

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**  
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

-----□□□□-----



**BÁO CÁO**

**MÔN HỌC: CÁC HỆ THỐNG PHÂN TÁN**

**HỆ THỐNG CHAT NGANG HÀNG (P2P)**

<b>Giảng viên</b>	<b>: TS. Nguyễn Duy Phương</b>
<b>Học viên</b>	<b>: Lê Đăng Khoa - B25CHHT031</b> <b>Đỗ Thị Ngọc Mai - B25CHHT039</b> <b>Đặng Hoàng Đức - B25CHHT013</b>
<b>Lớp</b>	<b>: M25CQHT01-B</b>

**HÀ NỘI – NĂM 2025**

# MỤC LỤC

<b>I. Giới thiệu đề tài</b>	<b>1</b>
1. Lý do chọn đề tài	1
2. Mục tiêu đề tài	1
<b>II. Công nghệ sử dụng</b>	<b>1</b>
<b>III. Thiết kế hệ thống</b>	<b>1</b>
1. Phân tích yêu cầu	1
1.5. Phân tích yêu cầu phi chức năng (Non-functional Requirements)	2
1.6. Mô hình luồng dữ liệu (Data Flow)	2
2. Thiết kế phần mềm	3
1. Kiến trúc Hệ thống (System Architecture)	3
2. Thuật toán Phân tán (Distributed Algorithms)	4
3. Cơ chế Bảo mật (Security Mechanisms)	4
4. Xử lý đồng thời & Chịu lỗi (Concurrency & Fault Tolerance)	4

## I. Giới thiệu đề tài

### 1. Lý do chọn đề tài

Trong bối cảnh nhu cầu trao đổi thông tin ngày càng cao, các ứng dụng chat đóng vai trò quan trọng trong đời sống và công việc. Mô hình chat ngang hàng (P2P) cho phép các thiết bị giao tiếp trực tiếp với nhau mà không cần máy chủ trung tâm, giúp giảm chi phí hạ tầng, tăng tính riêng tư và giảm độ trễ.

Đề tài này đặt mục tiêu là tạo ra 1 ứng dụng cho một team nhỏ (< 20 người) có một ứng dụng để nói chuyện riêng tư với nhau trong mạng nội bộ của công ty

### 2. Mục tiêu đề tài

- Tìm hiểu mô hình mạng P2P
- Xây dựng ứng dụng chat cho phép các client giao tiếp trực tiếp
- Đảm bảo gửi/nhận tin nhắn theo thời gian thực
- Hiểu rõ cơ chế kết nối mạng, socket và truyền dữ liệu

## II. Công nghệ sử dụng

- Ngôn ngữ: Python 3.10 (phù hợp với môi trường Conda).
- Giao thức truyền tải:
  - TCP: Giao tiếp P2P tin cậy.
  - UDP Broadcast: Cơ chế tự động tìm kiếm Discovery Server trong mạng LAN.
  - HTTP/WebSocket: Giao tiếp nội bộ giữa Backend (Python) và Frontend (Web UI).
- Framework: Flask (Python) để xây dựng Web UI cho từng nút mạng.
- Deployment: Docker & Docker Compose giúp đóng gói môi trường nhất quán trên nhiều máy tính.

## III. Thiết kế hệ thống

### 1. Phân tích yêu cầu

#### 1.1. Giao tiếp Node-to-Node (Cốt lõi)

- Chat trực tiếp: Thiết lập kết nối trực tiếp (TCP/UDP) giữa hai node khi cả hai cùng online.
- Cơ chế hàng đợi tin nhắn (Store-and-Forward): Nếu người nhận offline, tin nhắn sẽ được lưu tạm tại máy người gửi (hoặc một node trung gian được chỉ định) và tự động đẩy đi khi người nhận online trở lại.
- Lịch sử trò chuyện: Dữ liệu chat được lưu trữ cục bộ (Local Storage/SQLite) tại mỗi máy của người dùng, đảm bảo chỉ người trong cuộc mới có quyền truy cập.

## 1.2. Quản lý Mạng và Định danh

- Server Danh bạ (Directory Server): Đóng vai trò là "Tracker". Server này chỉ lưu trữ thông tin: ID người dùng, Tên, địa chỉ IP hiện tại và trạng thái (Online/Offline). Tuyệt đối không lưu nội dung tin nhắn.
- Cơ chế phát hiện (Discovery): Các node tự động đăng ký IP với Server khi khởi động và cập nhật danh sách bạn bè đang hoạt động.

## 1.3. Quản lý tin nhắn nhóm và Đồng bộ

- Thứ tự tin nhắn (Message Ordering): Sử dụng thuật toán đếm thời gian (như Lamport Timestamp hoặc Vector Clock) để đảm bảo thứ tự tin nhắn nhất quán trên tất cả các máy, tránh hiện tượng tin nhắn đến sau lại hiển thị trước.
- Đồng bộ hóa: Khả năng hợp nhất lịch sử chat từ nhiều nguồn khi một node vừa quay trở lại mạng sau một thời gian offline.

## 1.4. Bảo mật

- Mã hóa đầu cuối (End-to-End Encryption): Tin nhắn được mã hóa tại máy gửi và chỉ được giải mã tại máy nhận bằng khóa riêng biệt.

## 1.5. Phân tích yêu cầu phi chức năng (Non-functional Requirements)

Tiêu chí	Yêu cầu chi tiết
Tính sẵn sàng	Hệ thống vẫn hoạt động (chat 1-1) ngay cả khi Server danh bạ gặp sự cố (nếu đã biết IP của nhau).
Hiệu năng	Độ trễ tin nhắn trong mạng nội bộ < 100ms. Hỗ trợ gửi file dung lượng nhỏ.
Khả năng mở rộng	Tối ưu hóa cho nhóm < 20 người, đảm bảo không bị nghẽn băng thông do cơ chế flooding/broadcast.
Tính tin cậy	Đảm bảo tin nhắn không bị mất (Lossless) thông qua cơ chế ACK (Acknowledge) trong giao thức TCP.

## 1.6. Mô hình luồng dữ liệu (Data Flow)

- Giai đoạn kết nối: Node A gửi yêu cầu lên Server để lấy IP của Node B.
- Giai đoạn gửi tin:
  - Nếu Node B Online: A thiết lập kết nối socket tới B và gửi tin.
  - Nếu Node B Offline: Tin nhắn vào hàng đợi tại Local DB của A, gắn cờ "Pending".

- Giai đoạn nhận tin: Node B online -> Server thông báo cho A -> A tự động đẩy tin nhắn "Pending" cho B.

### 1.7. Liệt kê các tính năng

Node - Node: Có thể chat trực tiếp, lưu trữ lịch sử chat, lưu dữ liệu chat chờ để gửi khi online, bảo mật thông tin, có thể tìm được nhau:

- Có thể chat trực tiếp
- Lưu trữ lịch sử chat
- Lưu dữ liệu chat chờ để gửi khi online
- Bảo mật thông tin
- Có thể tìm được nhau

### Các chức năng chính :

- UPDATE: Cập nhật danh sách người dùng mới nhất từ Server hoặc qua mạng LAN.
- PEERS: Xem danh sách những người đang online.
- CHAT \[username\] \[nội dung\]: Gửi tin nhắn mã hóa.
- HISTORY: Xem lịch sử trò chuyện (chỉ hiển thị các tin nhắn bạn có quyền đọc).
- EXIT: Thoát ứng dụng và lưu dữ liệu.

## 2. Thiết kế phần mềm

Hệ thống cho phép các máy trong cùng mạng LAN chat với nhau theo hướng P2P, trong đó mỗi peer vừa gửi vừa nhận tin. Thiết kế hiện tại tập trung vào:

- Discovery peer trong LAN bằng UDP Broadcast.
- Truyền tin P2P bằng TCP + JSON payload.
- Đồng bộ dữ liệu theo *eventual consistency* bằng Gossip + Vector Clock.
- Bảo mật: Mã hóa nội dung khi truyền bằng RSA (public key của peer đích).

Hệ thống có kèm Discovery Server (tùy chọn) để đăng ký danh bạ peer.

### 2.1. Kiến trúc Hệ thống (System Architecture)

- Mô hình Hybrid P2P: Kết hợp giữa Server trung tâm làm nhiệm vụ điều phối (Discovery Server) và các nút mạng giao tiếp trực tiếp (Peer-to-Peer).
- Discovery Service: Server chỉ đóng vai trò là "danh bạ" lưu trữ IP/Port và Public Key, không lưu trữ hay luân chuyển nội dung tin nhắn.
- Direct Communication (TCP Sockets): Tin nhắn được truyền trực tiếp giữa các Peer sau khi đã lấy được địa chỉ từ Server, đảm bảo tính phi tập trung.

- Local Storage (Shared-nothing): Mỗi node tự quản lý dữ liệu riêng, đảm bảo dữ liệu vẫn tồn tại cục bộ ngay cả khi Server bị sập.

## 2.2. Thuật toán Phân tán (Distributed Algorithms)

- Vector Clock (Đồng hồ Vector):
  - Sử dụng để xác định Thứ tự nhân quả (Causal Ordering) giữa các tin nhắn.
  - Giúp hệ thống sắp xếp lịch sử chat chính xác mà không cần đồng bộ thời gian thực (Physical Clock) giữa các máy.
- Eventual Consistency (Nhất quán cuối cùng): Đảm bảo sau khi tin nhắn được lan truyền, tất cả các node tham gia sẽ có nội dung lịch sử giống nhau.
- Gossip-like Propagation: Cơ chế cập nhật danh sách Peer liên tục từ Server giúp các node luôn biết trạng thái Online/Offline của nhau.

## 2.3. Cơ chế Bảo mật (Security Mechanisms)

- Mã hóa bất đối xứng (RSA-2048):
  - Dùng để mã hóa đầu cuối (E2EE) tin nhắn.
  - Public Key dùng để mã hóa khi gửi, Private Key dùng để giải mã khi nhận.
- Mã hóa đối xứng (AES-256):
  - Sử dụng thư viện Fernet để mã hóa file lưu trữ cục bộ ([.json](#)).
  - Bảo vệ dữ liệu "ngủ" (Data-at-rest), yêu cầu Passphrase chính xác mới có thể truy cập lịch sử chat.
- PBKDF2 (Password-Based Key Derivation Function 2): Thuật toán băm (hashing) để tạo ra khóa AES từ Passphrase của người dùng, chống tấn công dò mật khẩu.

## 2.4. Xử lý đồng thời & Chịu lỗi (Concurrency & Fault Tolerance)

- Multi-threading: Mỗi kết nối P2P đến được xử lý trên một luồng riêng, không chặn (Non-blocking) luồng chính của giao diện.
- No Single Point of Failure (SPOF): Nếu Server Discovery sập, các Peer đã biết IP của nhau vẫn có thể tiếp tục chat trực tiếp và truy cập dữ liệu cũ lưu tại local.
- Mutex/Lock Control: Sử dụng threading.Lock để bảo vệ tài nguyên dùng chung (Ledger, Peer List), tránh lỗi Race Condition khi nhiều tin nhắn đến cùng lúc.