Câu hỏi ôn tập chương 5 và chương 6

Câu 1

a) Trình bày sự khác nhau giữa **tên phi cấu trúc (flat name)**, **tên có cấu trúc (structured name)** và **tên dựa trên thuộc tính (attribute-based name)**.
b) Cho ví dụ minh họa cho từng loại tên trong một hệ thống phân tán.

a) Phân biệt các loại tên:

- **Tên phi cấu trúc (Flat name):**
  Là tên thuần túy, không mang ý nghĩa nào khác ngoài việc phân biệt các thực thể. Thường được biểu diễn bằng chuỗi bit hoặc ký tự ngẫu nhiên (pure name).
  Loại tên này không chứa thông tin về vị trí hay cấu trúc, dùng được trong các hệ thống phân tán quy mô lớn.
  Ví dụ: địa chỉ MAC, GUID, hoặc khóa trong hệ thống DHT.
- **Tên có cấu trúc (Structured name):**
  Là tên được tổ chức theo cấu trúc, thường ở dạng cây hoặc đồ thị (naming graph). Các nút thư mục (directory nodes) chứa bảng ánh xạ tên con sang nút đích, còn các nút lá biểu diễn thực thể.
  Ví dụ: đường dẫn /home/steen/mbox trong hệ thống file.
- **Tên dựa trên thuộc tính (Attribute-based name):**
  Tên không được xác định tường minh, mà được tìm bằng các thuộc tính đặc trưng của thực thể.
  Ví dụ: truy vấn "máy in có thuộc tính color=true" để tìm máy in màu.

b) Ví dụ minh họa:

- Flat name: Địa chỉ IP, GUID, hoặc khóa trong Chord.
- Structured name: /home/user/file.txt trong hệ thống file.
- Attribute-based name: Truy vấn thiết bị theo thuộc tính.

Câu 2

a) Mô tả cấu trúc và thuật toán phân giải khóa k trong **mạng Chord** (DHT) với không gian định danh m-bit.
b) Giải thích cách mạng Chord đạt được độ phức tạp O(logN)O(\log N)O(logN) cho mỗi phép tra cứu.
c) Nêu cơ chế chèn và loại bỏ nút, cũng như cách xử lý trường hợp nút đột ngột rời mạng.

a) Cấu trúc và thuật toán phân giải khóa k trong Chord:

Mạng Chord tổ chức các nút trong một vòng định danh có kích thước 2^m.
Mỗi nút có một định danh (ID) m-bit, và mỗi thực thể (key) cũng được băm thành một ID m-bit.
Mỗi khóa k được gán cho nút có ID nhỏ nhất **≥ k**, gọi là **successor(k)**.

Mỗi nút p duy trì **bảng ngón tay (finger table)** gồm tối đa m mục:
FTp[i] = succ(p + 2^(i-1))
→ tức là mục i trỏ đến nút đầu tiên sau p ít nhất 2^(i-1).

Khi tìm khóa k, nút p gửi yêu cầu đến nút có ID nhỏ hơn nhưng gần nhất với k trong bảng ngón tay, cho đến khi đến được successor(k).

b) Độ phức tạp O(log N):

Vì mỗi bước trong quá trình tra cứu làm giảm ít nhất một nửa khoảng cách trong không gian định danh, nên số bước trung bình cần là $\log_2(N)$ — nghĩa là độ phức tạp O(log N).

c) Cơ chế chèn, loại bỏ và xử lý rời mạng:

- **Chèn nút mới:** Nút mới tìm successor của ID mình, cập nhật lại finger table và successor/predecessor của các nút liên quan.
- **Rời nút:** successor và predecessor được nối trực tiếp lại với nhau.
- **Rời đột ngột:** các nút lân cận phát hiện lỗi và dùng danh sách kế tiếp (successor list) để khôi phục liên kết, đảm bảo vòng vẫn nhất quán.

<u>Câu 3</u>
a) Định nghĩa "không gian tên có cấu trúc" (structured namespace).
b) Phân biệt **nút thư mục** và **nút lá** trong đồ thị tên có cấu trúc, nêu vai trò của mỗi loại.

a) Khái niệm không gian tên có cấu trúc:

Không gian tên có cấu trúc (structured namespace) là một đồ thị trong đó các nút lá (leaf nodes) biểu diễn thực thể được đặt tên, còn các nút thư mục (directory nodes) chứa bảng ánh xạ từ nhãn (label) sang nút khác. Đồ thị này thường có một nút gốc duy nhất.

b) Phân biệt nút thư mục và nút lá:

- **Nút thư mục:** Là nút trung gian, có vai trò điều hướng; chứa bảng gồm các cặp (tên con, định danh nút).

- **Nút lá:** Là nút kết thúc, đại diện cho một thực thể cụ thể như file, thiết bị hoặc dịch vụ.
  → Nút thư mục giúp xác định đường đi trong cây tên; nút lá là đích cuối của phân giải.

Câu 4

Mô tả chi tiết quá trình **phân giải tên** (name resolution) cho một đường dẫn cấu trúc, từ nút bắt đầu đến khi trả về định danh của nút đích.

- Nêu khái niệm **cơ chế đóng** (closed namespace) và tác động của nó.
- Ví dụ minh họa với biến môi trường HOME trong Unix.

Quá trình phân giải tên (name resolution):

Quá trình này bắt đầu từ một nút gốc hoặc nút khởi đầu được xác định bởi **cơ chế đóng (closure mechanism**), sau đó lần lượt tra từng phần của đường dẫn.

Ví dụ: với /home/maarten/mbox, hệ thống sẽ:

1. Bắt đầu tại nút gốc /.
2. Tìm mục "home" trong bảng của nút gốc.
3. Tiếp tục tại nút "home", tìm "maarten".
4. Cuối cùng đến "mbox", là nút lá – định danh của thực thể cần tìm.

Cơ chế đóng (closed namespace):

Xác định nơi bắt đầu phân giải tên. Ví dụ:

- www.distributed-systems.net → bắt đầu ở máy chủ DNS.
- /home/maarten/mbox → bắt đầu tại NFS server.
- Biến môi trường HOME trong Unix chứa đường dẫn đến thư mục cá nhân, ví dụ HOME=/home/steen, nên khi gõ ~/mbox, nó được phân giải thành /home/steen/mbox.

Câu 5

Trong trường hợp muốn **gắn kết (mount) không gian tên** của một máy chủ từ xa vào cây tên cục bộ của máy khách:

a) Liệt kê ba thông tin cần thiết (giao thức, địa chỉ máy chủ, điểm gắn).

b) Mô tả trình tự phân giải tên khi truy cập /remote/ptit/mbox trên máy khách (ví dụ NFS).

c) Phân tích tính trong suốt (transparency) và những giới hạn về hiệu năng của phương pháp này.

a) Ba thông tin cần để mount không gian tên từ xa:

1. Giao thức truy cập: ví dụ NFS, SMB,...
2. Tên máy chủ: ví dụ flits.cs.vu.nl.
3. Điểm gắn (mount point): vị trí trong cây tên cục bộ nơi nối vào, ví dụ /remote/ptit.

b) Trình tự phân giải /remote/ptit/mbox (NFS):

- Bắt đầu từ cây tên cục bộ của máy khách, hệ thống gặp thư mục /remote là mount point.
- Tại đó, trình phân giải biết phải chuyển sang không gian tên từ xa thông qua giao thức NFS.
- Sau đó, yêu cầu được gửi đến máy chủ từ xa, nơi tiếp tục phân giải ptit/mbox như trong cây của máy chủ.
  → Kết quả cuối cùng trả về định danh của tệp /remote/ptit/mbox.

c) Tính trong suốt và giới hạn:

- **Tính trong suốt (transparency):** Người dùng không cần biết phần nào là cục bộ hay từ xa; toàn bộ cây tên hiển thị như một hệ thống duy nhất.
- **Giới hạn hiệu năng:** Truy cập mạng chậm hơn truy cập cục bộ; nếu máy chủ từ xa bị lỗi hoặc mạng ngắt, việc phân giải tên sẽ thất bại.

<u>Câu 6 (Chương 6)</u>

- **Giải thuật Cristian:** Trình bày nguyên tắc hoạt động của giải thuật đồng bộ thời gian Cristian. Giải thích cách thuật toán ước lượng độ trễ mạng và tính toán độ lệch giữa đồng hồ máy khách và máy chủ để hiệu chỉnh đồng hồ máy khách.
- **Giải thuật Berkeley:** Mô tả cách thức hoạt động của giải thuật đồng bộ Berkeley. So sánh giải thuật Berkeley với giải thuật Cristian về vai trò của máy chủ và máy khách trong quá trình đồng bộ, cũng như ưu nhược điểm của mỗi giải thuật.
- **Giải thuật trung bình và tham chiếu quảng bá:** Hãy giải thích nguyên tắc hoạt động của giải thuật đồng bộ trung bình và giải thuật đồng bộ tham chiếu quảng bá trong hệ thống phân tán. Nêu những ưu điểm và nhược điểm chính của từng giải thuật.
- Hãy giải thích vì sao việc đồng bộ hóa thời gian vật lý trong hệ thống phân tán gặp nhiều khó khăn. Đồng thời, trình bày khái niệm đồng hồ logic do Lamport đề xuất và vai trò của nó trong việc giải quyết vấn đề này.
- Trình bày cách hoạt động của thuật toán đồng hồ Lamport trong việc gán nhãn thời gian cho các sự kiện: mỗi tiến trình cập nhật bộ đếm của mình như thế nào khi xảy ra sự kiện nội bộ, khi gửi thông điệp và khi nhận thông điệp. Hãy giải thích cơ chế

hiệu chỉnh nhãn thời gian khi một thông điệp đến với nhãn thời gian nhỏ hơn nhãn của thông điệp đã được gửi

a) Giải thuật Cristian

- Client gửi yêu cầu đến server tại T_1, server nhận tại T_2, gửi lại phản hồi tại T_3, client nhận tại T_4.
- Giả định độ trễ đối xứng:
  - Độ trễ mạng: $\delta = (T_4 - T_1) - (T_3 - T_2)$
  - Độ lệch: $\theta = ((T_2 - T_1) + (T_3 - T_4)) / 2$
    → Đồng hồ client được hiệu chỉnh thành T_server + θ.
- Cristian giúp ước lượng độ trễ mạng và điều chỉnh thời gian của client theo server.

b) Giải thuật Berkeley

- Một máy được chọn làm **time daemon** (máy chủ thời gian).
- Nó hỏi thời gian của các máy khác, tính trung bình (bao gồm cả thời gian của mình), rồi gửi hiệu chỉnh đến các máy.
- Không phụ thuộc UTC, chỉ đảm bảo đồng bộ nội bộ giữa các máy.
- Không được phép "vặn ngược" thời gian – chỉ điều chỉnh tốc độ nhanh/chậm để đạt đồng bộ.
  → So sánh:
  - *Cristian:* máy chủ cố định, client điều chỉnh.
  - *Berkeley:* mọi máy tham gia điều chỉnh, đồng bộ tập thể.

c) Giải thuật trung bình và tham chiếu quảng bá:

- **Đồng bộ trung bình:** Mỗi nút chia sẻ thời gian của mình, tính trung bình, rồi hiệu chỉnh về giá trị trung bình → đồng bộ cao nhưng không tuyệt đối.
- **Tham chiếu quảng bá (Reference Broadcast Synchronization – RBS):**
  Một nút phát gói tin tham chiếu, các nút khác ghi lại thời điểm nhận (theo đồng hồ cục bộ). So sánh thời gian nhận để tính sai lệch tương đối giữa các đồng hồ.
  Ưu điểm: giảm ảnh hưởng trễ truyền; nhược: chỉ đồng bộ tương đối, không có "thời gian tuyệt đối".

d) Khó khăn trong đồng bộ vật lý:

- Mỗi máy có sai số trôi (clock drift), độ trễ truyền thay đổi, và không có đồng hồ toàn cục hoàn hảo.

e) Đồng hồ logic Lamport:

- Giải pháp thay thế khi không thể đồng bộ tuyệt đối – chỉ cần đảm bảo thứ tự nhân quả giữa các sự kiện.
- **Nguyên tắc:**
    1. Mỗi tiến trình có đồng hồ logic C_i.
    2. Khi có sự kiện nội bộ → $C_i = C_i + 1$.
    3. Khi gửi thông điệp m → gắn ts(m) = C_i.
    4. Khi nhận thông điệp → C□ = max(C□, ts(m)) + 1.

- Nhờ đó, nếu a → b (a xảy ra trước b) thì C(a) < C(b).
- Nếu thông điệp đến với nhãn nhỏ hơn, đồng hồ vẫn tăng để đảm bảo thứ tự nhân quả.