**BÀI TẬP TRÊN LỚP MÔN**

**HỌC: HỆ PHÂN TÁN**

**CHƯƠNG 1: TỔNG QUAN VÀ KIẾN TRÚC HỆ PHÂN TÁN**

HỌ TÊN SV: Nguyễn Thanh Hà MSSV: 20210298

MÃ LỚP: 154056 MÃ HỌC PHẦN: IT4611

**Câu hỏi 1:** Em hãy nêu thêm 2 ví dụ về dịch vụ được coi là Hệ Phân Tán (ngoài 2 ví dụ WWW và Email đã trình bày trên lớp). Dựa vào định nghĩa, giải thích tại sao chúng được coi là Hệ Phân Tán.

**1. Hệ thống quản lý cơ sở dữ liệu phân tán (Distributed Database System)**

- **Ví dụ**: MongoDB, Cassandra
- **Giải thích**:
  - Hệ thống quản lý cơ sở dữ liệu phân tán lưu trữ và quản lý dữ liệu trên nhiều máy chủ nằm ở các địa điểm khác nhau. Dữ liệu được phân phối và sao chép trên các node để đảm bảo tính sẵn sàng và khả năng chịu lỗi.
  - Dữ liệu từ các node khác nhau có thể được truy cập đồng thời từ nhiều người dùng, mà không cần quan tâm đến việc dữ liệu thực sự nằm ở đâu.
  - Đây là hệ phân tán vì hệ thống này bao gồm nhiều máy tính hoạt động độc lập, nhưng người dùng vẫn cảm thấy như đang làm việc với một cơ sở dữ liệu duy nhất. Hệ thống cung cấp khả năng phân tán dữ liệu, đảm bảo tính toàn vẹn và hiệu suất cao trong môi trường có nhiều máy.

**2. Hệ thống lưu trữ tệp phân tán (Distributed File System)**

- **Ví dụ**: Google File System (GFS), Hadoop Distributed File System (HDFS), Amazon S3.
- **Giải thích**:
  - Hệ thống lưu trữ tệp phân tán chia nhỏ và lưu trữ các tệp trên nhiều máy tính khác nhau. Khi người dùng hoặc ứng dụng yêu cầu truy cập một tệp, hệ thống sẽ tự động thu thập dữ liệu từ các máy khác nhau và cung cấp tệp đầy đủ.
  - Các hệ thống này được thiết kế để cung cấp khả năng lưu trữ lớn và sẵn sàng cao, ngay cả khi một hoặc nhiều node gặp sự cố.
  - Đây là hệ phân tán vì chúng hoạt động trên nhiều máy tính và người dùng có thể truy cập vào tệp mà không cần biết tệp thực sự đang được lưu ở đâu. Hệ thống cũng cung cấp tính chịu lỗi và khả năng mở rộng, một đặc điểm nổi bật của các hệ phân tán.

**Câu hỏi 2:** Tại sao nói tính chia sẻ tài nguyên của Hệ Phân Tán có khả năng: Giảm chi phí, tăng tính sẵn sàng và hỗ trợ làm việc nhóm? Tuy nhiên lại tăng rủi ro về an toàn thông tin? Giải thích.

### 1. Giảm chi phí

- Khi nhiều người dùng có thể chia sẻ tài nguyên (như máy chủ, phần cứng, hoặc ứng dụng), tổng chi phí đầu tư vào tài nguyên sẽ được chia sẻ giữa các thành viên hoặc các ứng dụng. Thay vì phải mua và duy trì nhiều hệ thống độc lập, hệ phân tán cho phép tận dụng tốt hơn nguồn lực sẵn có và tối ưu hóa việc sử dụng tài nguyên, từ đó tiết kiệm chi phí.

### 2. Tăng tính sẵn sàng

- Hệ phân tán thường có các bản sao của tài nguyên và dữ liệu trên nhiều máy chủ và địa điểm khác nhau. Nếu một phần của hệ thống gặp sự cố (chẳng hạn như máy chủ bị hỏng), các phần khác vẫn có thể tiếp tục hoạt động, giúp đảm bảo dịch vụ vẫn khả dụng cho người dùng. Điều này giúp hệ thống có tính sẵn sàng cao hơn so với hệ thống tập trung.

### 3. Hỗ trợ làm việc nhóm

- Hệ phân tán cho phép nhiều người dùng từ các địa điểm khác nhau truy cập và làm việc trên cùng một tập tài nguyên một cách đồng thời. Điều này giúp tối ưu hóa quá trình cộng tác trong các nhóm làm việc từ xa hoặc giữa các đơn vị khác nhau trong tổ chức. Ví dụ, các hệ thống lưu trữ đám mây như Google Drive hoặc Dropbox giúp nhiều người cùng làm việc trên một tệp dữ liệu, đồng thời hỗ trợ kiểm soát phiên bản.

### 4. Tăng rủi ro về an toàn thông tin

- Khi tài nguyên được chia sẻ và phân tán trên nhiều địa điểm hoặc nhiều máy chủ, hệ thống trở nên phức tạp hơn và có nhiều điểm đầu mối dễ bị tấn công. Những nguy cơ như tấn công mạng, mất quyền kiểm soát tài nguyên, lỗ hổng bảo mật có thể xảy ra khi hệ thống không được quản lý chặt chẽ. Việc nhiều người dùng truy cập vào cùng một tài nguyên cũng làm tăng nguy cơ về việc dữ liệu nhạy cảm bị truy cập trái phép.

**Câu hỏi 3:** Liên quan đến *tính trong suốt*, giải thích tại sao nhà quản trị hệ thống phải xem xét việc cân bằng giữa hiệu năng và độ trong suốt? Đưa ra ví dụ cụ thể để giải thích.

## Cân bằng giữa hiệu năng và độ trong suốt

- Độ trong suốt càng cao thì hệ thống càng phải xử lý thêm nhiều tác vụ để che giấu những chi tiết phức tạp. Điều này có thể làm giảm hiệu năng, vì hệ thống cần thời gian và tài nguyên để duy trì độ trong suốt.
- Ngược lại, nếu tối ưu hóa cho hiệu năng, đôi khi phải giảm bớt tính trong suốt, ví dụ như để người dùng biết rõ nơi tài nguyên được lưu trữ hoặc lỗi xảy ra để có thể phản hồi kịp thời.

## Ví dụ:

- **Hệ thống lưu trữ đám mây**:
    - Trong hệ thống lưu trữ đám mây (như Google Drive hay Dropbox), để đạt được độ trong suốt, người dùng không cần biết dữ liệu của họ thực sự được lưu trữ ở đâu (trong suốt về vị trí). Điều này giúp người dùng cảm thấy thuận tiện vì chỉ cần truy cập vào một địa chỉ duy nhất.
    - Tuy nhiên, nếu hệ thống phải liên tục duy trì việc đồng bộ dữ liệu giữa nhiều máy chủ phân tán trên toàn cầu, hiệu năng có thể bị giảm, đặc biệt khi người dùng từ một địa điểm xa yêu cầu truy cập vào dữ liệu lưu trữ ở một khu vực khác.
    - Để cải thiện hiệu năng, một số hệ thống có thể giảm tính trong suốt bằng cách cho người dùng biết khu vực lưu trữ và đề xuất chọn máy chủ gần với địa điểm của họ hơn. Điều này giảm độ phức tạp trong việc đồng bộ, nhưng giảm bớt tính trong suốt về vị trí.

**Câu hỏi 5:** So sánh 2 kiểu HĐH DOS và NOS. Giải thích tại sao việc sử dụng Middleware là sự kết hợp ưu điểm của cả 2 mô hình trên.

### 1. Hệ điều hành phân tán (DOS)

- **Đặc điểm**:
    - Hoạt động như một hệ điều hành duy nhất trên một nhóm các máy tính kết nối với nhau qua mạng.
    - Người dùng cảm nhận như họ chỉ đang làm việc với một máy tính duy nhất mặc dù các tài nguyên được phân tán trên nhiều máy.
    - Có tính trong suốt cao về truy cập, vị trí, và sự sao chép.
- **Ưu điểm**:
    - **Tính trong suốt**: Người dùng không cần biết tài nguyên được lưu trữ ở đâu hay dữ liệu đến từ đâu.
    - **Quản lý tập trung**: Việc quản lý tài nguyên, tiến trình và bảo mật được thực hiện thống nhất cho toàn hệ thống.
- **Nhược điểm**:
    - **Hiệu năng**: Vì mọi thứ đều phải được đồng bộ và quản lý tập trung, điều này có thể dẫn đến giảm hiệu năng khi hệ thống lớn.
    - **Phức tạp**: Việc xây dựng và duy trì một hệ điều hành phân tán phức tạp đòi hỏi nhiều nguồn lực và kỹ thuật cao.

### 2. Hệ điều hành mạng (NOS)

- **Đặc điểm**:
    - Mỗi máy trong hệ thống mạng chạy một hệ điều hành riêng và giao tiếp với nhau thông qua mạng.
    - Người dùng phải biết rõ tài nguyên nào nằm ở đâu và phải truy cập tài nguyên thông qua các giao thức mạng.
- **Ưu điểm**:
    - **Hiệu năng tốt hơn**: Vì mỗi máy hoạt động độc lập với tài nguyên riêng, không phải đồng bộ hóa hoặc quản lý tài nguyên chung.
    - **Dễ triển khai**: Không cần một hệ thống tập trung hoặc phức tạp, mỗi máy chỉ cần chạy hệ điều hành của riêng nó.
- **Nhược điểm**:

- **Thiếu tính trong suốt**: Người dùng phải hiểu rõ cấu trúc của hệ thống và truy cập từng tài nguyên một cách thủ công.
- **Quản lý phân tán**: Mỗi máy có thể có cách quản lý tài nguyên và bảo mật riêng, dẫn đến khó khăn trong việc quản lý chung toàn bộ hệ thống.

## 3. Middleware: Kết hợp ưu điểm của DOS và NOS

- **Middleware** là một tầng trung gian giữa các hệ điều hành và các ứng dụng phân tán, cho phép các thành phần hệ thống tương tác với nhau một cách trơn tru hơn.
- **Middleware kết hợp ưu điểm của cả DOS và NOS**:
  - **Tính trong suốt như DOS**: Middleware che giấu sự phức tạp của các hệ thống khác nhau, cung cấp các giao diện thống nhất và trong suốt cho người dùng. Người dùng không cần biết các tài nguyên thực sự ở đâu hoặc cách thức quản lý.
  - **Hiệu năng và tính linh hoạt như NOS**: Mỗi máy vẫn có thể chạy hệ điều hành riêng của mình và quản lý tài nguyên cục bộ một cách độc lập. Middleware không can thiệp vào việc quản lý hệ điều hành mà chỉ hỗ trợ kết nối và chia sẻ tài nguyên.
- **Ví dụ về Middleware**: Các hệ thống như **CORBA**, **RPC**, và **Java RMI** cung cấp khả năng kết nối và giao tiếp giữa các hệ thống khác nhau mà không yêu cầu phải có một hệ điều hành phân tán thực sự.

**Ví dụ về Middleware**: Các hệ thống như **CORBA**, **RPC**, và **Java RMI** cung cấp khả năng kết nối và giao tiếp giữa các hệ thống khác nhau mà không yêu cầu phải có một hệ điều hành phân tán thực sự.

**Câu hỏi 6:** Trong mô hình kiến trúc phân tầng OSI của Mạng máy tính, hãy trình bày tóm tắt chức năng của từng tầng. Lấy ví dụ cụ thể khi chúng ta thay đổi/cập nhật một tầng bất kỳ thì không ảnh hưởng đến hoạt động của các tầng khác.

## 1. Tầng Vật lý (Physical Layer):

- **Chức năng**: Chuyển đổi và truyền tải các bit (dạng tín hiệu điện, ánh sáng, hoặc sóng radio) qua các phương tiện vật lý như cáp quang, dây đồng, hoặc không dây. Tầng này liên quan đến các thiết bị phần cứng như cáp và bộ chuyển mạch.
- **Ví dụ thay đổi**: Khi thay đổi loại cáp từ cáp đồng sang cáp quang (liên quan đến tầng vật lý), các tầng trên như tầng mạng và tầng vận chuyển vẫn hoạt động bình thường.

## 2. Tầng Liên kết dữ liệu (Data Link Layer):

- **Chức năng**: Đảm bảo việc truyền dữ liệu giữa hai nút liền kề qua cùng một phương tiện vật lý, kiểm soát lỗi phát sinh từ tầng vật lý, đồng thời thực hiện đóng gói dữ liệu vào các **frame** (khung dữ liệu). Các giao thức như Ethernet và PPP hoạt động tại tầng này.
- **Ví dụ thay đổi**: Khi thay đổi giao thức liên kết dữ liệu từ Ethernet sang Wi-Fi (liên quan đến tầng liên kết dữ liệu), tầng mạng và các tầng cao hơn vẫn không bị ảnh hưởng.

## 3. Tầng Mạng (Network Layer):

- **Chức năng**: Định tuyến và chuyển tiếp các gói tin qua mạng từ nguồn đến đích, có thể đi qua nhiều mạng khác nhau. Địa chỉ IP được sử dụng ở tầng này. Giao thức chính bao gồm IP (Internet Protocol).
- **Ví dụ thay đổi**: Nếu thay đổi địa chỉ IP của máy tính, các tầng như liên kết dữ liệu hay tầng vận chuyển vẫn không bị ảnh hưởng.

## 4. Tầng Vận chuyển (Transport Layer):

- **Chức năng**: Quản lý việc truyền dữ liệu từ đầu đến cuối giữa các ứng dụng. Tầng này đảm bảo rằng dữ liệu được truyền nguyên vẹn và theo đúng thứ tự. Các giao thức phổ biến là TCP (giao thức kết nối) và UDP (giao thức không kết nối).
- **Ví dụ thay đổi**: Khi chuyển đổi từ giao thức TCP sang UDP cho một ứng dụng cụ thể, các tầng dưới như mạng hoặc liên kết dữ liệu vẫn không bị ảnh hưởng.

## 5. Tầng Phiên (Session Layer):

- **Chức năng**: Quản lý việc thiết lập, duy trì và kết thúc phiên giao tiếp giữa hai ứng dụng. Nó cung cấp khả năng điều khiển phiên và đồng bộ hóa dữ liệu giữa hai bên.
- **Ví dụ thay đổi**: Nếu một ứng dụng thay đổi cách thiết lập và duy trì phiên (thay đổi tầng phiên), tầng vận chuyển vẫn sẽ tiếp tục đảm bảo việc truyền tải dữ liệu mà không bị ảnh hưởng.

## 6. Tầng Trình bày (Presentation Layer):

- **Chức năng**: Chuyển đổi dữ liệu giữa định dạng của ứng dụng và định dạng chung mà mạng có thể hiểu được. Tầng này thực hiện các tác vụ như mã hóa, giải mã, và nén dữ liệu.
- **Ví dụ thay đổi**: Khi thay đổi cách mã hóa dữ liệu (liên quan đến tầng trình bày), tầng ứng dụng và các tầng dưới (như tầng vận chuyển) vẫn tiếp tục hoạt động mà không bị ảnh hưởng.

## 7. Tầng Ứng dụng (Application Layer):

- **Chức năng**: Cung cấp giao diện trực tiếp giữa người dùng và mạng, bao gồm các giao thức cho các ứng dụng mạng như HTTP, FTP, SMTP. Đây là tầng mà các ứng dụng cuối cùng sử dụng để tương tác với mạng.
- **Ví dụ thay đổi**: Nếu một trang web chuyển từ giao thức HTTP sang HTTPS (liên quan đến tầng ứng dụng), các tầng dưới như tầng vận chuyển hoặc tầng mạng không bị thay đổi hay ảnh hưởng.

**Câu hỏi 7:** Cho ví dụ và phân tích một mô hình kiến trúc thuê bao/xuất bản (publish/subscribe).

**Mô hình kiến trúc thuê bao/xuất bản (publish/subscribe)** là một kiến trúc truyền thông trong hệ thống phân tán, nơi các thành phần hệ thống giao tiếp thông qua việc xuất bản (publish) và đăng ký nhận thông tin (subscribe) mà không cần biết về sự tồn tại của nhau. Đây là một kiến trúc không đồng bộ, thường được sử dụng để phân phối dữ liệu và thông báo theo sự kiện một cách hiệu quả.

## 1. Ví dụ về mô hình Publish/Subscribe

Một ví dụ cụ thể về mô hình này là hệ thống **gửi thông báo thời tiết**:

- **Nhà cung cấp dịch vụ thời tiết** (nhà xuất bản) phát ra thông tin thời tiết mới nhất (như cảnh báo bão, nhiệt độ, độ ẩm).
- **Người dùng** (người đăng ký) quan tâm đến một loại thông tin cụ thể, như cảnh báo bão, đăng ký để nhận được thông báo về cảnh báo bão cho khu vực của họ.
- Khi có sự kiện xảy ra (ví dụ, bão sắp đến), nhà cung cấp dịch vụ thời tiết sẽ **xuất bản** thông báo về sự kiện đó. Tất cả những người đăng ký liên quan đến cảnh báo bão sẽ **nhận** được thông tin mà họ quan tâm mà không cần tương tác trực tiếp với nhà xuất bản.

## 2. Phân tích mô hình kiến trúc Publish/Subscribe

- **Thành phần chính**:
  - **Nhà xuất bản (Publisher)**: Là thành phần phát ra các sự kiện hoặc thông tin mà không cần biết ai sẽ nhận thông tin.
  - **Người đăng ký (Subscriber)**: Là các thành phần đăng ký với hệ thống để nhận thông tin hoặc sự kiện mà họ quan tâm.
  - **Broker (Bộ điều phối)**: Một trung gian giữa nhà xuất bản và người đăng ký, chịu trách nhiệm chuyển tiếp các sự kiện từ nhà xuất bản đến những người đăng ký tương ứng. Broker giúp ngắt kết nối giữa nhà xuất bản và người đăng ký, đảm bảo họ không cần biết đến nhau.
- **Cách thức hoạt động**:
  1. **Người đăng ký đăng ký chủ đề (Topic)**: Người dùng hoặc các thành phần hệ thống khác nhau đăng ký nhận các thông tin liên quan đến một chủ đề cụ thể mà họ quan tâm.
  2. **Nhà xuất bản phát ra thông tin**: Nhà xuất bản phát thông báo, dữ liệu, hoặc sự kiện liên quan đến một chủ đề nào đó.
  3. **Broker phân phối thông tin**: Broker nhận thông tin từ nhà xuất bản và chuyển thông tin đó đến tất cả những người đăng ký chủ đề đó. Sự phân phối này thường dựa trên một số tiêu chí, như chủ đề (topic-based), hoặc nội dung (content-based).
- **Các kiểu publish/subscribe**:
  - **Topic-based**: Người đăng ký chỉ định rõ họ muốn đăng ký vào một chủ đề nhất định (ví dụ, "cảnh báo thời tiết").
  - **Content-based**: Người đăng ký có thể định nghĩa rõ họ muốn nhận những thông tin nào dựa trên nội dung của thông báo (ví dụ, chỉ nhận cảnh báo bão ở một khu vực nhất định).

## 3. Ưu điểm của mô hình Publish/Subscribe:

- **Khả năng mở rộng**: Do không có mối liên kết trực tiếp giữa nhà xuất bản và người đăng ký, hệ thống có thể mở rộng rất tốt, với số lượng lớn người đăng ký và nhà xuất bản.
- **Tính không đồng bộ**: Người đăng ký không cần phải trực tiếp kết nối hoặc biết thời gian xuất bản chính xác, họ có thể nhận thông tin bất cứ khi nào nó được xuất bản.
- **Tính tách biệt**: Nhà xuất bản và người đăng ký không cần biết về sự tồn tại của nhau, tạo ra sự linh hoạt trong phát triển và bảo trì hệ thống.

- **Dễ dàng quản lý sự kiện**: Hệ thống có thể quản lý các sự kiện một cách hiệu quả, đảm bảo rằng thông tin đến đúng đối tượng quan tâm.

## 4. Nhược điểm của mô hình Publish/Subscribe:

- **Khó khăn trong đảm bảo tin cậy**: Đối với các hệ thống đòi hỏi độ tin cậy cao (như hệ thống giao dịch tài chính), mô hình này có thể gặp khó khăn trong việc đảm bảo rằng tất cả người đăng ký đều nhận được thông tin.
- **Tính phức tạp của broker**: Broker phải xử lý các yêu cầu và thông báo từ nhiều nhà xuất bản và người đăng ký cùng một lúc, điều này có thể làm tăng độ phức tạp và yêu cầu tài nguyên lớn.
- **Bảo mật và quyền riêng tư**: Việc chia sẻ thông tin một cách rộng rãi đòi hỏi các cơ chế bảo mật tốt để tránh rò rỉ dữ liệu hoặc bị tấn công.

## 5. Ví dụ thực tế về Publish/Subscribe:

- **Hệ thống gửi thông báo thời tiết** (ví dụ đã đề cập).
- **Hệ thống tin nhắn MQTT**: MQTT (Message Queuing Telemetry Transport) là giao thức phổ biến trong Internet of Things (IoT), nơi các cảm biến (nhà xuất bản) gửi dữ liệu liên tục và các thiết bị khác (người đăng ký) nhận thông tin để xử lý sự kiện.
- **Hệ thống nhận email thông báo**: Ví dụ, khi bạn đăng ký nhận email từ một website, website đó sẽ xuất bản các thông tin mới (như bài viết mới, sản phẩm mới), và bạn nhận được thông báo trong hộp thư email của mình.

**Câu hỏi 8:** Sự khác nhau giữa phân tán dọc và phân tán ngang là gì?

## 1. Phân tán dọc (Vertical Scaling):

- **Khái niệm**: Phân tán dọc liên quan đến việc **tăng cường sức mạnh cho một máy chủ duy nhất** bằng cách nâng cấp phần cứng (RAM, CPU, lưu trữ) hoặc phần mềm để máy chủ đó có thể xử lý nhiều công việc hơn.
- **Cách thực hiện**:
  - Thay thế các thành phần phần cứng của máy chủ hiện có (ví dụ, tăng thêm RAM, nâng cấp CPU mạnh hơn).
  - Sử dụng máy chủ mạnh hơn thay vì chia tải cho nhiều máy.
- **Ưu điểm**:
  - Đơn giản trong việc quản lý và triển khai vì chỉ cần xử lý một máy chủ duy nhất.
  - Không cần thay đổi kiến trúc ứng dụng hiện tại nhiều.
- **Nhược điểm**:
  - **Giới hạn phần cứng**: Có giới hạn về khả năng nâng cấp phần cứng, do máy chủ không thể mở rộng mãi.
  - **Tăng chi phí**: Việc nâng cấp phần cứng có thể trở nên rất đắt đỏ khi đến một mức độ nào đó.
  - **Điểm nghẽn**: Máy chủ duy nhất có thể trở thành điểm tắc nghẽn nếu hệ thống quá tải.

- **Ví dụ**: Một công ty nâng cấp máy chủ từ 8 CPU lên 16 CPU để tăng khả năng xử lý của hệ thống cơ sở dữ liệu.

## 2. Phân tán ngang (Horizontal Scaling):

- **Khái niệm**: Phân tán ngang liên quan đến việc **mở rộng hệ thống bằng cách thêm nhiều máy chủ** (hoặc nút) hơn vào hệ thống để chia tải công việc giữa chúng.
- **Cách thực hiện**:
  - Thêm các máy chủ mới để chia sẻ công việc, thường sử dụng các giải pháp như cân bằng tải (load balancer) để phân phối yêu cầu tới các máy chủ.
  - Dữ liệu có thể được phân phối hoặc nhân bản giữa các nút.
- **Ưu điểm**:
  - **Khả năng mở rộng tốt**: Không có giới hạn cứng về khả năng mở rộng vì có thể thêm nhiều máy chủ một cách dễ dàng.
  - **Tăng tính sẵn sàng**: Nếu một máy chủ gặp sự cố, các máy khác có thể tiếp tục xử lý công việc, tăng tính dự phòng và khả năng chịu lỗi.
- **Nhược điểm**:
  - **Quản lý phức tạp hơn**: Cần các hệ thống quản lý phân tán và các công cụ cân bằng tải, đồng bộ dữ liệu giữa các nút.
  - **Độ trễ trong truyền thông**: Việc phân phối dữ liệu giữa các nút có thể tạo ra độ trễ, ảnh hưởng đến hiệu suất nếu không được tối ưu hóa.
- **Ví dụ**: Một hệ thống web server có thể thêm nhiều máy chủ mới để phục vụ các yêu cầu người dùng, sử dụng load balancer để phân phối yêu cầu giữa các máy chủ.

**Câu hỏi 9:** Phân tích ưu nhược điểm của kiến trúc tập trung và kiến trúc không tập trung.

## 1. Kiến trúc tập trung (Centralized Architecture)

*Khái niệm:*

Trong kiến trúc tập trung, mọi hoạt động xử lý dữ liệu, lưu trữ và quản lý hệ thống đều tập trung vào một hoặc một vài máy chủ trung tâm (central server). Các máy trạm hoặc thiết bị đầu cuối sẽ kết nối và gửi yêu cầu đến máy chủ này để xử lý.

*Ưu điểm:*

- **Dễ quản lý**: Vì mọi tài nguyên và dữ liệu đều nằm tại một điểm tập trung, việc quản lý, bảo trì, và nâng cấp hệ thống trở nên dễ dàng hơn. Nhà quản trị có thể kiểm soát hệ thống từ một vị trí duy nhất.
- **Bảo mật tốt hơn**: Kiến trúc tập trung giúp việc bảo mật dữ liệu trở nên hiệu quả hơn do mọi thông tin đều được lưu trữ tại một nơi, dễ dàng kiểm soát truy cập và thực thi các chính sách bảo mật.
- **Tối ưu hóa tài nguyên**: Tài nguyên được sử dụng tập trung, dễ dàng tối ưu và tránh lãng phí tài nguyên do không có sự phân tán.
- **Đồng bộ dễ dàng**: Việc duy trì tính nhất quán dữ liệu và đồng bộ hóa dễ thực hiện hơn vì toàn bộ dữ liệu tập trung tại một vị trí.

*Nhược điểm:*

- **Điểm đơn thất bại (Single Point of Failure)**: Hệ thống tập trung có một điểm yếu lớn là nếu máy chủ trung tâm bị hỏng hoặc bị tấn công, toàn bộ hệ thống sẽ ngừng hoạt động. Điều này có thể gây ra sự cố nghiêm trọng cho doanh nghiệp hoặc dịch vụ.
- **Khả năng mở rộng hạn chế**: Hệ thống tập trung gặp khó khăn khi cần mở rộng, đặc biệt khi số lượng yêu cầu và người dùng tăng lên nhanh chóng. Máy chủ trung tâm có thể trở thành điểm tắc nghẽn nếu không đủ tài nguyên xử lý.
- **Độ trễ cao**: Khi có quá nhiều người dùng hoặc thiết bị yêu cầu tài nguyên từ máy chủ trung tâm, độ trễ trong việc xử lý yêu cầu có thể tăng cao, ảnh hưởng đến hiệu suất.
- **Phụ thuộc vào kết nối mạng**: Máy trạm phải luôn kết nối với máy chủ trung tâm, điều này gây khó khăn nếu kết nối mạng không ổn định hoặc bị gián đoạn.

*Ví dụ:*

- **Hệ thống quản lý tài chính tập trung**: Trong nhiều tổ chức, tất cả các giao dịch tài chính đều được xử lý tại một máy chủ trung tâm. Các phòng ban gửi yêu cầu đến máy chủ trung tâm để xử lý các giao dịch này.
- **Hệ thống thông tin quản lý nhân sự**: Thông tin về nhân viên được lưu trữ trên một máy chủ duy nhất, từ đó mọi bộ phận có thể truy cập và quản lý thông tin này từ máy chủ.

---

## 2. Kiến trúc không tập trung (Decentralized Architecture)

*Khái niệm:*

Kiến trúc không tập trung, còn được gọi là **phân tán**, trong đó dữ liệu và các tài nguyên xử lý không được tập trung tại một điểm duy nhất mà được phân phối trên nhiều nút hoặc máy chủ khác nhau. Các nút có thể hoạt động độc lập hoặc kết hợp với nhau để xử lý các yêu cầu từ người dùng.

*Ưu điểm:*

- **Khả năng chịu lỗi tốt (Fault Tolerance)**: Hệ thống không tập trung không có điểm đơn thất bại. Nếu một hoặc nhiều nút gặp sự cố, các nút còn lại vẫn có thể hoạt động, đảm bảo tính liên tục của dịch vụ.
- **Khả năng mở rộng cao**: Hệ thống không tập trung dễ dàng mở rộng bằng cách thêm nhiều nút hơn vào hệ thống. Điều này giúp hệ thống phân tán có thể xử lý lượng lớn yêu cầu và người dùng mà không làm giảm hiệu suất.
- **Giảm tải cho từng nút**: Nhờ việc phân chia công việc xử lý giữa các nút, tải trọng được phân tán đều đặn, tránh tắc nghẽn tại một điểm duy nhất.
- **Giảm độ trễ**: Trong hệ thống không tập trung, các yêu cầu có thể được xử lý tại các nút gần với người dùng nhất, giúp giảm độ trễ và cải thiện hiệu suất.
- **Tính sẵn sàng cao**: Nhờ sự phân tán, ngay cả khi một số nút bị gián đoạn hoặc hỏng, các nút khác vẫn có thể hoạt động bình thường, đảm bảo dịch vụ luôn sẵn sàng.

*Nhược điểm:*

- **Quản lý phức tạp hơn**: Do có nhiều nút hoạt động độc lập, việc quản lý, bảo trì và cập nhật hệ thống không tập trung trở nên khó khăn hơn. Việc đồng bộ dữ liệu giữa các nút cũng cần được chú ý.
- **Bảo mật phức tạp**: Với dữ liệu phân tán trên nhiều nút, việc đảm bảo an ninh mạng và bảo vệ dữ liệu trở nên phức tạp hơn so với hệ thống tập trung. Cần có các giải pháp bảo mật cho từng nút cũng như cơ chế phân quyền chặt chẽ.
- **Khó đồng bộ dữ liệu**: Khi có nhiều nút hoạt động độc lập, việc đồng bộ hóa dữ liệu giữa chúng có thể gặp khó khăn, đặc biệt trong các hệ thống yêu cầu dữ liệu nhất quán cao.

*Ví dụ:*

- **Blockchain**: Là ví dụ điển hình của hệ thống không tập trung, trong đó dữ liệu được lưu trữ trên nhiều nút khác nhau và không có điểm tập trung duy nhất. Mỗi nút trong mạng blockchain đều có một bản sao của sổ cái.
- **Hệ thống chia sẻ tệp P2P (Peer-to-Peer)**: Các hệ thống như BitTorrent hoạt động dựa trên mô hình không tập trung, trong đó mỗi máy tính người dùng đều có thể đóng vai trò như một nút, tải về và chia sẻ dữ liệu trực tiếp với các nút khác mà không cần máy chủ trung tâm.

**Câu hỏi 10:** Trong một mạng overlay có cấu trúc, các thông điệp được định tuyến dựa theo hình trạng mạng (topology). Nhược điểm quan trọng của hướng tiếp cận này là gì?

## Nhược điểm chính của hướng tiếp cận này là:

1. **Khả năng chịu lỗi thấp hơn**:
   - Vì các nút và kết nối trong mạng overlay có cấu trúc phụ thuộc chặt chẽ vào một cấu trúc định trước, nếu một số nút hoặc liên kết trong mạng gặp sự cố (như mất kết nối hoặc hỏng hóc), điều này có thể ảnh hưởng lớn đến quá trình định tuyến và làm giảm hiệu suất của mạng.
   - Quá trình phục hồi khi một nút thất bại thường phức tạp hơn so với mạng overlay không cấu trúc vì phải duy trì đúng hình trạng mạng.
2. **Phức tạp trong việc bảo trì cấu trúc**:
   - Để duy trì mạng trong một cấu trúc xác định (như vòng DHT - Distributed Hash Table), hệ thống phải liên tục cập nhật và tái cấu trúc khi có sự gia nhập hoặc rời bỏ của các nút (node churn). Điều này làm tăng chi phí bảo trì mạng và độ phức tạp của các thuật toán.
   - Việc bảo đảm tính nhất quán của mạng và tránh các lỗi định tuyến đòi hỏi các nút phải thường xuyên giao tiếp để đồng bộ, gây ra chi phí liên lạc cao.
3. **Khó khăn trong việc mở rộng**:
   - Khi mạng mở rộng hoặc co lại một cách nhanh chóng, việc duy trì hình trạng cố định của mạng có thể gặp nhiều thách thức. Mỗi lần có nút mới tham gia hoặc rời đi, cần có các cơ chế để tái cân bằng cấu trúc mạng, điều này có thể làm chậm trễ quá trình xử lý các yêu cầu.
4. **Định tuyến cứng nhắc**:

- Vì định tuyến dựa vào một hình trạng cố định, các tuyến đường đi giữa các nút có thể không linh hoạt và tối ưu trong mọi tình huống. Ví dụ, một số đường dẫn có thể trở nên quá tải hoặc tắc nghẽn, nhưng do cấu trúc cứng nhắc, hệ thống không thể dễ dàng tìm các đường dẫn khác thay thế.

5. **Hiệu suất kém trong môi trường động**:
   - Mạng overlay có cấu trúc không phù hợp lắm trong môi trường có nhiều thay đổi về nút (như mạng P2P hoặc mạng di động), vì sự thay đổi liên tục của nút yêu cầu hệ thống phải tái cấu trúc lại mạng, làm giảm hiệu suất tổng thể và tăng chi phí xử lý.

**Câu hỏi 11:** Xét một chuỗi các tiến trình P1, P2, ..., Pn triển khai một kiến trúc client-server đa tầng. Cơ chế hoạt động của tổ chức đó như sau: tiến trình Pi là client của tiến trình Pi+1, và Pi sẽ trả lời Pi-1 chỉ khi đã nhận được câu trả lời từ Pi+1.

Vậy những vấn đề nào sẽ nảy sinh với tổ chức này khi xem xét hiệu năng yêu cầu-trả lời tới P1?

## 1. Độ trễ tích lũy:

- Do mỗi tiến trình **Pi** chỉ trả lời cho **Pi-1** sau khi nhận được câu trả lời từ **Pi+1**, thời gian cần thiết để hoàn tất một yêu cầu từ **P1** đến **Pn** sẽ tăng lên theo mỗi tầng.
- Độ trễ của toàn bộ hệ thống là tổng độ trễ của từng tầng. Nếu mỗi tầng có độ trễ cố định, tổng độ trễ sẽ tăng tuyến tính theo số tầng, dẫn đến độ trễ tổng thể lớn hơn so với kiến trúc ít tầng hơn.

**Ví dụ**: Giả sử mỗi tiến trình mất 100 ms để xử lý một yêu cầu và gửi câu trả lời cho tiến trình trước, với 5 tầng, tổng độ trễ sẽ là 500 ms.

## 2. Hiệu ứng nút cổ chai:

- Nếu một tầng (ví dụ, **Pi**) có hiệu suất xử lý kém hơn hoặc gặp vấn đề về tài nguyên (CPU, bộ nhớ), nó sẽ trở thành **nút cổ chai** và làm chậm toàn bộ hệ thống.
- Tầng này sẽ khiến tất cả các yêu cầu bị đình trệ, vì các tầng phía trên phụ thuộc vào việc nhận được câu trả lời từ tầng này trước khi có thể gửi phản hồi.

**Ví dụ**: Nếu **P3** xử lý yêu cầu chậm hơn **P2** và **P1**, tốc độ phản hồi của toàn bộ hệ thống sẽ bị ảnh hưởng bởi tốc độ chậm của **P3**.

## 3. Khả năng chịu lỗi kém:

- Trong tổ chức này, nếu một tiến trình **Pi** gặp sự cố hoặc không thể trả lời, các tiến trình phía trên (từ **P1** đến **Pi-1**) sẽ không thể nhận được câu trả lời và do đó cũng không thể trả lời yêu cầu ban đầu. Điều này tạo ra **sự phụ thuộc mạnh mẽ** giữa các tầng, và một lỗi ở bất kỳ tầng nào cũng có thể gây ra lỗi toàn hệ thống.

**Ví dụ**: Nếu **P4** gặp lỗi và không thể trả lời **P3**, thì **P1**, **P2**, và **P3** cũng sẽ không thể hoàn tất quá trình phản hồi.

## 4. Tính đồng bộ hóa và tắc nghẽn:

- Các tiến trình sẽ phải **chờ đợi** phản hồi từ tầng tiếp theo trước khi có thể phản hồi yêu cầu từ tầng trước đó. Điều này gây ra tắc nghẽn trong hệ thống, đặc biệt là khi một tiến trình xử lý lâu hoặc gặp sự cố, khiến các tiến trình phía trên phải chờ đợi không cần thiết.
- Nếu có nhiều yêu cầu cùng được gửi đi từ các tiến trình khác nhau trong chuỗi, các yêu cầu có thể xếp hàng chờ nhau, gây tắc nghẽn hệ thống.

**Ví dụ**: Nếu nhiều yêu cầu cùng được gửi đến **P1**, nhưng **P3** đang xử lý chậm, toàn bộ hệ thống sẽ bị ảnh hưởng do phải đồng bộ hóa phản hồi.

## 5. Khó khăn trong mở rộng:

- Khi hệ thống ngày càng có nhiều tầng (do yêu cầu mở rộng hoặc thêm dịch vụ mới), độ phức tạp của hệ thống tăng lên đáng kể. Đặc biệt, việc duy trì hiệu suất đồng đều giữa các tầng rất khó khăn và dễ dẫn đến tình trạng mất cân bằng tài nguyên giữa các tầng, gây giảm hiệu năng.

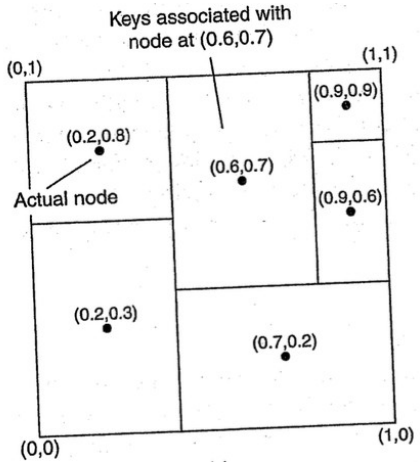
**Ví dụ**: Khi thêm các tầng mới vào hệ thống, có thể cần phải điều chỉnh cơ sở hạ tầng, như phân bổ thêm tài nguyên để đảm bảo tất cả các tầng đều hoạt động với tốc độ tương đương.

## 6. Chi phí truyền thông cao:

- Mỗi lần một yêu cầu phải được gửi qua nhiều tiến trình trước khi có thể nhận được phản hồi, **chi phí truyền thông** (thời gian, băng thông mạng, tài nguyên tính toán) sẽ tăng lên.
- Đặc biệt, trong hệ thống phân tán với các tiến trình được triển khai trên nhiều máy khác nhau, việc truyền thông qua mạng có thể tạo ra **chi phí lớn** và làm giảm hiệu năng tổng thể.

**Ví dụ**: Nếu mỗi tiến trình nằm trên một máy chủ khác nhau, thời gian truyền thông qua mạng giữa các tầng sẽ góp phần lớn vào tổng độ trễ.

**Câu hỏi 12:** Xét mạng CAN như trong hình. Giả sử tất cả các node đều biết node hàng xóm của mình. Một giải thuật định tuyến được đưa ra đó là gửi các gói tin cho node hàng xóm gần mình nhất và hướng đến đích. Giải thích.




Có thể tính được đường đi ngắn nhất từ (0.2, 0.3) đến (0.9, 0.9) là qua (0.6, 0.7), tuy nhiên theo giải thuật CAN thì gửi gói tin cho node hàng xóm gần nhất trước nên tuyến đường sẽ đi qua (0.2, 0.8). Vì vậy có thể thấy giải thuật này chưa phải giải thuật tốt nhất. Tổng khoảng cách từ đầu tới cuối tối ưu nhất chưa chắc đã đi qua những điểm gần nó nhất.