(server_fd, 5) == -1) {
    perror("Listen failed");
    exit(EXIT_FAILURE);
}

printf("Server listening on port %d...\n", PORT);

while (1) {
    addr_size = sizeof(client_addr);
    client_socket = accept(server_fd, (struct sockaddr *)&client_addr, &addr_size);
    if (client_socket == -1) {
        perror("Accept failed");
        continue;
```

```
        }

        printf("Client connected!\n");

        // Receive filename from client
        memset(filename, 0, BUFFER_SIZE);
        recv(client_socket, filename, BUFFER_SIZE, 0);
        printf("Client requested file: %s\n", filename);

        // Send file contents to client
        send_file_contents(client_socket, filename);

        close(client_socket);
    }

    close(server_fd);
    return 0;
}
```

Client.c

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

#define SERVER_IP "127.0.0.1"
#define PORT 8080
#define BUFFER_SIZE 1024
```

```
int main() {
    int client_socket;
    struct sockaddr_in server_addr;
    char filename[BUFFER_SIZE];
    char buffer[BUFFER_SIZE];

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

    // Configure server address
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);

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

    printf("Connected to server.\n");
    printf("Enter file name to request: ");
    scanf("%s", filename);

    // Send filename to server
```

```
    send(client_socket, filename, strlen(filename), 0);

    // Receive and print file contents
    printf("\nReceived file contents:\n");
    while (recv(client_socket, buffer, BUFFER_SIZE, 0) > 0) {
        printf("%s", buffer);
        memset(buffer, 0, BUFFER_SIZE);
    }

    close(client_socket);
    return 0;
}
```

Output: