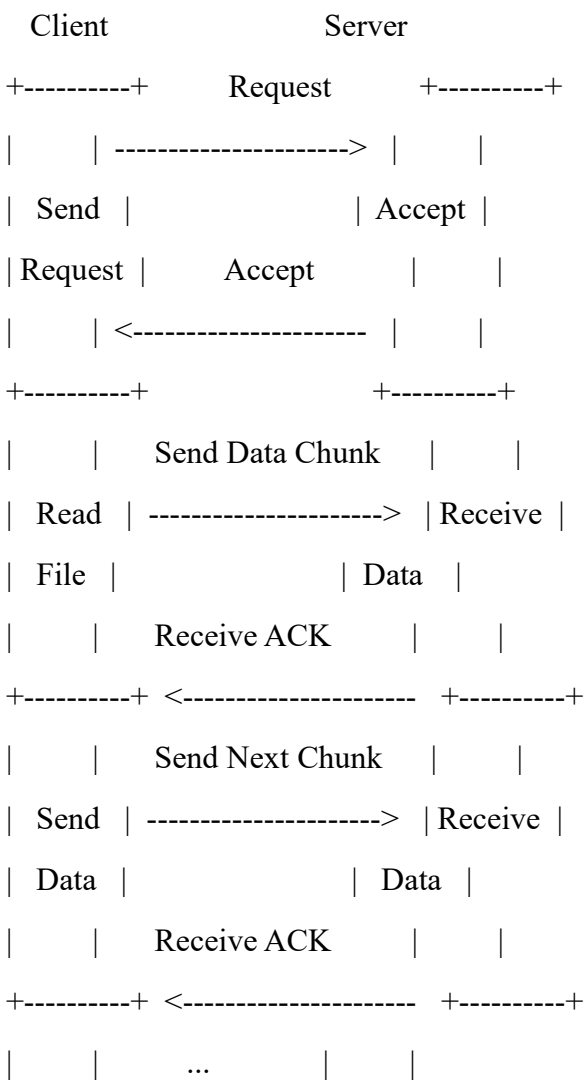
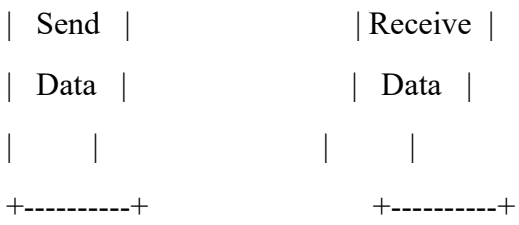
1. The protocol design:

A simple protocol for file transfer over TCP/IP sockets can be designed using a client-server model. The protocol involves steps such as establishing a connection, sending the file from the client to the server, and receiving the file on the server side. Here's a brief outline of the protocol:

- Client sends a request to the server to initiate file transfer.
- Server accepts the connection and waits for data.
- Client reads the file to be sent and divides it into chunks.
- Client sends each chunk of data to the server.
- Server receives the data and writes it to a file.

The figure will look like this:



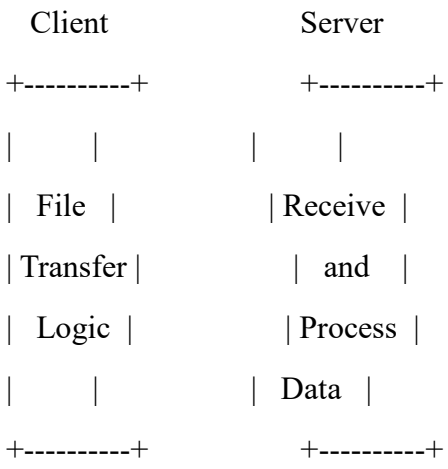


2. The system organization:

The system is organized into two main components: the client and the server:

- The client is responsible for reading the file to be sent and establishing a connection with the server.
- The server is responsible for accepting connections from clients, receiving data, and writing the received data to a file.

The figure of the system organization will look like this:



3. The file transfer implementation:

- The file transfer is implemented using TCP/IP sockets in C programming language.
- The client reads the file to be sent, divides it into chunks, and sends each chunk of data to the server using the **send** function:

```c
// File reading and sending logic
FILE *file_to_send = fopen("file_to_send.txt", "r");
if (file_to_send == NULL) {
    perror("File open error");
    exit(EXIT_FAILURE);
}
```

- The server accepts connections from clients, receives data using the **recv** function, and writes the received data to a file:

```c
// File receiving logic
FILE *received_file = fopen("received_file.txt", "w");
if (received_file == NULL) {
    perror("File open error");
    exit(EXIT_FAILURE);
}
```

4. Roles:

- The client is responsible for initiating the file transfer, reading the file to be sent, and sending data chunks to the server.
- The server is responsible for accepting connections from clients, receiving data chunks from clients, and writing the received data to a file.