File Transfer using TCP Socket in C

**Abstract:** This project implements a robust and efficient file transfer application using TCP sockets in the C programming language. The application consists of a client and a server component, communicating over a reliable TCP connection. The client reads a text file, divides its contents into manageable chunks, and sends these chunks to the server. The server receives the chunks, reassembles them into the original file, and writes the reconstructed file to a new location.

The project leverages the Transmission Control Protocol (TCP), a connection-oriented protocol known for its reliability and error-checking mechanisms. TCP establishes a reliable connection between the client and server through a three-way handshake, ensuring that data is transmitted accurately and without loss. This connection-oriented nature of TCP provides a reliable foundation for file transfer, as it guarantees that all data is delivered in the correct order and without any errors.

The project structure is organized into two main files: client.c and server.c. The client.c file encapsulates the client-side logic, which includes the following steps:

Opening the input file: The client opens the text file to be transferred using standard file I/O functions.

Reading the file in chunks: The client reads the file contents in manageable chunks to avoid overwhelming the network connection. These chunks can be of a fixed size or dynamically determined based on factors such as network bandwidth and file size.

Sending the chunks to the server: The client sends each chunk to the server over the established TCP connection using the send function. The send function ensures reliable delivery of the data by incorporating error detection and recovery mechanisms.

Handling errors: The client handles potential errors that may occur during the transmission process. This includes errors such as network timeouts, connection failures, or data corruption. If an error occurs, the client can attempt to resend the affected data or terminate the transfer gracefully.

The server.c file manages the server-side operations, which involve the following tasks:

Listening for incoming connections: The server creates a TCP socket and binds it to a specific port number. It then listens for incoming connection requests from clients.

Accepting connections: When a client connects to the server, the server accepts the connection and establishes a TCP connection.

Receiving chunks from the client: The server receives the chunks sent by the client using the recv function. The recv function ensures reliable reception of the data by incorporating error detection and recovery mechanisms.

Reassembling the file: The server reassembles the received chunks into the original file by concatenating them in the correct order.

Creating a new output file: The server creates a new file to store the reconstructed data.

Writing the file to disk: The server writes the reconstructed file to the specified location using standard file I/O functions.

To simplify the compilation process, a Make file is provided. By executing make server, the server code is compiled, and running make client compiles the client code. To compile both components, the make all command can be used.

TEAM-5

2320030162

2320030172

2320030299