Practical 1 Report - File Transfer over TCP/IP in CLI

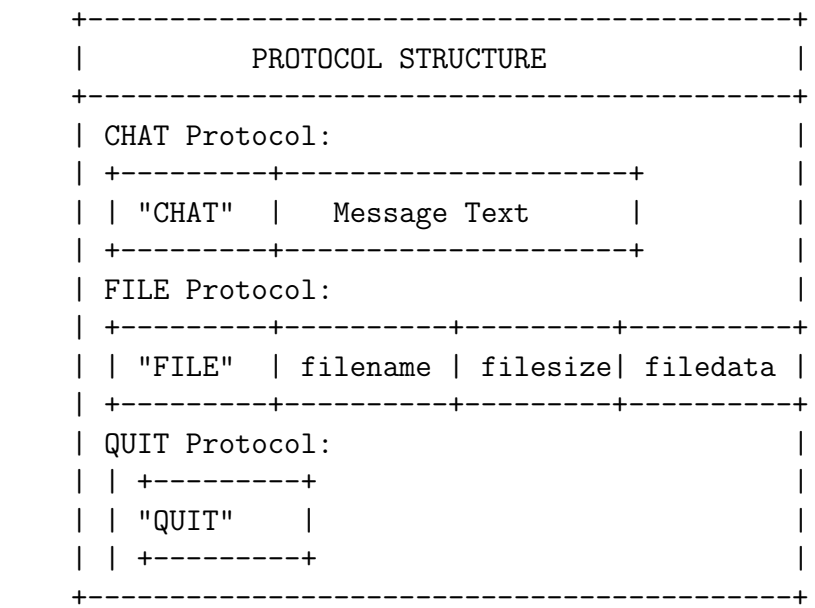
Nguyen Pham Truong An - 23BI14004

20 November, 2025

1 Protocol Design

1.1 Message Protocol Framework

The system uses a framed protocol to separate messages into chatting messages and file transfers, preventing stream corruption.



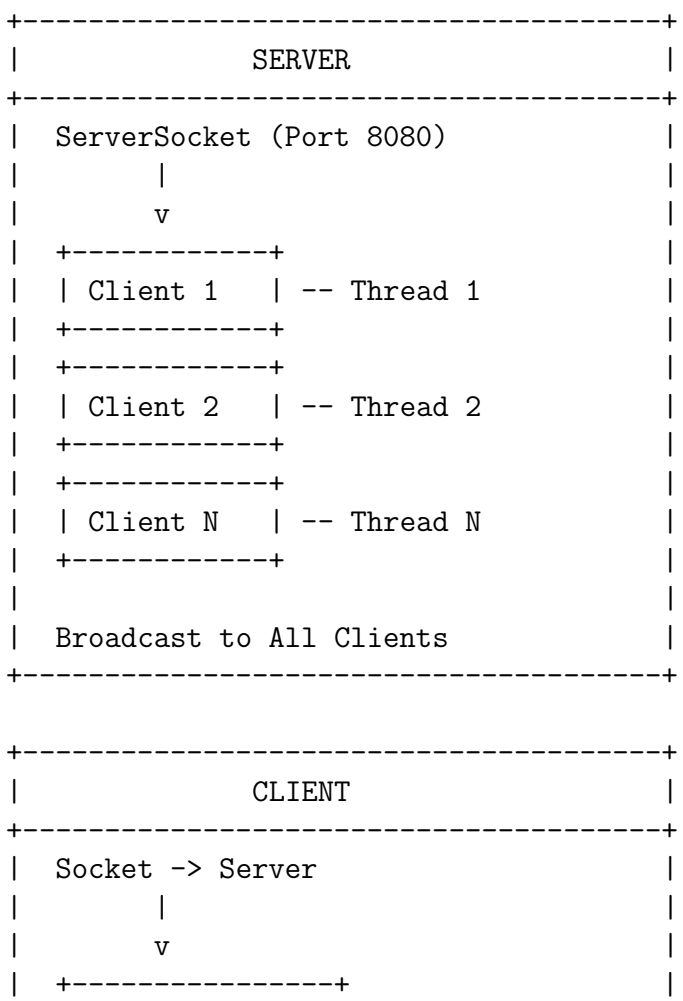
Protocol Types:

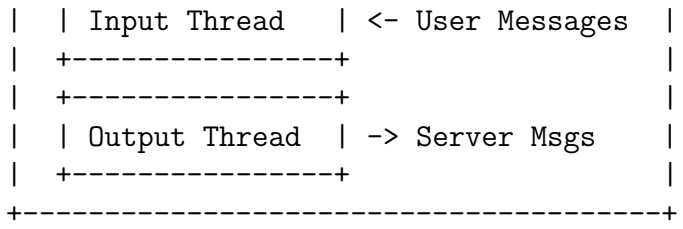
- `CHAT` - Text message: `CHAT + message`
- `FILE` - File transfer: `FILE + filename + filesize + filedata`
- `QUIT` - Client disconnection: `QUIT`

2 System Architecture

2.1 Client-Server Organization

The system follows a multi-threaded client-server model.





Key Components:

- **Server**: Manages client connections, broadcasts messages, handles file storage
- **Client**: Sends messages/files, receives broadcasts, has separate I/O threads
- **Thread Management**: Each client runs in separate thread with synchronized data structures

3 File Transfer Implementation

3.1 Core File Transfer Code

The file transfer uses `DataInputStream` and `DataOutputStream` for proper protocol framing.

Listing 1: File Transfer Implementation

```
// Client-side file sending
private static void sendFile(String filename, DataOutputStream dataOut)
    File file = new File(filename);

    // Send file protocol header
    dataOut.writeUTF("FILE");
    dataOut.writeUTF(file.getName());
    dataOut.writeLong(file.length());
    dataOut.flush();

    // Stream file data in chunks
    FileInputStream fileIn = new FileInputStream(file);
    byte[] buffer = new byte[4096];
```

```
    int bytesRead;

    while ((bytesRead = fileIn.read(buffer)) != -1) {
        dataOut.write(buffer, 0, bytesRead);
    }
    dataOut.flush();
    fileIn.close();
}

// Server-side file receiving
private static void handleFileTransfer(DataInputStream dataIn) {
    // Read file metadata
    String filename = dataIn.readUTF();
    long fileSize = dataIn.readLong();

    // Receive exact file size
    FileOutputStream fileOut = new FileOutputStream(filename);
    byte[] buffer = new byte[4096];
    long bytesReceived = 0;
    int bytesRead;

    while (bytesReceived < fileSize &&
           (bytesRead = dataIn.read(buffer)) != -1) {
        fileOut.write(buffer, 0, bytesRead);
        bytesReceived += bytesRead;
    }
    fileOut.close();
}
```

3.2 Key Features

- **Protocol Framing**: Clear separation between text and binary data
- **Size Tracking**: Prevents over-reading using file size headers
- **Stream Integrity**: Maintains binary data integrity across socket
- **Error Handling**: Proper resource management and exception handling