
MÔ TẢ BÀI TOÁN

Xây dựng hệ thống **"Quản lý tập tin tập trung"** hoạt động theo mô hình Client-Server sử dụng kỹ thuật **Socket TCP/IP**. Hệ thống mô phỏng giao thức truyền tải Command/Response (tương tự FTP/POP3) cho phép người dùng xác thực và truyền tải tập tin.

YÊU CẦU KỸ THUẬT

1. **Giao thức giao tiếp:** Sử dụng **TCP Socket**
    - Server lắng nghe tại cổng **2024**.
    - Server phải hỗ trợ **Đa luồng (Multithreading)** để phục vụ nhiều Client cùng lúc.

2. **Cơ sở dữ liệu:** Sử dụng **Microsoft Access** để lưu thông tin người dùng (Bảng: USERS).
3. **Lưu trữ:**
    - **Server:** Tự động tạo thư mục gốc E://server khi khởi động.
    - **Client:** Tự động tạo thư mục gốc E://client khi khởi động.
4. **Đóng gói dữ liệu (Pack/Unpack):** Khi truyền tải file, bắt buộc phải gửi kích thước file (Size) trước, sau đó mới gửi nội dung (Data byte) để đảm bảo toàn vẹn dữ liệu.

CHI TIẾT GIAO THỨC (PROTOCOL)

Hệ thống hoạt động dựa trên các dòng lệnh (Text-based commands) kết hợp với truyền luồng byte (Byte-stream) khi tải file.

I. TRẠNG THÁI TRƯỚC KHI ĐĂNG NHẬP (NOT_LOGGED_IN)

Khi Client vừa kết nối đến Server (`Socket s = new Socket(...)`), phiên làm việc ở trạng thái chưa xác thực.

1. **Lệnh: UNAME <username>**
    - **Client gửi:** Chuỗi UNAME kèm tên đăng nhập.
    - **Server xử lý:** Truy vấn CSDL kiểm tra sự tồn tại của user.
    - **Phản hồi:**
        - Thành công: "OK User accepted".
        - Thất bại: "ERR User not found".
2. **Lệnh: PASS <password>**
    - **Điều kiện:** Chỉ chấp nhận sau khi lệnh UNAME đã trả về OK.
    - **Server xử lý:** Dùng **JDBC** kiểm tra mật khẩu
    - **Phản hồi:**
        - Thành công: "OK Login success". **Chuyển trạng thái sang LOGGED_IN**.
        - Thất bại: "ERR Wrong password".
3. **Lệnh: EXIT**

- **Mô tả:** Đóng kết nối Socket và kết thúc luồng xử lý tại Server.

II. TRẠNG THÁI SAU KHI ĐĂNG NHẬP (LOGGED_IN)

Mỗi Client kết nối sẽ được Server cấp phát một luồng riêng (ThreadedEchoHandler). Server cần lưu trữ biến currentDirectory riêng cho từng luồng này.

4. **Lệnh: SS_DIR <folder_path> (Set Server Directory)**
   - **Mô tả:** Yêu cầu Server thay đổi thư mục làm việc hiện hành sang đường dẫn mới.
   - **Xử lý:** Server kiểm tra đường dẫn bằng java.io.File.
   - **Phản hồi:** "OK New path is <path>" hoặc "ERR Path invalid".
5. **Lệnh: SC_DIR <folder_path> (Set Client Directory)**
   - **Mô tả:** Thay đổi thư mục hiện hành tại phía **Client**.
   - **Xử lý:** Client tự kiểm tra và cập nhật biến đường dẫn cục bộ, không gửi lệnh qua mạng.
6. **Lệnh: UPLOAD <filename>**
   - **Mô tả:** Client gửi file lên Server.
   - **Quy trình Đóng gói (Pack) & Gửi:**
     1. Client gửi lệnh: UPLOAD ten_file.
     2. Server kiểm tra quyền ghi -> Phản hồi "READY".
     3. Client gửi kích thước file (long).
     4. Client gửi luồng byte của file.
     5. Server đọc đúng số byte kích thước đã nhận và ghi ra đĩa.
   - **Phản hồi cuối:** "OK Upload completed".
7. **Lệnh: DOWNLOAD <filename>**
   - **Mô tả:** Client tải file từ Server.
   - **Quy trình:**
     1. Client gửi lệnh: DOWNLOAD ten_file.
     2. Server kiểm tra file tồn tại -> Gửi kích thước file (long). (Nếu không có file, gửi kích thước -1 hoặc thông báo lỗi).
     3. Server gửi luồng byte của file.
     4. Client nhận kích thước, sau đó đọc đủ số byte và ghi ra đĩa tại E://client.
8. **Lệnh: EXIT**
   - **Mô tả:** Ngắt kết nối.

HƯỚNG DẪN

1. Về Socket & Đa luồng:

- Server bắt buộc phải dùng vòng lặp while(true) để accept() client.
- Khi có Client kết nối, tạo một Class mới (ví dụ: ClientHandler implements Runnable) và

truyền Socket vừa nhận được vào đó để xử lý.
+1

- Mọi logic UNAME, PASS, UPLOAD... nằm trong hàm run() của ClientHandler.

2. Về Luồng dữ liệu (I/O Streams - Module 1):

- Nên sử dụng DataInputStream và DataOutputStream bọc lấy socket.getInputStream() và socket.getOutputStream().
- Lý do: Các lớp này hỗ trợ phương thức readUTF()/writeUTF() để gửi lệnh text và readLong()/writeLong() để gửi kích thước file, rất tiện cho việc "đóng gói".
- *Lưu ý khi gửi file:* Sử dụng một bộ đệm (buffer) ví dụ byte[] buffer = new byte[4096] để đọc và gửi dần dần, tránh đọc toàn bộ file lớn vào RAM.

3. Về JDBC (Module 4):

- Tương tự bài trước, kết nối Access qua DriverManager.getConnection(url).

- Nên khởi tạo Connection một lần trong Server hoặc trong mỗi Thread (tùy chiến lược), nhưng phải đảm bảo đóng Connection khi Client thoát.

4. Gợi ý xử lý lệnh (Command Parsing):

- Khi nhận chuỗi từ Client (ví dụ: "UPLOAD data.txt"), sử dụng String.split(" ") hoặc StringTokenizer để tách lệnh ("UPLOAD") và tham số ("data.txt").
