Bài tập chương 3

Câu 1 (1): a) Định nghĩa "tên" (name) trong hệ thống máy tính phân tán

Trong **hệ thống máy tính phân tán**, **"tên" (name)** là **một chuỗi ký tự dùng để tham chiếu đến một thực thể (entity)** như tiến trình, dịch vụ, tập tin, máy chủ, hoặc điểm truy cập mạng.

Tên đóng vai trò **trung gian giúp con người hoặc hệ thống xác định, truy xuất và tương tác với thực thể** tương ứng trong hệ thống. Mỗi tên phải tuân theo một **cú pháp và quy tắc đặt tên** do hệ thống quy định.

- Tập hợp các tên tham chiếu đến thực thể được gọi là **không gian tên (namespace)**.
- Xét dưới góc độ hình học, không gian tên được mô hình hóa như một **đồ thị có hướng có cấu trúc**, trong đó mỗi nút lá biểu diễn một thực thể, còn các nút thư mục có thể chứa nhiều thực thể con.

b) Phân biệt giữa tên thân thiện (friendly name) và định danh (identifier)

Tiêu chí	Tên thân thiện (Friendly Name)	Định danh (Identifier)
Định nghĩa	Là tên được đặt theo cách dễ hiểu, gần gũi với con người	Là tên đặc biệt đảm bảo duy nhất trong hệ thống
Mục đích sử dụng	Giúp người dùng dễ nhớ, dễ tra cứu, dễ thao tác	Dùng cho máy tính định danh thực thể duy nhất, phục vụ truy xuất và xử lý
Tính duy nhất	Không bắt buộc phải duy nhất	**Bắt buộc duy nhất toàn cục** trong hệ thống
Cách tạo ra	Do con người đặt theo ý nghĩa (vd: "may_chu_1", "sinhvien123")	Thường được hệ thống sinh ra tự động (vd: mã hash, UUID)
Tính ổn định theo thời gian	Có thể thay đổi (vd: đổi tên thư mục, nickname người dùng)	Ổn định, không thay đổi trong suốt vòng đời thực thể
Khả năng mô tả vị trí	Không nhất thiết mô tả vị trí	Có thể gắn liền với vị trí (nếu được dùng như địa chỉ mạng)

Câu 3 (2):

Giải pháp đơn giản cho tìm kiếm thực thể dùng tên phi cấu trúc

Tên phi cấu trúc (tên phẳng) là chuỗi bit ngẫu nhiên không phản ánh thông tin gì về thực thể. Do đó, để tìm được địa chỉ của thực thể, cần có **giải pháp tìm kiếm**, và **giải pháp đơn giản** là một trong những cách tiếp cận đầu tiên và dễ thực hiện nhất.

1. Nguyên lý hoạt động

a) Truyền thông quảng bá (Broadcast)

- Máy khách gửi một thông điệp chứa **định danh thực thể cần tìm** đến **toàn bộ các máy trong hệ thống**.
- Mọi máy trong mạng đều **lắng nghe và xử lý yêu cầu**.
- Nếu máy nào đang lưu trữ thực thể với định danh phù hợp, **nó sẽ phản hồi địa chỉ điểm truy nhập** cho máy khách.

Ví dụ minh họa:
Giao thức **ARP** sử dụng broadcast để tìm địa chỉ MAC khi biết địa chỉ IP. Khung dữ liệu được gửi với địa chỉ đích là `FFFFFFFFFFFF` (địa chỉ broadcast vật lý), mọi máy trong mạng sẽ nhận được yêu cầu, máy nào có IP trùng khớp sẽ phản hồi.

b) Truyền thông theo nhóm (Multicast)

- Máy khách gửi yêu cầu **đến một nhóm máy đã định nghĩa trước**, không phải toàn bộ mạng.
- Các máy trong nhóm **kiểm tra định danh thực thể** và **phản hồi nếu trùng khớp**.

Chi tiết kỹ thuật:

- Theo chuẩn **Ethernet 802.3**, nếu **bit thấp nhất của byte cao nhất bằng 1**, thì đó là địa chỉ vật lý multicast.
- Tầng mạng sử dụng **địa chỉ IP lớp D (224.0.0.0 – 239.255.255.255)** để định danh nhóm.

2. Ưu – Nhược điểm và Phạm vi áp dụng

Phương pháp	Ưu điểm	Nhược điểm	Phạm vi áp dụng
Broadcast	- Đơn giản, dễ triển khai - Không cần kho dữ liệu	- Lãng phí băng thông - Gây nhiễu hoạt động của máy khác - Không hiệu quả với mạng lớn	- **Mạng cục bộ nhỏ (LAN)**

Multicast	- Giảm tải so với broadcast - Tập trung vào nhóm máy liên quan	- Cần tổ chức và duy trì nhóm multicast - Không phù hợp nếu nhóm không rõ ràng hoặc mạng rộng và phân mảnh	- **Mạng vừa hoặc mạng có phân nhóm rõ ràng**

Câu 4: (3)

a) Giải thích cơ chế con trỏ chuyển tiếp (forwarding pointer) trong tìm kiếm thực thể di động

Cơ chế **con trỏ chuyển tiếp (forwarding pointer)** được dùng để theo dõi **thực thể di động** trong hệ thống phân tán khi thực thể thay đổi vị trí qua các nút trong mạng.

Nguyên lý hoạt động:

- Khi một thực thể **di chuyển** từ nút A sang nút B, **tại nút A sẽ lưu lại một con trỏ chuyển tiếp** (forwarding pointer) trỏ tới nút B – vị trí mới của thực thể.
- Khi máy khách cần **truy cập thực thể**, nó gửi yêu cầu đến vị trí cũ (nút A).
    - Nếu thực thể còn ở A → xử lý tại chỗ.
    - Nếu đã chuyển sang B → A **chuyển tiếp yêu cầu** đến B.
- Quá trình chuyển tiếp có thể **tiếp tục** nếu thực thể đã di chuyển nhiều lần (A → B → C → ...).
- Sau cùng, thực thể được định vị, và **kết quả truy vấn được trả về** cho máy khách (trực tiếp hoặc gián tiếp).

Biến thể mở rộng – định hướng lại con trỏ chuyển tiếp:

- Sau lần truy cập đầu tiên, máy chủ cũ (ví dụ: A) có thể cung cấp **địa chỉ mới (ví dụ: B)** cho máy khách.
- Máy khách **cập nhật cấu hình**, và **những lần truy cập sau sẽ gọi thẳng đến B**, không cần qua A.
- Điều này giúp **tăng hiệu năng**, nhưng **mất tính trong suốt** và có thể ảnh hưởng tới **bảo mật**.

b) Phân tích tác động đến tính trong suốt, hiệu năng và tính sẵn sàng khi chuỗi con trỏ quá dài hoặc có nút chuyển tiếp lỗi

Khía cạnh	Phân tích chi tiết
Tính trong suốt	- Khi con trỏ hoạt động tốt, **quá trình tìm kiếm hoàn toàn trong suốt đối với máy khách** – khách chỉ thấy như đang tương tác với một máy chủ duy nhất. - Tuy nhiên, nếu sử dụng phương pháp định hướng lại, máy khách **biết được vị trí thực tế của thực thể**, **mất đi tính trong suốt vị trí**.
Hiệu năng	- Nếu chuỗi con trỏ ngắn: hiệu năng tốt. - Nếu thực thể **di chuyển nhiều lần**, chuỗi con trỏ sẽ dài → **tăng độ trễ** vì yêu cầu phải đi qua nhiều nút trung gian. - Càng nhiều con trỏ → **tăng độ phức tạp xử lý và chi phí truyền thông**.
Tính sẵn sàng	- Mỗi nút trong chuỗi là một **điểm phụ thuộc**. Nếu một nút **gặp lỗi hoặc không phản hồi**, toàn bộ chuỗi con trỏ sẽ **bị đứt**, không thể định vị được thực thể. - Điều này **làm giảm độ sẵn sàng** của hệ thống. - Giải pháp: **cài đặt một máy chủ dự phòng**, luôn giữ thông tin vị trí hiện tại của thực thể, chỉ dùng đến khi con trỏ bị lỗi.

Câu 5: (4)

1. Nguyên lý hoạt động của giải pháp dựa trên nguồn gốc (Home-based Origin Approach)

Giải pháp này được sử dụng để quản lý việc **tìm kiếm thực thể di động** trong hệ thống có **quy mô lớn**, khi thực thể thường xuyên thay đổi vị trí mạng (IP).

Cơ chế chính:

- Mỗi **thực thể di động** sẽ được **đăng ký địa chỉ IP cố định** với một **máy chủ gốc (home agent)**.

- Khi thực thể **di chuyển sang mạng mới**, nó sẽ được **cấp một địa chỉ IP tạm thời** bởi **máy chủ tạm thời (foreign agent)** thuộc mạng mới đó.

- Địa chỉ IP tạm thời này sẽ được gửi về **máy chủ gốc** để **cập nhật cơ sở dữ liệu** vị trí.

- Mọi **liên lạc đến thực thể di động** thông qua IP gốc sẽ được **chuyển tiếp qua home agent** → home agent tạo **đường hầm (tunnel)** để chuyển tiếp dữ liệu đến địa chỉ hiện hành (tức là foreign agent).

- Khi bên gửi đã nhận được thông tin định tuyến mới, các **gói tin tiếp theo** có thể được **gửi trực tiếp đến địa chỉ tạm thời**, **không cần qua máy chủ gốc**.

2. Vai trò của các thành phần

Máy chủ gốc (Home Agent):

- Là nơi **đăng ký địa chỉ IP cố định** của thực thể di động.

- **Quản lý vị trí hiện tại** của thực thể thông qua cơ sở dữ liệu cập nhật.

- **Chuyển tiếp gói tin** đến địa chỉ mới nếu thực thể đã di chuyển.

- **Tạo đường hầm (tunnel)** để chuyển dữ liệu đến địa chỉ tạm thời.

Máy chủ tạm thời (Foreign Agent):

- **Cấp phát địa chỉ IP tạm thời** cho thực thể khi nó gia nhập mạng mới.

- **Tiếp nhận dữ liệu chuyển tiếp từ home agent**, rồi chuyển đến thực thể.

- Có thể **trả lời bên gửi**, giúp rút ngắn hành trình gói tin về sau.

3. Ưu – nhược điểm

Ưu điểm:

- **Hỗ trợ tốt cho thực thể di động thường xuyên thay đổi vị trí.**

- **Bảo toàn địa chỉ IP gốc**, nên **không gián đoạn kết nối** ứng dụng (tầng ứng dụng thấy như không có gì thay đổi).

- Khi đã nhận biết địa chỉ mới, bên gửi có thể **giao tiếp trực tiếp**, **giảm độ trễ** cho các gói tin sau.

Nhược điểm:

Vấn đề	Mô tả
Độ trễ ban đầu cao	Gói tin đầu tiên phải **đi qua home agent**, rồi mới tới foreign agent.
Phụ thuộc vào home agent	Nếu **máy chủ gốc bị lỗi**, hệ thống **mất khả năng chuyển tiếp** hoặc cập nhật vị trí mới.
Chi phí duy trì	Home agent phải **duy trì cơ sở dữ liệu cập nhật liên tục**, gây tốn tài nguyên.

Câu 7: (5)

1. Khả năng mở rộng (Scalability)

Tiêu chí	Phân cấp (Hierarchical Naming)	DHT (Chord)
Cơ chế tổ chức	Cây phân cấp logic gồm các miền (gốc → trung gian → lá)	Vòng định danh m-bit, khóa được phân bố ngẫu nhiên qua hàm băm
Tăng quy mô hệ thống	Cần thêm nhiều nút và mở rộng cây, nhưng nút gốc là điểm nghẽn	Dễ mở rộng: mỗi nút chỉ cần bảng định tuyến O(logN), không nút trung tâm
Hiệu suất khi số nút lớn	Giảm dần nếu nút gốc hoặc nút trung gian bị quá tải	Duy trì O(logN) bước truy vấn ngay cả khi có hàng triệu nút
Tổng kết	Thích hợp cho hệ thống nhỏ hoặc trung bình Không phù hợp khi quá lớn do nút gốc quá tải	Rất mở rộng tốt, phù hợp cho hệ thống phân tán quy mô rất lớn

2. Khai thác tính cục bộ (Locality Awareness)

Tiêu chí	Phân cấp (Hierarchical Naming)	DHT (Chord)
Cơ chế hoạt động	Ưu tiên tìm kiếm tại nút lá gần máy khách trước khi chuyển lên nút cha	Hàm băm phân bố khóa ngẫu nhiên nên không gắn với vị trí địa lý
Tính cục bộ	Tốt: tìm kiếm thường chỉ giới hạn trong khu vực gần	Kém: không khai thác được tính gần (locality)
Tình huống thực tế	Người dùng trong cùng một vùng thường tra cứu thực thể gần → hiệu quả cao	Truy vấn có thể phải đi qua các nút rất xa về mặt mạng vật lý
Tổng kết	Khai thác tốt tính cục bộ, giảm độ trễ truy vấn	Không khai thác locality, độ trễ có thể cao nếu nút ngẫu nhiên phân bố

3. Chi phí cập nhật khi thực thể di động (Update Cost when entity moves)

Tiêu chí	Phân cấp (Hierarchical Naming)	DHT (Chord)
Thực thể di chuyển vị trí	Cần cập nhật địa chỉ trong nút lá mới và chuỗi các nút cha đến nút gốc nếu địa chỉ thay đổi	Chỉ cần cập nhật lại khóa thực thể ở nút mới được xác định bằng `succ(hash(k))`
Chi phí cập nhật	Tốn kém, nhất là nếu truy cập xuyên miền: phải duyệt từ lá đến gốc hoặc ngược lại để cập nhật con trỏ	Ít tốn kém, chỉ liên quan đến một số nút gần `succ(k)` và bảng định tuyến cục bộ
Độ nhạy với di động	Không linh hoạt với thực thể di động liên tục – phải cập nhật nhiều nơi	Phù hợp hơn với thực thể di chuyển thường xuyên

Tổng kết	Chi phí cập nhật cao nếu thực thể di động hoặc nhân bản ở nhiều miền	Cập nhật đơn giản, chỉ cần cập nhật tại nút chịu trách nhiệm khóa đó

Câu 12: (6)

a. Mô tả chi tiết quá trình phân giải tên (name resolution)

Phân giải tên là quá trình **truy xuất định danh** của một thực thể trong **không gian tên có cấu trúc**, dựa trên **đường dẫn tên** được cung cấp.

Quy trình:

1. **Bắt đầu từ một nút gốc xác định** (ví dụ: `/` hoặc giá trị từ một biến môi trường như `HOME`).
2. **Truy vấn bảng thư mục** tại nút hiện tại để tìm nút con tương ứng với thành phần tiếp theo trong đường dẫn.
3. **Lặp lại bước 2** cho đến khi phân giải hết các thành phần trong đường dẫn.
4. **Kết quả cuối cùng** là định danh (identifier) của **nút đích**, thường là một nút lá (file) hoặc thư mục.

Ví dụ:

Phân giải tên `/home/steen/mbox`:

- Bắt đầu từ nút `/` (gốc).
- Truy cập bảng thư mục của `/` để tìm nút `home`.
- Truy cập bảng thư mục của `home` để tìm nút `steen`.
- Truy cập bảng thư mục của `steen` để tìm nút `mbox`.
- Trả về định danh của `mbox`.

b. Khái niệm cơ chế đóng (closed namespace) và tác động của nó

Định nghĩa:

Cơ chế đóng là yêu cầu của hệ thống phân giải tên rằng **quá trình phân giải phải biết chính xác nút bắt đầu (điểm gốc)** của không gian tên.

Tác động:

- **Tăng độ chính xác:** vì hệ thống cần có "điểm vào" rõ ràng để bắt đầu phân giải.
- **Giảm tính linh hoạt và dễ hiểu:** nếu không biết ý nghĩa hoặc bối cảnh của tên, **không thể phân giải chính xác**.
    - Ví dụ: chuỗi `"84986677028"` chỉ có ý nghĩa nếu bạn **ngầm hiểu** nó là một **số điện thoại di động**, từ đó biết bắt đầu phân giải từ cơ sở dữ liệu số điện thoại.
- **Yêu cầu có thông tin bổ sung:** như biến môi trường hoặc tham chiếu ngữ cảnh để định nghĩa đúng **điểm gốc** của đường dẫn.

c. Ví dụ minh họa với biến môi trường `HOME` trong Unix

Trong hệ điều hành **Unix**, mỗi người dùng có một **thư mục chính riêng**, ví dụ: `/home/huan`.

Cách phân giải:

- Biến môi trường `HOME` chứa đường dẫn đến thư mục gốc của người dùng (ví dụ: `HOME=/home/huan`).
- Khi người dùng nhập đường dẫn như `~/Documents`:
    - Hệ thống **ngầm thay thế `~` bằng giá trị của `HOME`**, tức là `/home/huan`.
    - Sau đó, thực hiện **phân giải tên** theo trình tự:
        1. Truy cập bảng thư mục của `/` → tìm `home`.
        2. Truy cập bảng thư mục `home` → tìm `huan`.
        3. Truy cập bảng thư mục `huan` → tìm `Documents`.

Liên hệ với cơ chế đóng:

- `HOME` chính là **cơ chế đóng**, xác định **nút gốc** cho quá trình phân giải tên.
- Nếu không có giá trị `HOME`, hệ thống không biết nên bắt đầu từ đâu để tìm `Documents`.

Câu 19: (7)

a) Mô tả giải pháp ngang hàng (P2P) dùng bảng chỉ mục phân tán để hỗ trợ tìm kiếm theo thuộc tính

Mục tiêu của giải pháp:

Hỗ trợ **tìm kiếm các thực thể dựa trên mô tả thuộc tính** (ví dụ như `{tên = "main server", quốc gia = "VN"}`) **trên hệ thống phân tán ngang hàng (P2P)**, nơi mà mỗi nút có thể lưu trữ dữ liệu và không tồn tại máy chủ trung tâm.

Nguyên lý hoạt động:

1. **Mỗi thực thể được mô tả bởi tập các cặp {thuộc tính, giá trị}**.

2. **Xây dựng hệ thống chỉ mục phân tán**, trong đó:

    - **Mỗi thuộc tính** được gán cho một hoặc nhiều **máy chủ chỉ mục** chuyên trách lưu thông tin về các thực thể có giá trị tương ứng với thuộc tính đó.

    - Ví dụ: Máy chủ cho thuộc tính `Country` (C) lưu tập `{(E1, "VN"), (E2, "US"), ...}`.

Khi máy khách gửi truy vấn, ví dụ:
`search((C=VN)(O=Bộ thông tin và truyền thông)(CN=Main Server))`

3.
    - Truy vấn sẽ được **tách thành từng cặp {thuộc tính, giá trị}**, sau đó **gửi đến các máy chủ chỉ mục tương ứng** với từng thuộc tính.

4. **Mỗi máy chủ chỉ mục sẽ trả về danh sách các thực thể** thỏa mãn cặp thuộc tính-giá trị mà nó quản lý.

5. Máy khách sẽ **tính giao nhau (intersection)** của các tập thực thể này để tìm ra **thực thể phù hợp với toàn bộ truy vấn**.

Ví dụ minh họa:

Truy vấn: `(C=VN) ∧ (O=Thông tin) ∧ (CN=Main Server)`

- Gửi đến 3 máy chủ chỉ mục tương ứng với thuộc tính `C`, `O`, `CN`.

- Máy chủ `C` trả về `{E1, E2, E3}`
- Máy chủ `O` trả về `{E1, E4}`
- Máy chủ `CN` trả về `{E1, E5}`

⇒ Máy khách tính giao nhau: `{E1}` là thực thể thỏa mãn.

b) Phân tích ba hạn chế cơ bản của cách tiếp cận này

1. Số lượng máy chủ chỉ mục tăng theo số thuộc tính

- **Vấn đề:** Mỗi thuộc tính yêu cầu ít nhất một máy chủ chỉ mục để lưu thông tin `{thực thể, giá trị}`.
- Khi có hàng trăm thuộc tính thì cần **hàng trăm máy chủ chỉ mục**, gây **tăng chi phí lưu trữ, quản lý và bảo trì hệ thống**.
- Đồng thời, **mỗi truy vấn có k thuộc tính sẽ tạo ra k lần truy cập mạng**.

Hệ quả: Gây **quá tải truyền thông**, giảm hiệu suất nếu truy vấn phức tạp với nhiều thuộc tính.

2. Kích thước tập kết quả (result set) có thể rất lớn

- Với các thuộc tính phổ biến (ví dụ `Country = VN`), có thể có **hàng triệu thực thể** thỏa mãn.
- Khi máy chỉ mục trả về kết quả quá lớn, máy khách phải xử lý một lượng lớn dữ liệu để lọc kết quả.

Hệ quả: Gây **quá tải bộ nhớ, CPU trên máy khách**, làm chậm quá trình tìm kiếm.

3. Không hỗ trợ truy vấn theo phạm vi (range query)

- Cách ánh xạ mỗi giá trị thuộc tính đến một máy chủ chỉ mục chỉ **phù hợp với truy vấn so sánh bằng (=)**.

Không thể xử lý truy vấn phạm vi, ví dụ:
`(Số lượng nhân viên >= 1000 ∧ <= 2000)`

- **Nguyên nhân:** Không có cách nào biểu diễn phạm vi này bằng các ánh xạ chính xác tới một hoặc vài máy chỉ mục.
  Dữ liệu bị phân tán không liên tiếp, **không thể xử lý so sánh lớn hơn/nhỏ hơn một cách hiệu quả**.