Ampliación del Sistema de Gestión de Ficheros Remoto Avanzado(TCP)

1. Contexto de la Tarea

Una vez implementado el núcleo del sistema cliente-servidor para la gestión de archivos, el objetivo de esta ampliación es dotar al servidor de capacidades de **auditoría** y añadir **funcionalidades avanzadas** de inspección de archivos para mejorar la experiencia del usuario.

2. Sistema de Registro (Logging)

El servidor ahora debe realizar un seguimiento de la actividad de los clientes. Cada vez que ocurra un evento, se debe mostrar por pantalla (o guardar en un archivo `server.log`) una línea con el formato `[FECHA_HORA] [IP_CLIENTE] [EVENTO]`:

- **Conexiones**: Registrar cuando un nuevo cliente se conecta.
- **Peticiones**: Registrar cada comando recibido del cliente (ej: `RECV: list /home/user->RESP OK`).
- **Desconexiones**: Registrar cuando un cliente finaliza la sesión con el comando `quit` o por cierre inesperado.

3. Nuevos Comandos Soportados

Debes añadir los siguientes cuatro comandos al protocolo de comunicación:

1. **`upload <nombre_archivo>`**: El cliente envía un archivo desde su máquina local al directorio actual del servidor.
    - **Protocolo**:
        1. El cliente envía `upload <nombre>`.
        2. El servidor responde `OK` si tiene permisos para escribir.
        3. El cliente envía el tamaño del archivo (en bytes) y, acto seguido, los bytes del archivo.
    - **Error**: El servidor responde `KO` si el archivo ya existe o no hay espacio/permisos.
2. **`download <ruta_archivo>`**: El servidor envía un archivo al cliente.
    - **Protocolo**:
        1. El cliente envía `download <ruta>`.
        2. El servidor comprueba si el archivo existe.
        3. Si existe, responde `OK` seguido del tamaño en bytes y el flujo de datos.
    - **Error**: Si el archivo no existe o es un directorio, responde `KO`.

4. Requisitos Técnicos Actualizados

- **Mantenimiento de Protocolo**: Todos los comandos deben seguir terminando en salto de línea (`\n`) y mantener la estructura de respuesta de éxito (`OK`) o error (`KO`).
- **Persistencia del Servidor**: El servidor debe seguir siendo multihilo, asegurando que el registro de logs de diferentes hilos no se entrelace de forma ilegible (gestión de concurrencia).

Guía técnica para el cambio de flujo

Hasta ahora, la comunicación era mediante `BufferedReader` y `PrintWriter` (modo texto). Para enviar archivos (imágenes, PDFs, etc.), deben usar:

- **`DataInputStream` / `DataOutputStream`**: Para enviar el tamaño del archivo (long) y luego los bytes.
- **Buffers de transferencia**: No lean el archivo byte a byte. Usen un array de bytes (ej. `byte[] buffer = new byte[4096]`) para que la transferencia sea eficiente.

Advertencia sobre el "Mezclado" de Flujos

Es crucial saber que tras enviar el `OK` (texto), el flujo debe pasar a binario. Debes tener cuidado de no cerrar el `Socket` ni los `Wrappers` de texto prematuramente, o la conexión se cortará antes de terminar la descarga.

Ejemplo de implementación en el Servidor

Para que el servidor sea multihilo y gestione los comandos, la inicialización típica dentro del hilo (`Thread`) del cliente sería:

```
// Obtener el flujo binario del socket
InputStream input = socket.getInputStream();

// Capa 1: Convertir bytes a caracteres
InputStreamReader reader = new InputStreamReader(input);

// Capa 2: Agrupar caracteres en líneas de texto
BufferedReader bufferedReader = new BufferedReader(reader);

// Uso: Leer el comando enviado por el cliente (ej: list, show, upload)
String comando = bufferedReader.readLine(); // Devuelve el comando sin el \n
```

Guía de Implementación: Logging Concurrente

1. Centralización del Log

En lugar de que cada hilo use `System.out.println` de forma desordenada, lo ideal es crear una clase dedicada o un método **sincronizado**. Esto garantiza que solo un hilo escriba a la vez.

2. Formato de Trazabilidad

Para que un log sea útil en red, debe incluir:

- **Timestamp**: Cuándo ocurrió (usando `LocalDateTime`).
- **Identificador**: Quién lo hizo (dirección IP).
- **Acción**: Qué comando se ejecutó y cuál fue el resultado.

3. Ejemplo de salida del login.

[2024-05-20 11:00:05] [192.168.1.15] RECV: download /docs/guia.pdf -> RESP: OK (Sent 15420 bytes)
[2024-05-20 11:02:10] [192.168.1.30] RECV: upload virus.exe -> RESP: KO (File doesn't exists)
[2024-05-20 11:05:45] [192.168.1.30] RECV: upload informe.docx -> RESP: OK (Received 2048 KB)
[2024-05-20 11:10:12] [127.0.0.1]   RECV: download secreto.txt -> RESP: KO (File not found)