**🌐 KẾ HOẠCH HỆ THỐNG DFSS + BITTORRENT (DHT - KADEMLIA)**

**🎯 MỤC TIÊU**

**Xây dựng một mạng chia sẻ và lưu trữ tệp phân tán (DFSS) hoạt động theo cơ chế BitTorrent + DHT Kademlia, có một Web Gateway (Spring Boot) làm cầu nối với người dùng Internet.  
Mục tiêu:**

* **Phi tập trung (no central server).**
* **Có khả năng lưu trữ lâu dài (DFSS).**
* **Hỗ trợ truyền tệp hiệu quả (BitTorrent).**
* **Dễ truy cập và quản lý qua web (Gateway).**

**🏗️ TỔNG QUAN KIẾN TRÚC**

**🌍 Người dùng Internet**

**│**

**▼**

**┌─────────────────────────────────────────────────────────────┐**

**│ 🌐 WEB GATEWAY SERVER (Spring Boot) │**

**│─────────────────────────────────────────────────────────────│**

**│ - Cung cấp giao diện Web UI / REST API │**

**│ - Nhận file tải lên từ người dùng │**

**│ - Tách file → Chunking + Hashing │**

**│ - Giao tiếp với mạng DFSS/BitTorrent qua API nội bộ │**

**│ - Truy vấn DHT để tìm vị trí file │**

**│ - Tải file từ các node & tái hợp │**

**│ - Quản lý metadata (DB hoặc file manifest) │**

**└──────────┬──────────────────────────────────────────────────┘**

**│ HTTP/gRPC nội bộ**

**▼**

**┌───────────────────────────────────────────────────────────────┐**

**│ 🌐 MẠNG DFSS + BITTORRENT │**

**│───────────────────────────────────────────────────────────────│**

**│ Node1 Node2 Node3 Node4 Node5 ... NodeN │**

**│ │ │ │ │ │ │ │**

**│ └─────────UDP RPC + Kademlia DHT Routing────────────────────┘**

**│ ↳ DHT lưu: {chunk\_hash → [node\_addresses]} │**

**│ ↳ Mỗi node lưu 1 phần dữ liệu (chunk) + metadata │**

**│ ↳ Tải/gửi chunk qua TCP (giống BitTorrent) │**

**└───────────────────────────────────────────────────────────────┘**

**⚙️ CÁC THÀNH PHẦN CHÍNH**

**1️⃣ WEB GATEWAY SERVER (Spring Boot + Web UI)**

**Cầu nối giữa người dùng bên ngoài và mạng P2P DFSS-BitTorrent.**

**🔧 Nhiệm vụ chính:**

| **Chức năng** | **Mô tả** |
| --- | --- |
| **Upload file** | **Nhận file → tách thành chunk → tạo hash list** |
| **Chunking** | **Tách file lớn thành nhiều phần nhỏ (256 KB - 1 MB)** |
| **Hashing** | **Sinh SHA-256 cho từng chunk và SHA-256 tổng thể** |
| **Metadata generation** | **Tạo file manifest.json mô tả file: {file\_hash, chunk\_list}** |
| **Giao tiếp DHT** | **Gửi STORE RPC lưu {chunk\_hash → node\_address}** |
| **Download file** | **Dựa trên metadata, truy vấn DHT để tải chunk từ nhiều node** |
| **Tái hợp file** | **Nối lại toàn bộ chunk theo thứ tự** |
| **Hiển thị thông tin file** | **Cho phép user tìm kiếm, xem, tải về, chia sẻ link hash** |

**⚙ Công nghệ:**

* **Spring Boot (REST API, WebSocket)**
* **Thymeleaf / React Web UI**
* **Database metadata (SQLite hoặc PostgreSQL)**
* **gRPC hoặc HTTP client nội bộ để nói chuyện với node cục bộ**

**2️⃣ MẠNG DFSS (Distributed File Storage System)**

**Đây là tầng lưu trữ thực tế, lưu dữ liệu vật lý của file và bản sao của các chunk.**

**🧱 Kiến trúc Node DFSS:**

| **Thành phần** | **Mô tả** |
| --- | --- |
| **Storage Engine** | **Lưu chunk file và metadata tại chỗ (ổ đĩa cục bộ hoặc S3-like backend)** |
| **Replication Manager** | **Đảm bảo mỗi chunk có ít nhất K bản sao trên các node khác nhau** |
| **Health Checker** | **Kiểm tra định kỳ node chết → kích hoạt tái sao lưu** |
| **Data Encryptor** | **Mã hóa dữ liệu khi lưu trữ** |
| **DHT Client** | **Đăng ký, tìm kiếm chunk trên Kademlia** |
| **BitTorrent Module** | **Chia sẻ chunk với các node khác qua TCP** |

**3️⃣ BITTORRENT MODULE (trên mỗi Node DFSS)**

**Dùng để truyền file/chunk giữa các node hiệu quả.**

**🔩 Chức năng:**

| **Tính năng** | **Mô tả** |
| --- | --- |
| **Piece Exchange Protocol** | **Giống BitTorrent, gửi/nhận các phần (piece) của chunk** |
| **Peer Discovery** | **Hỏi DHT để biết node nào có chunk cần tải** |
| **Parallel Downloading** | **Tải nhiều chunk cùng lúc từ các node khác nhau** |
| **Upload service** | **Cho phép node khác tải chunk từ mình** |
| **Integrity Check** | **Dùng SHA-256 để kiểm tra từng chunk sau khi tải** |

**4️⃣ KADEMLIA DHT (P2P Network Layer)**

**Mạng định tuyến phân tán, đóng vai trò lookup table toàn cục.**

**🔑 Key-value lưu trữ:**

**Key = hash(chunk)**

**Value = [list of node\_addresses chứa chunk đó]**

**🔁 RPC cơ bản:**

* **PING(node\_id) → Kiểm tra node còn sống**
* **STORE(key, value) → Lưu vị trí chunk**
* **FIND\_NODE(target\_id) → Tìm node gần nhất theo XOR**
* **FIND\_VALUE(key) → Truy xuất danh sách node có chunk**

**⚙ Cấu trúc:**

* **Node ID: 160-bit SHA-1**
* **Routing table: K-buckets (k = 20 thường)**
* **Khoảng cách XOR metric để tìm node gần nhất**
* **Lookup phức tạp O(log N)**

**🔗 LUỒNG HOẠT ĐỘNG**

**🔹 Upload file (người dùng → Gateway → DFSS)**

1. **Người dùng tải file lên Gateway.**
2. **Gateway chia file thành nhiều chunk.**
3. **Gateway gửi từng chunk đến các Node DFSS ngẫu nhiên để lưu.**
4. **Mỗi node lưu chunk → đăng ký {chunk\_hash → node\_ip} vào DHT.**
5. **Gateway tạo file manifest.json (chứa danh sách hash + metadata).**
6. **Gateway trả về file\_hash (định danh duy nhất của file).**

**🔹 Download file**

1. **Người dùng gửi yêu cầu tải file (file\_hash) tới Gateway.**
2. **Gateway tải manifest.json (metadata).**
3. **Với mỗi chunk\_hash, Gateway hỏi DHT → nhận danh sách node chứa chunk.**
4. **Gateway (hoặc client) tải đồng thời các chunk từ nhiều node.**
5. **Khi đủ tất cả chunk → ghép thành file gốc → trả cho người dùng.**

**🔹 File replication (nội bộ DFSS)**

1. **DFSS Node định kỳ kiểm tra DHT để xem số lượng bản sao cho mỗi chunk.**
2. **Nếu số lượng < K → node khác tự động tải chunk và lưu bản sao.**
3. **Cập nhật lại DHT với node mới.**

**🔐 ƯU ĐIỂM KIẾN TRÚC**

**✅ Hoàn toàn phi tập trung — không cần tracker hay server trung tâm.  
✅ Kết hợp DHT + BitTorrent → vừa tìm kiếm nhanh, vừa truyền tải hiệu quả.  
✅ Có lớp DFSS đảm bảo dữ liệu không mất khi node rời mạng.  
✅ Web Gateway giúp người dùng Internet dễ sử dụng (giao diện thân thiện).  
✅ Mở rộng dễ dàng: thêm node hoặc gateway mới mà không cần cấu hình lại mạng.**

**🔮 HƯỚNG PHÁT TRIỂN TƯƠNG LAI**

| **Mục tiêu** | **Hướng triển khai** |
| --- | --- |
| **Bảo mật** | **Mã hóa end-to-end, chữ ký số cho file manifest** |
| **Tìm kiếm thông minh** | **Dùng metadata index (ElasticSearch / Redis)** |
| **Cân bằng tải** | **Triển khai nhiều Gateway + Load Balancer** |
| **Chia sẻ công khai** | **Cho phép public file qua link hash** |
| **Web3 integration** | **Ghi metadata lên blockchain để chống giả mạo** |

**📘 TÓM TẮT TẦNG KIẾN TRÚC**

| **Tầng** | **Công nghệ** | **Chức năng** |
| --- | --- | --- |
| **Application Layer** | **Spring Boot (Gateway UI/API)** | **Tải lên, tải xuống, giao diện người dùng** |
| **P2P Transport Layer** | **TCP/UDP (BitTorrent Protocol)** | **Truyền tải chunk song song** |
| **DHT Routing Layer** | **Kademlia (UDP RPC)** | **Lookup và lưu trữ định vị chunk** |
| **Storage Layer (DFSS)** | **Local/S3 Storage + Replication** | **Lưu dữ liệu phân tán** |
| **Security Layer** | **AES, RSA, Hashing** | **Mã hóa và kiểm chứng tính toàn vẹn** |

**Nếu bạn muốn, tôi có thể vẽ sơ đồ hệ thống DFSS + BitTorrent + Gateway bằng sơ đồ đồ họa (dạng SVG đẹp như kiến trúc IPFS) để dễ trình bày trong báo cáo hoặc luận văn.**

**👉 Bạn muốn tôi vẽ dạng kỹ thuật (chi tiết tầng) hay dạng kiến trúc hệ thống tổng quan (dễ nhìn, có biểu tượng)?**

FileSharingSystem/

├── pom.xml

├── README.md

├── docker-compose.yml

│

├── gateway/ ← 🌐 Web Gateway (Spring Boot)

│ ├── pom.xml

│ ├── src/main/java/com/dfss/gateway/

│ │ ├── controller/

│ │ │ ├── FileController.java ← Upload, search, download file

│ │ │ ├── TorrentController.java ← Giao tiếp tracker / swarm

│ │ │ └── NodeController.java ← Danh sách node trong mạng

│ │ ├── service/

│ │ │ ├── GatewayService.java ← Điều phối yêu cầu đến node

│ │ │ ├── NodeDiscoveryService.java ← Phát hiện node qua DHT

│ │ │ ├── TorrentManager.java ← Sinh torrent file, seed file

│ │ │ └── DHTClient.java ← Kết nối với DHT (UDP RPC)

│ │ ├── model/

│ │ │ ├── FileInfo.java

│ │ │ ├── NodeInfo.java

│ │ │ └── TorrentInfo.java

│ │ ├── util/

│ │ │ └── HashUtil.java ← SHA-1 / SHA-256 cho infohash

│ │ └── GatewayApplication.java

│ └── src/main/resources/

│ ├── application.properties

│ ├── templates/

│ │ ├── index.html

│ │ ├── upload.html

│ │ ├── download.html

│ │ └── nodes.html

│ └── static/

│ ├── js/

│ │ ├── cli.js

│ │ ├── websocket.js

│ │ └── torrent-ui.js

│ └── css/style.css

│

├── node/ ← ⚙️ Node P2P (Kademlia + BitTorrent)

│ ├── pom.xml

│ ├── src/main/java/com/dfss/node/

│ │ ├── core/

│ │ │ ├── Node.java ← Đại diện 1 peer trong mạng

│ │ │ ├── NodeConfig.java ← Cấu hình node (port, id, path,…)

│ │ │ ├── NodeApplication.java ← Entry chính của node

│ │ │ └── NodeBootstrap.java ← Tham gia mạng DHT

│ │ │

│ │ ├── dht/ ← 🧭 Kademlia DHT Implementation

│ │ │ ├── KademliaNode.java

│ │ │ ├── RoutingTable.java

│ │ │ ├── KBucket.java

│ │ │ ├── KademliaRPC.java ← Gửi/nhận UDP RPC

│ │ │ ├── KademliaServer.java ← Lắng nghe gói tin DHT

│ │ │ ├── Message.java ← PING, STORE, FIND\_NODE, FIND\_VALUE

│ │ │ └── DHTUtils.java

│ │ │

│ │ ├── bittorrent/ ← 🌐 BitTorrent Protocol Layer

│ │ │ ├── TorrentManager.java ← Quản lý .torrent file, peer list

│ │ │ ├── PieceManager.java ← Quản lý các mảnh file (piece)

│ │ │ ├── PeerConnection.java ← Giao tiếp giữa các peer TCP/UDP

│ │ │ ├── TrackerClient.java ← Gửi announce đến tracker (nếu có)

│ │ │ ├── TorrentFile.java ← Metadata của file torrent

│ │ │ └── BitField.java ← Trạng thái mảnh (bitmap)

│ │ │

│ │ ├── file/ ← 💾 Quản lý tệp và chunk

│ │ │ ├── FileManager.java ← Chia, băm, ghép file

│ │ │ ├── FileChunk.java ← Metadata từng phần

│ │ │ └── ChunkStorage.java ← Lưu trữ thực tế trên đĩa

│ │ │

│ │ ├── network/ ← 📡 UDP / TCP giao tiếp

│ │ │ ├── RUDPServer.java

│ │ │ ├── RUDPClient.java

│ │ │ ├── RUDPMessage.java

│ │ │ ├── PacketUtils.java

│ │ │ └── MessageQueue.java

│ │ │

│ │ ├── replication/ ← 🔁 Sao lưu & khôi phục

│ │ │ ├── ReplicationManager.java

│ │ │ └── HeartbeatService.java

│ │ │

│ │ ├── util/

│ │ │ ├── HashUtil.java

│ │ │ ├── ConfigUtil.java

│ │ │ └── LoggerUtil.java

│ │ │

│ │ └── api/

│ │ ├── NodeAPI.java ← REST API cho node (nếu cần)

│ │ └── NodeStatusController.java

│ │

│ └── src/main/resources/

│ ├── application.properties

│ └── log4j2.xml

│

├── tracker/ ← 🧭 Optional Tracker (cho BitTorrent)

│ ├── pom.xml

│ ├── src/main/java/com/dfss/tracker/

│ │ ├── TrackerServer.java ← Lắng nghe announce

│ │ ├── PeerRegistry.java ← Quản lý danh sách peer theo infoHash

│ │ ├── TrackerController.java ← REST API cho Gateway

│ │ └── TrackerApplication.java

│ └── src/main/resources/

│ ├── application.properties

│ └── log4j2.xml

│

└── data/

├── node1/

│ ├── chunks/

│ ├── torrents/

│ └── cache/

├── node2/

│ ├── chunks/

│ ├── torrents/

│ └── cache/

├── node3/

│ ├── chunks/

│ ├── torrents/

│ └── cache/

└── shared/

└── upload/ ← File người dùng tải lên từ Gateway