File Transfer System Over TCP/IP in CLI

This report analyzes the design, organization, and implementation of the provided Java socket-based file transfer application, which consists of FileServer,ClientConnection, and FileClient classes.

1. Protocol Design

The system implements a simple, request-response application-level protocol built on top of TCP sockets, using the DataOutputStream method for data exchange. This ensures predictable communication between the client and the server.

The protocol steps are as follows:

1. Server Menu Transmission: The server initiates the communication by sending two pieces of information to the client: the total count of files available and the formatted text menu listing all files in the sendfiles directory.
2. Client Selection: The client parses the file count and the menu, prompts the user, and sends the user's 1-indexed selection back to the server.
3. Server File Transfer: The server reads the selected index, locates the corresponding file, and sends the raw file name and the entire file content back to the client .

This design is synchronous and blocks the connection until the file is completely sent and the connection is closed.

2. System Organization

The system is organized using a classic multi-threaded Client-Server Architecture.

The main components and their responsibilities are:

Component	Role	Description
`FileServer`	Entry Point	Initializes a `ServerSocket` on port 3000 and continuously waits for new client connections in a `while(true)` loop.

`ClientConnection`	Server Handler	Runs in a separate thread for each connected client. It manages the menu, handles file selection, and performs the actual file reading and sending.
`FileClient`	Client Application	Connects to the server, handles user input for file selection, and receives the file content for local saving into the `receivedfile` directory.

The key to its organization is concurrency: the `FileServer` immediately delegates each accepted `Socket` to a new `Thread` running a `ClientConnection` instance, allowing the server to handle multiple simultaneous file transfers.

3. File Transfer Implementation

The core file transfer logic resides in the `ClientConnection.sendSelectedFile` method on the server. The entire content of the selected file is read into memory as a single string before being transmitted, which is suitable for smaller files but may be inefficient for very large files.

The following code snippet shows how the server reads the file content and sends both the file name and the content over the socket's output stream:

```
private void sendSelectedFile(int index) throws IOException {

        File[] filelist = new File(FileServer.FILES_PATH).listFiles();

        File selectedFile = filelist[index];

        out.writeUTF(selectedFile.getName());

        List<String> filelines =
Files.readAllLines(selectedFile.toPath());

        String fileContent = String.join("\n",filelines);

        out.writeUTF(fileContent);

    }
```