Câu hỏi bài tập chương 2: Trao đổi thông tin trong hệ thống phân tán

Câu 1:

1. Mô hình OSI và cơ chế bao đóng dữ liệu

a) Bảy tầng của mô hình OSI và chức năng chính

Mô hình OSI (Open Systems Interconnection) bao gồm 7 tầng, mỗi tầng có chức năng cụ thể để hỗ trợ truyền thông giữa các thiết bị mạng:

Tầng	Tên tầng	Chức năng chính
7	Application	Giao tiếp trực tiếp với phần mềm ứng dụng (HTTP, SMTP, FTP...).
6	Presentation	Chuyển đổi định dạng dữ liệu, mã hóa, giải mã, nén và giải nén.
5	Session	Quản lý phiên kết nối giữa hai thiết bị, thiết lập/duy trì/kết thúc phiên.
4	Transport	Đảm bảo truyền dữ liệu tin cậy (TCP, UDP), chia nhỏ dữ liệu thành segment.
3	Network	Định tuyến và chuyển tiếp gói tin giữa các mạng (IP, ICMP...).
2	Data Link	Đóng khung dữ liệu, kiểm tra lỗi, kiểm soát truy cập đường truyền (Ethernet, PPP).
1	Physical	Truyền dữ liệu dạng bit qua các phương tiện vật lý như cáp, sóng vô tuyến.

b) Cơ chế "bao đóng" (Encapsulation) và "bóc tách" (Decapsulation)

Encapsulation (Bao đóng)

Là quá trình khi dữ liệu đi từ tầng Application xuống tầng Physical, mỗi tầng sẽ thêm phần điều khiển (header/trailer) riêng vào dữ liệu của tầng trước.

Ví dụ:

- Tầng 4 (Transport): thêm TCP/UDP header vào dữ liệu ứng dụng.
- Tầng 3 (Network): thêm IP header vào TCP segment.

- Tầng 2 (Data Link): thêm MAC header và trailer (ví dụ: FCS).
- Tầng 1 (Physical): chuyển dữ liệu thành tín hiệu điện/ánh sáng/sóng.

Decapsulation (Bóc tách)

Xảy ra tại thiết bị nhận, khi dữ liệu đi từ tầng Physical lên Application. Mỗi tầng lần lượt gỡ bỏ thông tin điều khiển và xử lý phần payload.

Ví dụ thông tin điều khiển thêm ở các tầng

- Tầng 4: TCP header chứa số cổng, số thứ tự, checksum.
- Tầng 3: IP header chứa địa chỉ IP nguồn/đích, TTL.
- Tầng 2: Ethernet header chứa địa chỉ MAC nguồn/đích.

2. Phân mảnh gói dữ liệu và quá trình xử lý trong IP

a) Tại sao cần phân mảnh gói?

Phân mảnh (Fragmentation) là cần thiết vì:

- Giới hạn MTU (Maximum Transmission Unit): mỗi loại mạng vật lý có giới hạn kích thước gói tin có thể truyền (ví dụ: Ethernet MTU = 1500 byte).
- Nếu gói tin IP vượt quá MTU của tầng Data Link, thì cần phân mảnh để tránh mất mát dữ liệu.

---

b) Quá trình phân mảnh và tái cấu trúc

Tại sender (hoặc router):

- Gói tin IP lớn được chia thành nhiều mảnh (fragment).
- Mỗi fragment có:
    - Identification: chung cho các fragment cùng gốc.
    - Fragment Offset: vị trí của fragment trong toàn bộ gói gốc.
    - MF (More Fragment): đánh dấu fragment cuối (MF = 0).

Tại receiver:

- Bộ nhận kiểm tra Identification để nhóm các fragment lại.
- Sắp xếp các fragment theo Offset và ghép lại để tạo gói IP ban đầu.
- Nếu thiếu fragment hoặc trễ thời gian, gói sẽ bị loại bỏ.

---

c) Ảnh hưởng của phân mảnh đến hiệu năng và độ tin cậy

Tác động đến hiệu năng:

- Tăng chi phí xử lý tại thiết bị (tách và ghép fragment).
- Nếu một fragment bị mất, toàn bộ gói phải gửi lại.
- Fragment càng nhỏ → overhead càng lớn do mỗi fragment có thêm header.

Tác động đến độ tin cậy:

- Tăng khả năng lỗi: mất một fragment = mất toàn bộ gói.
- Khó khăn trong quản lý và bảo trì.
- Một số thiết bị chặn các gói đã bị phân mảnh → ảnh hưởng khả năng truyền thông.

Câu 2: Phần mềm trung gian (Middleware)

a) Định nghĩa và vị trí của Middleware trong mô hình OSI và TCP/IP

Định nghĩa Middleware:

Middleware là phần mềm trung gian nằm giữa hệ điều hành và các ứng dụng phân tán. Nó đóng vai trò như cầu nối giúp các ứng dụng trên các máy khác nhau giao tiếp và phối hợp với nhau một cách trơn tru, ẩn đi các chi tiết phức tạp về mạng, hệ điều hành và phần cứng.

Vị trí trong ngăn xếp giao thức:

- Middleware không nằm trong mô hình OSI hoặc TCP/IP chính thức nhưng hoạt động ở tầng trên cùng của mô hình này, thường giữa tầng Transport và Application.
- Trong mô hình OSI:
    - Middleware thường hoạt động trên tầng 4 (Transport) và hỗ trợ các tầng 5-7 (Session, Presentation, Application).
- Trong mô hình TCP/IP:
    - Middleware thường hoạt động trên tầng Transport, cung cấp dịch vụ cho tầng Application.

b) Phân loại Middleware và ví dụ tiêu biểu

Middleware được phân thành các loại theo mức độ mở và khả năng tương tác:

Loại Middleware	Đặc điểm chính	Ví dụ tiêu biểu
Mở hoàn toàn	Hệ thống độc lập, không cần chuẩn chung; thường do một nhà cung cấp kiểm soát	ActiveX, OLE (Microsoft), Java RMI
Giao diện mở	Có API chuẩn nhưng không đảm bảo tương thích hoàn toàn trong giao tiếp mạng	ODBC (Open Database Connectivity)

Giao thức mở	Có giao thức chung cho truyền thông và tương tác giữa các hệ khác nhau	CORBA (Common Object Request Broker Architecture), DRDA (IBM)
Riêng (Proprietary)	Đóng, chỉ dùng nội bộ, không công bố tiêu chuẩn giao tiếp	Các giải pháp middleware độc quyền của doanh nghiệp như SAP NetWeaver

c) Vai trò của Middleware trong hệ thống phân tán

Middleware đóng vai trò trung gian thông minh và linh hoạt, cụ thể:

1. Ẩn chi tiết kỹ thuật mạng:
    - Ứng dụng không cần lo lắng về giao thức truyền thông (TCP/IP, RPC...) hay định dạng dữ liệu.
2. Tạo giao diện lập trình chung (API):
    - Cho phép các ứng dụng lập trình viên viết ra dễ dàng giao tiếp với dịch vụ từ xa mà không quan tâm đến vị trí hay hệ điều hành đích.
3. Hỗ trợ tính phân tán và mở rộng:
    - Giúp các ứng dụng mở rộng quy mô ra nhiều máy chủ mà vẫn giữ tính nhất quán.
4. Đảm bảo tính tương thích:
    - Kết nối giữa các hệ thống dị chủng (Windows ↔ Unix, Java ↔ C++) nhờ vào định dạng và giao thức chuẩn hóa.
5. Hỗ trợ an toàn và quản lý giao dịch:
    - Middleware hiện đại thường hỗ trợ cơ chế bảo mật, quản lý phiên làm việc, và giao dịch phân tán (ví dụ: hai-phase commit).

Câu 3: Khái niệm và vai trò

a) Định nghĩa "thông điệp" và "chuyển thông điệp" trong hệ thống phân tán

Thông điệp (Message):

Là đơn vị dữ liệu dùng để truyền thông tin giữa các tiến trình trong một hệ thống phân tán. Mỗi thông điệp thường bao gồm:

- Header: chứa địa chỉ nguồn, đích, kiểu dữ liệu...
- Payload: phần dữ liệu chính được truyền tải.

Ví dụ: Một thông điệp chứa dữ liệu yêu cầu truy xuất cơ sở dữ liệu từ client đến server.

Chuyển thông điệp (Message Passing):

Là cơ chế giao tiếp chính trong hệ thống phân tán, cho phép các tiến trình độc lập (thường nằm trên các máy tính khác nhau) trao đổi dữ liệu bằng cách gửi và nhận thông điệp thông qua mạng.

Hai hình thức phổ biến:

- Gửi đồng bộ (Synchronous): tiến trình gửi phải chờ đến khi nhận xong.
- Gửi bất đồng bộ (Asynchronous): tiến trình gửi không cần chờ, tiếp tục xử lý sau khi gửi.

---

b) Ưu – nhược điểm của chuyển thông điệp so với chia sẻ bộ nhớ

🔷 *Ưu điểm của chuyển thông điệp:*

Tiêu chí	Ưu điểm
Tính trong suốt	Phù hợp với môi trường phân tán vì không yêu cầu không gian bộ nhớ chung. Các tiến trình không cần biết vị trí vật lý của nhau.
Tính linh hoạt	Cho phép giao tiếp qua mạng Internet, vượt qua ranh giới địa lý và hệ điều hành.
Độ bảo mật	Dễ kiểm soát thông điệp gửi – nhận, hạn chế việc tiến trình khác truy cập trái phép vùng nhớ.
Tăng khả năng mở rộng	Dễ mở rộng ra nhiều máy chủ hoặc tiến trình mà không ảnh hưởng đến kiến trúc.

🔶 Nhược điểm của chuyển thông điệp:

Tiêu chí	Nhược điểm
Hiệu năng	Giao tiếp qua mạng tốn thời gian hơn so với truy cập bộ nhớ chung nội bộ.
Độ phức tạp	Phải xử lý các vấn đề về định tuyến, mất gói, trễ mạng, thứ tự thông điệp.
Tính tin cậy	Dễ bị mất thông điệp nếu không có cơ chế xác nhận hoặc truyền lại.

Câu 4: Giao diện truyền thông điệp MPI:

---

a) Mục đích và bối cảnh ra đời của MPI (Message Passing Interface)

Mục đích của MPI:

MPI (Message Passing Interface) là một chuẩn giao tiếp để lập trình song song trong các hệ thống phân tán. Mục tiêu của MPI là cung cấp một tập hợp giao diện lập trình chuẩn, độc lập với nền tảng, giúp các tiến trình trên nhiều máy tính (hoặc nhiều lõi) có thể giao tiếp hiệu quả thông qua cơ chế gửi – nhận thông điệp.

Bối cảnh ra đời:

- Trước khi MPI xuất hiện, có nhiều thư viện message passing khác nhau như PVM (Parallel Virtual Machine), Express, NX… nhưng không có một chuẩn thống nhất.
- Năm 1994, MPI-1 được phát triển bởi MPI Forum (gồm các nhà nghiên cứu, công ty phần mềm, và tổ chức siêu máy tính).
- MPI ra đời để:
    - Chuẩn hóa giao tiếp trong hệ thống phân tán.
    - Tăng khả năng tương thích giữa các hệ thống và phần mềm.
    - Tối ưu hiệu năng cho môi trường High Performance Computing (HPC).

b) Bốn hàm nguyên thủy cơ bản trong MPI

Hàm MPI	Đồng bộ / Bất đồng bộ	Cơ chế vùng đệm	Mô tả ngắn gọn
MPI_Send	Đồng bộ chuẩn (standard)	Không đảm bảo sử dụng vùng đệm riêng	Hàm gửi thông điệp, tiến trình gửi có thể bị chặn cho đến khi hệ thống đảm bảo message có thể được lưu trữ an toàn (có thể gửi hoặc được nhận).
MPI_Bsend	Không đồng bộ có vùng đệm	Có dùng vùng đệm do lập trình viên cung cấp (qua MPI_Buffer_attach)	Cho phép tiến trình gửi tiếp tục thực thi ngay sau khi copy thông điệp vào vùng đệm.
MPI_Ssend	Đồng bộ chặt (synchronous)	Không dùng vùng đệm	Gửi thông điệp nhưng chỉ hoàn tất khi tiến trình nhận thực sự nhận message (gọi MPI_Recv). Cả hai bên phải phối hợp cùng lúc.
MPI_Irecv	Không đồng bộ (asynchronous)	Không chặn tiến trình nhận	Tiến trình đăng ký nhận một thông điệp mà không cần chờ đợi hoàn tất ngay lập tức, thường dùng kết hợp với MPI_Wait hoặc MPI_Test để kiểm tra trạng thái.

Câu 6: Cơ chế hoạt động chung của RPC (Remote Procedure Call):

a) Mô hình Stub–Skeleton trong RPC

Khái niệm RPC:

RPC (Remote Procedure Call) là cơ chế cho phép một tiến trình gọi hàm từ xa như thể nó đang gọi một hàm cục bộ, ẩn đi các chi tiết liên quan đến truyền thông mạng.

Mô hình Stub–Skeleton:

Mô hình này là kiến trúc phổ biến giúp triển khai RPC. Nó gồm các thành phần sau:

Thành phần	Vai trò
Client Stub	Thay thế cho hàm thật ở phía client. Nó đóng gói (marshal) tham số thành thông điệp và gửi đi.
Server Stub (Skeleton)	Nhận thông điệp từ client, bóc tách (unmarshal) và gọi hàm thực sự ở phía server.
RPC Runtime	Hệ thống trung gian hỗ trợ truyền thông điệp giữa client và server.

b) Phác họa 10 bước của một cuộc gọi RPC và ý nghĩa

Dưới đây là mô tả chi tiết 10 bước tuần tự của một cuộc gọi RPC, dựa trên hình 2.9 thường thấy trong giáo trình hệ phân tán (như "Distributed Systems – Tanenbaum"):

Tuần tự 10 bước trong một cuộc gọi RPC:

Bước	Mô tả chi tiết
1	Client gọi thủ tục (procedure) → Gọi hàm từ xa thông qua stub, giống gọi hàm cục bộ.
2	Client stub nhận lời gọi, thực hiện marshal tham số → Chuyển đổi tham số thành định dạng có thể truyền.
3	Stub gửi thông điệp yêu cầu đến hệ thống truyền thông.
4	Hệ thống truyền thông client gửi message qua mạng đến server.
5	Hệ thống truyền thông server nhận message, gửi đến server stub.
6	Server stub nhận message, thực hiện unmarshal, rồi gọi hàm thật.
7	Thủ tục thật trên server được thực thi và trả về kết quả.
8	Server stub nhận kết quả, thực hiện marshal để đóng gói dữ liệu trả về.

9	Gửi message phản hồi về cho client qua hệ thống truyền thông.
10	Client stub nhận phản hồi, unmarshal kết quả → Trả về kết quả như hàm cục bộ.