**BÀI TẬP CHƯƠNG 1: TỔNG QUAN HỆ PHÂN TÁN**

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

- Dịch vụ chia sẻ file P2P (peer – to – peer) như Torrent. Vì: Trong dịch vụ này, tài nguyên không được lưu trữ toàn bộ tại một nơi nào đó mà được phân tán ra các client. Khi một người dung muốn tải file thì sẽ tải trực tiếp từ các client khác. Có một tracker sẽ đóng vai trò thông báo thông tin tài nguyên giữa các client.

- Dịch vụ phân giải tên miền DNS. Vì: Toàn bộ dữ liệu về IP và tên miền không được lưu trữ cục bộ tại một máy chủ mà được phân tán rộng rãi, có thể ở dạng phân cấp giúp cho việc truy vấn được nhanh hơn.

**Câu 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.**

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 vì:

- Trong hệ phân tán, việc chia sẻ tài nguyên của hệ thống (gồm các thiết bị phần cứng như ô đĩa, máy in... tới các đối tượng như file, cửa sổ, cơ sỡ dữ liệu...) được chia sẻ (sủ dụng chung) mà không bị hạn chế bởi tình trạng phân tán tài nguyên theo vị trí địa lí.

=>Giảm chi phí, tăng tính sẵn sàng và hỗ trợ làm việc nhóm tốt hơn.  
 - Tuy nhiên, việc sử dụng chung tài nguyên này lại không bị hạn chế, nên nếu kẻ xấu có thể dễ dàng lấy dữ liệu dùng chung, chính vì vậy dẫn tới việc tăng rủi ro về mặt an toàn thông tin.

- Để chia sẻ tài nguyên một cách hiệu quả thì mỗi tài nguyên cần được quản lý bởi một chương trình có giao diện truyền thông, các tài nguyên có thể truy nhập, cập nhật một cách tin cậy và nhất quán.

**Câu 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.**

- Tính trong suốt của Hệ phân tán có mục tiêu là che khuất đi những chi tiết phụ thuộc hệ thống mà không thích hợp với người dùng trong mọi hoàn cảnh và tạo ra môi trường thuần nhất với người dùng.

- Sư che khuất thông tin phụ thuộc hệ thống khỏi người dùng dưa trên việc cân bằng tính đơn giản và tính hiểu quả nhưng 2 tính chất này lạ xung đột với nhau.

- Tính trong suốt cung cấp cho người dùng 1 cái nhìn logic nhất của một hệ thống vật lý hồn tạp nhờ rút gọn, chia cắt vật lý của các đối tượng nhưng khi chia cắt vật lý thì làm cho hiệu năng bị giảm sút đi.

=> Chính vì vậy, cần phải cân bằng giữ trong suốt và hiệu năng.

**Câu 4: Tại sao giao diện (Interface) lại quan trọng đối với Tính mở của Hệ Phân Tán?**

- Tính mở của Hệ phân tán được hiểu là hệ thống có thể được tạo nên từ nhiều loại phần cứng và phần mềm của hiều nhà clun câp khác nhau (nhưng theo một tiêu chuẩn chung).

- Tính mở được xem xét theo mức độ bổ sung thêm các dịch vụ chia sẻ tài nguyên mà không phá hỏng hay nhân đôi lên dịch vụ đang tồn tại.

- Tính mở được được hoàn thiên bằng cách xác định hay phân rõ các giao diện chính của hệ phan tán và làm cho nó tương thích với các nhà phát triển phần mềm (tức là giao diện chính của hệ điều hành phân tán cần phổ dụng).

- Do hệ thống được tạo nên từ nhiều loại phần cứng và phần mềm từ nhiều nhà cung cấp khác nhau nên muốn thi hành được tính mở của hệ phân tán cần dựa trên việc cung cấp cơ chế truyền thông và công khai các giao diện người dùng để truy cập tài nguyên chung.

=> Giao diện (Interface) là quan trọng đối với Tính mở của Hệ phân tán.

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

So sánh DOS và NOS:

|  |  |  |
| --- | --- | --- |
| Hệ thống | Mô tả | Mục tiêu |
| DOS | Hệ điều hành liên kết chặt dùng cho các hệ thống máy tính thuần nhất | Che giấu và quản lý các tài nguyên phần cứng |
| NOS | Hệ điều hành liên kết lỏng, dùng cho các máy tính không thuần nhất (mạng LAN và mạng WAN) | Cung cấp các dịch vụ cục bộ cho các máy tính khác truy nhập từ xa. |

|  |  |  |  |
| --- | --- | --- | --- |
| Item | DOS | | NOS |
| Multiprocessor | Multicomputer |
| Mức độ trong suốt | Rất cao | Cao | Thấp |
| Một HĐH trên các nút | YES | YES | NO |
| Số lượng bản HĐH | 1 | N | N |
| Trao đổi thông tin | Bộ nhớ chia sẻ | Chuyển thông báo | Tệp |
| Quản lý tài nguyên | Toàn cục tập trung | Toàn cục phân tán | Theo nút |
| Co giãn | Không | Có thể | Có |
| Mở | Đóng | Đóng | Mở |

  Middlewave là sự kết hợp ưu điểm của cả 2 mô hình trên. Vì: Middleware thừa kế tính trong suốt của DOS, tính mở của NOS -> Thừa kế ưu điểm của cả 2

**Câu 6: Trình bày về những cơ chế “phân tán” có trong nền tảng Blockchain, ứng dụng vào mô hình sử dụng Bitcoin.**

a. Cơ chế phân tán trong Blockchain:

- Công nghệ Blockchain giải quyết được vấn đề chông gian lận trong giao dịch mà không cần tới bên trung gian thư 3 tin cậy.

- Blockchain là một cuốn sổ cái ghi lại tất cả các giao dịch. Dữ liệu trong cuốn sổ cái liên tục được mạng lưới máy tính ngang hàng trên thế giới cập nhật và bảo trì. Giao dịch khi A gửi X bitcoin cho B được ghi lại trên toàn hệ thống, tất cả các máy tính trong mạng này sẽ xác minh và ghi lại giao dịch đó vào cuốn sổ cái rồi cấp phát dữ liệu này tới các máy tính khác. Blockchain là một cơ sở dữ liệu phân tán vô chủ; các máy tính liên tục thực hiện việc kiểm toán độc lập bằng cách xác minh dữ liệu nhận tới và so sánh với chữ ký của giao dịch đó.

- Về công nghệ, các giao dịch được xác minh bởi thuật toán chữ ký số dựa trên đường cong Elliptic (ECDSA) và được xác nhận bởi chuỗi các quá trình xử lý lần lượt các hàm băm SHA256 bởi các thợ đào Bitcoin. Mỗi khối trong blockchain chứa tất cả thông tin giao dịch trong khối đó trong 1 cây Merkle – là một cây nhị phân có thứ tự được xây dựng từ một dãy các đối tượng dữ liệu sử dụng hàm băm – để đạt hiệu quả cao trong việc lưu trữ và xác minh với lượng dữ liệu lớn các giao dịch. Khi có một giao dịch không hợp lệ, hệ thống sẽ loại bỏ nó bằng cách chọn theo số đông. Càng có nhiều máy tính tham gia vào hệ thống ngang hàng cho blockchain thì sức mạnh xử lý và độ an toàn của hệ thống blockchain đó càng cao.

b. Ứng dụng vào mô hình sử dụng Bitcoin:

- Bất kỳ ai sở hữu Bitcoin đều được gán ít nhất một địa chỉ Bitcoin, nơi lưu trữ và ghi nhận trọng “ví” (wallet).

- Ví có địa chỉ công khai (public key) và khoá riêng tư (private key). Bất kỳ ai cũng có thể gửi Bitcoin đến một chiếc ví bằng địa chỉ công khai, còn khoá riêng tư phải được nhập khi chủ Ví muốn gửi Bitcoin đi.

- Vì vậy, việc sở hữu Bitcoin được định nghĩa là sự nắm giữ khoá riêng tư của 1 địa chỉ Bitcoin.

- Một khi khoá riêng tư bị mất, mạng lưới Bitcoin sẽ không thể xác nhận được việc sở hữu số bitcoin đó, và số bitcoin trong địa chỉ đó sẽ vĩnh viễn bị mất.

- Ví cho phép người dùng hoàn tất thanh toán giữa các địa chỉ khác nhau bằng cách cập nhật vào blockchain.

**Bài tập chương 2: Kiến trúc**

**Câu 1: 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.**

Chức năng của từng tầng trong mô hình phân tầng OSI:

* Tầng vật lý: Truyền dòng bit giữa các máy bằng đường truyền vật lý.
* Tầng liên kết dữ liệu: Chia nhỏ thành các khối dữ liệu, gắn cho mỗi khối các thông tin; Phát hiện lỗi trong việc truyền dữ liệu (thiếu, mất).
* Tầng mạng: Lập địa chỉ các thông điệp, diễn dịch địa chỉ và tên logic thành địa chỉ vật lý.
* Tầng giao vận: Thực hiện kiểm soát lỗi, kiểm soát luồng dữ liệu.
* Tầng phiên: Cung cấp phương tiện truyền thông giữa các ứng dụng.
* Tầng trình diễn: Quyết định dạng thức trao đổi dữ liệu giữa các máy tính mạng; chuyển đổi giao thức, biên dịch dữ liệu, mã hóa dữ liệu.
* Tầng ứng dụng: Cung cấp các phương tiện để người sử dụng truy nhập được vào môi trường OSI.

Ví dụ khi thay đổi một tầng bất kì thì không ảnh hưởng đến hoạt động của các tầng khác: Nếu thay đường truyền vật lý từ cáp đồng sang cáp quang thì các tầng khác vẫn không bị ảnh hưởng.

**Câu 2: Cho ví dụ và phân tích một mô hình kiến trúc thuê bao/xuất bản (publish/subscribe).**

-Thành phần hệ thống trao đổi thông tin với nhau thông qua các sự kiện   
-Các sự kiện chứa các thông tin cần trao đổi   
-Các sự kiện có thể kích hoạt các thao tác trong các tiến trình   
-Có thể thực hiện theo mô hình điểm điểm hoặc mô hình trục quảng bá sự kiện

**Câu 3: Sự khác nhau giữa phân tán dọc và phân tán ngang:**

* Phân tán dọc phân tầng theo chức năng.
* Phân tán ngang phân tầng theo tải.

**Câu 4: Phân tích ưu nhược điểm của kiến trúc tập trung và kiến trúc không tập trung.**

* Kiến trúc tập trung: Vấn đề đăng ký server, trùng lặp yêu cầu của client, sử dụng bộ nhớ trạng thái.
* Kiến trúc không tập trung: Vấn đề phân chia tài nguyên, dữ liệu giữa các máy.

**Câu 5: 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: Liên kết giữa các node chỉ là liên kết logic nên không thấy được các liên kết vật lý bên dưới.

**Câu 6:** **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?**

Các vấn đề nảy sinh khi xem xét hiệu năng yêu cầu – trả lời tới P1:

* Hiệu năng bị ảnh hưởng bởi n-2 tương tác ở phía trên.
* Nếu Pi bị treo thì sẽ không gửi được yêu cầu cho Pi-1.

**Câu 7:** **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 thuật này có tốt không? Giải thích.**

 Giải thuật này có tốt Vì Mạng CAN có khả năng chịu được những lỗi của node hoặc của mạng mà khi đó một node không thể tới được. Vấn đề này được giải quyết bằng cách: hàng xóm của node bị lỗi sẽ quản lý vùng của node bị lỗi, nhưng cơ sở dữ liệu các cặp (key, value) sẽ bị mất. Dữ liệu sẽ được cập nhật lại sau bởi người chủ của dữ liệu.

**Bài tập chương 3: Tiến trình và luồng**

**Câu 1: Có cần thiết phải giới hạn số lượng các luồng trong một tiến trình server?**

Có cần thiết phải giới hạn số lượng các luồng trong một tiến trình. Vì số lượng các luồng ảnh hưởng đến hiệu năng và chi phí lập trình.

**Câu 2: Không nên chỉ gắn một luồng đơn duy nhất với một tiến trình nhẹ.**

**Câu 3:**  **Không nên chỉ có một tiến trình nhẹ đơn gắn với một tiến trình.**

**Câu 4:** **Bài toán này yêu cầu bạn so sánh thời gian đọc một tệp (file) của một máy chủ tập tin (file server) đơn luồng và một máy chủ đa luồng. Phải mất tổng cộng 15 ms để nhận 1 yêu cầu (request) và thực hiện quá trình xử lý, giả định rằng các dữ liệu cần thiết nằm ở bộ nhớ đệm trong bộ nhớ chính. Nếu cần thiết phải thực hiện một thao tác truy cập ổ đĩa thì cần thêm 75 ms, biết rằng việc phải thực hiện thao tác này có xắc suất là 1/3. Hỏi máy chủ có thể nhận bao nhiêu yêu cầu/giây trong 2 trường hợp: máy chủ là đơn luồng và máy chủ là đa luồng (ngoài luồng nhận và xử lý request, sẽ có thêm 1 luồng để truy cập ổ đĩa nếu cần thiết)? Giải thích.**

Server đơn luồng có thể xử lý 1/ (0.015+0.075/3) = 25 yêu cầu /s

Server đa luồng có thể xử lý 1/0.015 = 67 yêu cầu /s

**Câu 5: Hệ thống X chỉ định máy của user chưa server, trong khi các ứng dụng lại được coi như client. Điều đó có vô lý không? Giải thích.**

Không. Vì máy của user có thể làm server để các máy khác truy cập, trong khi các ứng dụng của máy này có thể coi là client, truy cập đến các máy khác.

**Câu 6:** **Giao thức thiết kế cho hệ thống X gặp phải vấn đề về tính mở rộng. Chỉ ra các giải pháp để giải quyết vấn đề đó?**

- Không thêm một chức năng mới nào trừ khi người thực hiện không thể hoàn thành một ứng dụng thật sự nếu thiếu chức năng đó.  
- Việc quyết định hệ thống không phải là gì quan trọng không kém việc quyết định hệ thống là gì.  
- Không phục vụ nhu cầu của cả thế giới, thay vào đó, tạo cho hệ thống có khả năng mở rộng được, sao cho các tính năng bổ sung về sau có thể được đáp ứng theo kiểu tương thích xuôi (tương thích với các phiên bản sau).  
- Điều duy nhất tồi tệ hơn việc tổng quát hóa từ một ví dụ là tổng quát hóa không từ một ví dụ nào cả.  
Nếu không thể hiểu một vấn đề một cách trọn vẹn, thì có lẽ tốt nhất là đừng nên đưa ra một giải pháp nào hết."  
Nếu ta có một giải pháp có thể đạt được 90% hiệu ứng mong muốn cho 10% công việc, thì ta nên chọn giải pháp đơn giản hơn.  
Cô lập các phức tạp một cách hết mức có thể."  
- Cung cấp cơ chế chứ không cung cấp qui chế. Cụ thể, đặt các qui chế về giao diện người dùng vào tay của trình khách.  
Trong thời gian thiết kế X, nguyên lý đầu tiên đã được điều chỉnh thành: Không bổ sung tính năng mới, trừ phi bạn biết một ứng dụng cụ thể nào đó sẽ đòi hỏi tính năng đó.

**Câu 7:** **Với việc xây dựng một server đồng thời, hãy so sánh việc server này tạo một luồng mới và tạo một tiến trình mới khi nhận được yêu cầu từ phía client.**

Server đơn luồng:

+ Chỉ xử lý được một yêu cầu tại một thời điểm  
                          + Các yêu cầu có thể được xử lý tuần tự  
                          + Các yêu cầu có thể được xử lý bởi các tiến trình khác nhau   
                          + Không đảm bảo tính trong suốt  
Server đa luồng:  + Các yêu cầu mới cần chờ quá trình xử lý yêu cầu hiện tại kết thúc.  
                          + Lãng phí thời gian chờ đợi.

**Câu 8: Nếu bây giờ một webserver tổ chức lưu lại thông tin về địa chỉ IP của client và trang web client đó vừa truy cập. Khi có 1 client kết nối với server đó, server sẽ tra xem trong bảng thông tin, nếu tìm thấy thì sẽ gửi nội dung trang web đó cho client. Server này là có trạng thái (stateful) hay không trạng thái (stateless)?**

Đó là server không trạng thái.

**Câu 9: So sánh Docker và Virtual Machine:**

Giống nhau : Đều là công nghê ảo hóa   
Khác nhau : + VMware mô phỏng phần cứng ảo và phải tính đến tất cả các yêu cầu hệ thống cơ bản-sau đó, hình ảnh máy ảo lớn hơn đáng kể so với các container. Điều dó nói rằng cũng có thể chạy nhiều phiên bản hệ điều hành kín đáo song song trên một máy chủ duy nhất với VMWare cho phép các tổ chức xây dựng các giải pháp IaaS thật sự tại nhà.Khả năng di chuyển máy và cách ly lớn hơn, hãy đi với VMware  
                  + Docker không tạo ra toàn bộ hệ điều hành ảo - thay vào đó, tất cả các thành phần yêu cầu không chạy trên máy chủ được đóng gói bên trong container với ứng dụng. Kể từ khi hạt nhân máy chủ được chia sẻ giữa các container Docker, các ứng dụng chỉ vận chuyển với những gì họ cần để chạy-không nhiều hơn, không kém. Điều này làm cho ứng dụng Docker dễ dàng và nhẹ hơn để triển khai và khởi động nhanh hơn so với các máy ảo. Các container của Docker thường nhanh hơn và ít tốn nhiều tài nguyên hơn so với các máy ảo.

**Bài tập chương 4: Communication**

**Câu 1:** **Trong các giao thức phân tầng, mỗi tầng sẽ có một header riêng. Vậy có nên triển khai một hệ thống mà tất cả các header của các tầng đưa chung vào một phần (gọi là header chung), gắn vào đầu mỗi thông điệp để có thể xử lý chung? Giải thích.**

Không nên triển khai một hệ thống như vậy vì nếu làm vậy thì hệ thống sẽ không còn ý nghĩa phân tầng. Các tầng cần độc lập với nhau, do vậy chúng cần có các header riêng biệt.

**Câu 2: Xét 1 thủ tục incr với 2 tham số nguyên. Thủ tục làm nhiệm vụ là cộng 2 tham số đó với nhau. Bây giờ xét trường hợp chúng ta gọi thủ tục đó với cùng một biến 2 lần, ví dụ incr(i, j). Nếu biến i được khởi tạo giá trị 0, vậy giá trị của i sẽ là bao nhiêu sau khi gọi thủ tục này trong 2 trường hợp sau: - Lời gọi tham chiếu - Phương pháp sao chép-phục hồi được sử dụng.**

* Trường hợp 1: Lời gọi tham chiếu: Mỗi con trỏ đều trỏ tới i, nếu i tăng 1 đơn vị 2 lần => i = 2.
* Trường hợp 2: Phương pháp sao chép – phục hồi được sử dụng: Sao lưu không làm tăng giá trị => i = 1.

**Câu 3:** **Một kết nối socket cần 4 thông tin nào? Tại sao phải cần đủ 4 thông tin đó?**

Một kết nối socket cần có 4 thông tin:

* Địa chỉ IP của server.
* Port của server.
* Địa chỉ IP của client.
* Port của client.

Cần có đủ 4 thông tin này để kết nối socket có thể gửi và nhận dữ liệu chính xác.

**Câu 4:** **Tại sao giao thức yêu cầu-trả lời (request-reply) lại được coi là đồng bộ và tin cậy?**

Vì chỉ khi nhận được request từ 1 bên, phía bên kia mới gửi về thông điệp trả lời. Sau khi bên gửi request nhận được trả lời, máy tính sẽ gửi thông điệp xác nhận đã nhận được trả lời cho bên trả lời.

**Câu 5: Hai vấn đề chính của giao thức RPC:**

* Trễ giữa trong mỗi lần gọi thủ tục từ xa. Khi gọi càng nhiều, thời gian trể càng tăng có thể gây ra nghẽn.
* Việc bảo mật các tham số gửi đi và kết quả trả về. Nếu sử dụng mã hóa để bảo mật thì thời gian trễ tăng hơn nữa, càng dễ gây ra tắc nghẽn.

**Câu 6:** **Vấn đề đối với truyền tham biến trong RPC là gì? Còn đồi với truyền tham chiếu? Giải pháp đưa ra là gì?**

* Vấn đề đối với truyền tham chiếu: Một biến có thể bị tham chiếu đến nhiều lần, dẫn đến giá trị của biến bị thay đổi, làm sai kết quả của các lần gọi tiếp theo. Giải pháp: Phương pháp sao chép – phục hồi.
* Vấn đề đối với truyền tham biến: Giữa các máy phải có sự đồng nhất về việc biểu diễn dữ liệu và các kiểu dữ liệu.

**Câu 7: So sánh RPC và RMI:**

* Giống nhau: Dùng để gọi hàm/thủ tục từ xa.
* Khác nhau:
* RPC hướng thủ tục. Triển khai được cho hai hệ thống khác nhau, chỉ cần chương trình hai phía client và server viết cùng ngôn ngữ.
* RMI hướng đối tượng, triển khai bằng ngôn ngữ Java và được tích hợp sẵn trong ngôn ngữ Java. Cho phép tham chiếu đến đối tượng cũng như phương thức của đối tượng từ xa.

Nhược điểm của RMI so với RPC: Cần cài đặt môi trường lập trình Java.

**Câu hỏi 8: Hàm listen được sử dụng bởi TCP server có tham số là backlog. Giải thích ý nghĩa tham số đó.**

**+** backlog là số kết nối cho phép của hàng đợi. Các yêu cầu connect sẽ được lưu trong hàng đợi cho đến khi được accept.

**Câu 9:** **Trong trao đổi thông tin hướng dòng, những cơ chế thực thi QoS được thực hiện ở tầng nào? Giải thích. Trình bày một số cơ chế thực thi QoS để chứng minh điều đó.**

Trong trao đổi thông tin hướng dòng, những cơ chế thực thi QoS được thực hiện ở cả 2 tầng: Tầng liên kết dữ liệu và tầng mạng.

* Ở tầng mạng, ta sử dụng 3 bit đầu trong trường *loại dịch vụ* (Service Type). 3 bit này được gọi là trường IP Precedence. Mặc định trường này là 000 nghĩa là gói tin được truyền theo kiểu Best Effort. 7 giá trị còn lại được dùng để phân chia lưu lượng thành 7 lớp dịch vụ có thứ tự ưu tiên tăng dần.
* Ở tầng liên kết dữ liệu, ta có thể phân lưu lượng dựa vào việc chèn thêm các thẻ định danh VLAN (tag) theo giao thức 802.1Q/p. Mỗi tag gồm   
  4 byte trong đó có trường CoS gồm 3 bit có chức năng tương tự trường   
  IP Precedence.

**Chương 5: Naming**

**Câu 1:** **Tại sao không thể lấy địa chỉ của Access Point để sử dụng như địa chỉ của thực thể?**

Không thể lấy địa chỉ của Access Point để sử dụng như địa chỉ của thực thể vì thông tin có khả năng di chuyển từ Access này sang Access point khác.  
**Câu 2 : Với việc sử dụng Định Danh, các vấn đề gì cần phải xem xét?**

Với việc sử dụng Định Danh, các vấn đề cần phải xem xét :  
+ Không gian định danh hữu hạn-> Cạn kiệt định danh   
Giải pháp : Sử dụng lạ -> Có khả năng bị trùng   
**Câu 3:** **Xét một thực thể di chuyển từ vị trí A sang vị trí B. Trong quá trình di chuyển thực thể đó có đi qua các nút trung gian nhưng chỉ dừng lại ở đó khoảng thời gian ngắn. Khi đến B, thực thể đó dừng lại. Chúng ta biết rằng việc thay đổi địa chỉ trong một dịch vụ tổ chức vị trí phân cấp (hierarchical location service) là rất mất thời gian để hoàn thành, vì vậy cần tránh làm việc này khi thực thể tạm dừng ở các nút trung gian. Hãy đề xuất một mô hình kết hợp cả dịch vụ tổ chức vị trí phân cấp và cơ chế chuyển tiếp con trỏ (forwarding pointers) để có thể xác định được vị trí của thực thể khi nó ở các nút trung gian.**

- Khi thông tin  dịch chuyển trung gian A và B, để lại con trỏ khi đến B ghi thêm địa chỉ mới đó-> Mô hình phân cấp.  
           - Chuỗi con trỏ trung gian bị xóa đi, địa chỉ ở A cũng bị xóa.  
**Câu 4:** **Khi chúng ta thêm 1 node mới vào hệ thống Chord, chúng ta có cần phải cập nhật toàn bộ các bảng finger?**

Không cần phải cập nhật toàn bộ các bảng finger, chỉ cần cập nhật cho nút sau và nút trước vì trong hệ thống Chrod, mỗi nút sẽ quản lý khóa của nút trước đó.  
**Câu 5:**   
**Ưu điểm của phân giải có đệ quy so với phân giải không đệ quy:**- Phân giải tên có đệ quy xử lý nhiều ở bên server, phân giải không đệ quy xử lý nhiều bên client. Khi client ở xa, phân giải không đệ quy phải gửi đi gửi lại dẫn tới độ trễ cao.  
- Phân giả tên có đệ quy so các name server tổ chức lưu trữ bộ đêm, server lưu trữ kết quả , có thể trả lời luôn và nhanh cho các client   
- Tốn ít chi phí thông tin.

**Bài tập chương 6 Synchronization**

**Câu 1:** **Trình bày 1 ví dụ để mô phỏng vấn đề gặp phải khi các máy tính/tiến trình hoạt động trong hệ thống phân tán mà không có đồng hộ vật lý dùng chung.**

Vấn đề có thể gặp phải khi đó chẳng hạn là 1 máy tính nhận được 1 gói tin có thời gian gửi là sau thời gian hiện tại của máy đó.

**Câu 2: Tại sao Lamport lại đề xuất sử dụng đồng hồ logic thay cho đồng hồ vật lý trong hệ phân tán?**

Vì ta chỉ cần biết thứ tự của các yêu cầu hay gói tin chứ không cần biết thời gian cụ thể nhận được yêu cầu hay gói tin ấy.

**Câu 3:** **Đặc điểm gì của mạng không dây (wireless network) khiến cho thiết kế các giải thuật đồng bộ khác các kiểu mạng khác?**

Đặc điểm của mạng không dây là cần tối ưu việc nhận và gửi thông điệp.

**Câu 4:** **Giải thuật Lamport được đưa ra để thực hiện loại trừ lẫn nhau (mutual exclusion). Giải thuật được mô tả như sau:**

**Hệ thống có n tiến trình: P1, P2, ... Pn. Có 1 tài nguyên chia sẻ dùng chung gọi là SR (Shared Resource). Mỗi tiến trình sẽ lưu trữ một hàng đợi queuei để lưu các yêu cầu của các tiến trình khác khi chưa được thực hiện.**

**Khi tiến trình Pi muốn truy cập vào SR, nó sẽ quảng bá 1 thông điệp REQUEST(tsi,i) cho tất cả các tiến trình khác, đồng thời lưu trữ thông điệp đó vào hàng đợi của mình (queuei) trong đó tsi là timestamp của yêu cầu.**

**Khi 1 tiến trình Pj nhận được yêu cầu REQUEST(tsi,i) từ tiến trình Pi thì nó đưa yêu cầu đó vào hàng đợi của mình (queuej) và gửi trả lại cho Pi thông điệp REPLY.**

**Tiến trình Pi sẽ tự cho phép mình sử dụng SR khi nó kiểm tra thấy yêu cầu của nó nằm ở đầu hàng đợi queueivà các yêu cầu khác đều có timestamp lớn hơn yêu cầu của chính nó.**

**Tiến trình Pi, khi không dùng SR nữa sẽ xóa yêu cầu của nó khỏi hàng đợi và quảng bá thông điệp RELEASE cho tất cả các tiến trình khác.**

**Khi tiến trình Pj nhận được thông điệp RELEASE từ Pi thì nó sẽ xóa yêu cầu của Pi trong hàng đợi của nó.**

**Câu hỏi:**

**a) Để thực hiện thành công 1 tiến trình vào sử dụng SR, hệ thống cần tổng cộng bao nhiêu thông điệp?**

**b) Có 1 cách cải thiện thuật toán trên như sau: sau khi Pj gửi yêu cầu REQUEST cho các tiến trình khác thì nhận được thông điệp REQUEST từ Pi, nếu nó nhận thấy rằng timestamp của REQUEST nó vừa gửi lớn hơn timestamp của REQUEST của Pi, nó sẽ không gửi thông điệp REPLY cho Pi nữa.**

**Cải thiện trên có đúng hay không? Và với cải thiện này thì tổng số thông điệp cần để thực hiện thành công 1 tiến trình vào sử dụng SR là bao nhiêu? Giải thích.**

1. Hệ thống cần: n-1 thông điệp request, n-1 thông điệp reply, n-1 thông điệp release. Tổng cộng là 3n-3 thông điệp.
2. Trường hợp tốt nhất: 2n-2 thông điệp: Khi không cần phải gửi bất cứ thông điệp Reply nào. Trường hợp xấu nhất: 3n-3 thông điệp: Vẫn phải gửi các thông điệp Reply.

**Câu 5:** **Giải thuật Szymanski được thiết kế để thực hiện loại trừ lẫn nhau. Ý tưởng của giải thuật đó là xây dựng một *phòng chờ* (waiting room) và có *đường ra* và *đường vào*, tương ứng với *cổng ra* và *cổng vào*. Ban đầu *cổng vào* sẽ được mở, *cổng ra* sẽ đóng. Nếu có một nhóm các tiến trình cùng yêu cầu muốn được sử dụng tài nguyên chung SR (shared resource) thì các tiến trình đó sẽ được xếp hàng ở *cổng vào* và lần lượt vào *phòng chờ*. Khi tất cả đã vào *phòng chờ* rồi thì tiến trình cuối cùng vào phòng sẽ đóng *cổng vào* và mở *cổng ra*. Sau đó các tiến trình sẽ lần lượt được sử dụng tài nguyên chung. Tiến trình cuối cùng sử dụng tài nguyên sẽ đóng *cổng ra* và mở lại *cổng vào*.**

**Mỗi tiến trình Pi sẽ có 1 biến flagi, chỉ tiến trình Pi mới có quyền ghi, còn các tiến trình Pj (j ≠ i) thì chỉ đọc được. Trạng thái mở hay đóng cổng sẽ được xác định bằng việc đọc giá trị flag của các tiến trình khác. Mã giả của thuật toán đối với tiến trình *i* được viết như sau:**

***#Thực hiện vào phòng đợi***

**flag[i] ← 1**

**await(all flag[1..N]∈{0,1,2})**

**flag[i] ← 3**

**ifany flag[1..N]=1:**

**flag[i] ← 2**

**await(any flag[1..N]=4)**

**flag[i] ← 4**

**await(all flag[1..i-1]∈{0,1})**

***#Sử dụng tài nguyên***

***#...***

***#Thực hiện giải phóng tài nguyên***

**await(all flag[i+1..N]∈{0,1,4})**

**flag[i] ← 0**

**Giải thích ký pháp trong thuật toán:**

***await(điều\_kiện):* chờ đến khi thỏa mãn điều\_kiện**

***all*: tất cả**

***any*: có bất kỳ 1 cái nào**

**Câu hỏi:**

**flag[i] sẽ có 5 giá trị trạng thái từ 0-4. Dựa vào giải thuật trên, 5 giá trị đó mang ý nghĩa tương ứng nào sau đây (có giải thích):**

**- Chờ tiến trình khác vào *phòng chờ***

**- *Cổng vào* được đóng**

**- Tiến trình *i*đang ở ngoài *phòng chờ***

**- Rời phòng, mở lại *cổng vào* nếu không còn ai trong *phòng chờ***

**- Đứng đợi trong *phòng chờ***

Ý nghĩa của 5 giá trị:

* 0: Rời phòng, mở lại cổng vào nếu không còn ai trong phòng chờ.
* 1: Tiến trình i đang ở ngoài phòng chờ.
* 2: Chờ tiến trình khác vào phòng chờ.
* 3: Đứng đợi trong phòng chờ.
* 4: Cổng vào được đóng.

Giải thích:

* 0: Đợi tất cả các tiến trình khác đều rời phòng(0) hoặc ở ngoài phòng chờ(1) hoặc cổng vào đóng(4) thì tiến trình i cũng rời phòng và mở lại cổng vào.
* 1: Trạng thái khởi đầu, các tiến trình đều đang ở ngoài phòng chờ.
* 2: Nếu có bất kì tiến trình nào đang ở ngoài phòng chờ(1) thì tiến trình i báo chờ tiến trình khác vào phòng chờ.
* 3: Đợi tất cả các tiến trình khác đều rời phòng(0) hoặc ở ngoài phòng chờ(1) hoặc đang chờ tiến trình khác vào phòng chờ(2) thì tức là tiến trình i đang đứng đợi trong phòng chờ.
* 4: Đợi khi có bất kì tiến trình nào báo cổng vào được đóng(4) thì tiến trình i cũng báo cổng vào được đóng.

Bài tập chương sao lưu và thống nhất dữ liệu

Cấu 1:

Các lý do phải sao lưu dữ liệu:

* Liên quan đến độ tin cậy (tính sẵn sàng).
* Liên quan đến vấn đề hiệu năng.
* Liên quan đến khả năng co dãn.

Câu 3:

Conit dùng để định nghĩa độ không thống nhất, từu đó dung để đánh giá độ thống nhất. Conit được coi như là tập đơn vị dữ liệu của kho dữ liệu phân tán.

* Nếu đặt kích thước Conit quá lớn: Các bản sao sẽ sớm bị rơi vào trạng thái không thống nhất.
* Nếu đặt kích thước Conit quá nhỏ: Số lượng Conit nhiều => quản lý phức tạp.

Câu 4 :

Thống nhất nhân quả có tính thống nhất yếu hơn thống nhất tuần tự. Vì:

* Thống nhất tuần tự: Nếu thứ tự các thao tác cục bộ của một tiến trình khôngthay đổi trong thứ tự thực hiện chung trên kho dữ liệu => Kết quả luôn như nhau => Tính thống nhất cao.
* Thống nhất nhân quả: Các sự kiện có quan hệ nhân quả đảm bảo thứ tự, các thao tác ghi song song hoạc tương tranh khác không cần => Tính thống nhất yếu.

Câu 5 :

Vấn đề của mô hình Eventual Consistency:

* Nếu tiến trình khách luôn luôn truy cập vào 1 bản sao => Thống nhất luôn được đảm bảo.
* Nếu tiến trình khách truy cập vào các bản sao khác nhau => Vấn đề.

Mô hình thống nhất hướng Client: Cung ứng đảm bảo thống nhất cho các truy cập của một Client đơn vào kho dữ liệu (không đảm bảo thống nhất cho các truy cập cạnh tranh của các tiến trình khác).