ĐẠI HỌC BÁCH KHOA

CÁC HỆ PHÂN TÁN

CHƯƠNG 2
KIẾN TRÚC HỆ PHÂN TÁN

TS. TRẦN HẢI ANH

Tham khảo bài giảng của PGS. TS. Hà Quốc Trung

Nội dung

1. Khái niệm kiến trúc
2. Kiến trúc hệ thống
3. Middleware trong các kiến trúc
4. Quản lý tự động trong hệ phân tán

1. Khái niệm kiến trúc

1.1. Kiến trúc

- Kiến trúc
  - Cách thức phân chia hệ thống thành các thành phần, mô tả sự tương tác giữa các thành phần
  - Phân tầng, Client-Server, P2P,
- Hệ thống được tách thành các thành phần
- Mỗi thành phần có các giao diện định nghĩa và qui định cách sử dụng các chức năng của thành phần trong môi trường của thành phần
- Nếu giao diện không thay đổi thì có thể thay thế thành phần khác
- Các thành phần kết nối lẫn nhau
- Cách thức kết nối các thành phần: kiến trúc

1.2. Các loại kiến trúc thường dùng trong hệ phân tán

- Kiến trúc phân tầng
- Kiến trúc hướng đối tượng
- Kiến trúc hướng dữ liệu
- Kiến trúc hướng sự kiện

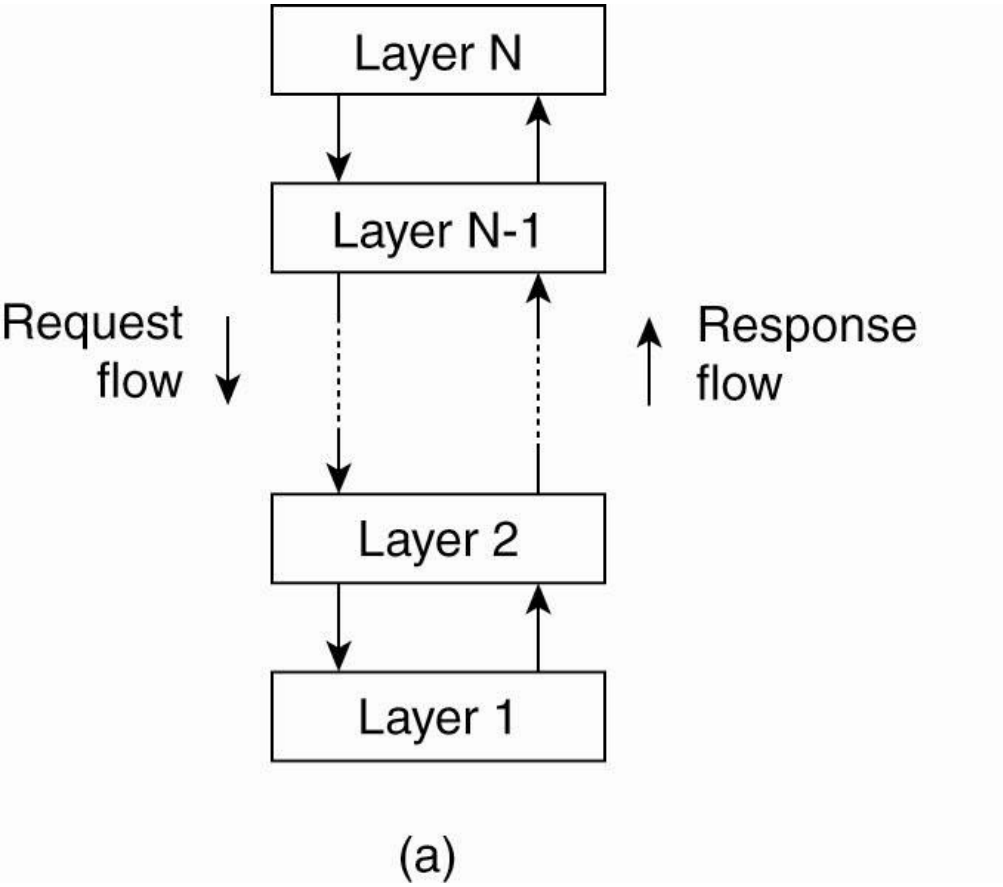
1.2.1. Kiến trúc phân tầng

- Chức năng trên hệ thống được phân rã thành các chức năng con
- Các chức năng con được thực hiện bởi các mô đun phần mềm – các thực thể phần mềm trên các hệ thống khác nhau tương tác với nhau
- Các mô đun phần mềm khác nhau trên cùng hệ thống phối hợp và tương tác với nhau để thực hiện chức năng chung
- Để đơn giản hệ thống cần giảm thiểu liên kết giữa các mô đun: kiến trúc phân tầng

Kiến trúc phân tầng

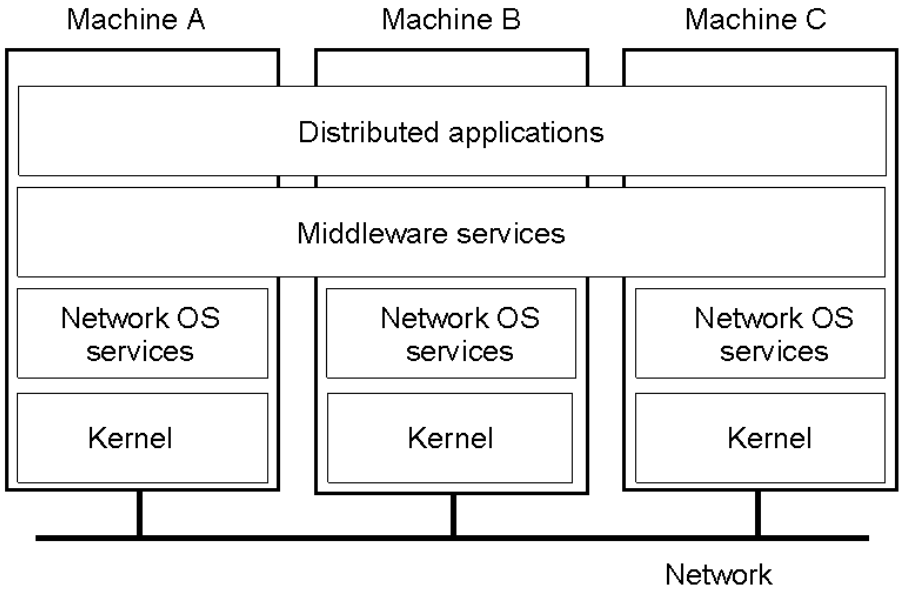
Tầng N

- Thực thể
- Giao thức (4 loại giao thức)
- Dịch vụ
- Điểm truy cập dịch vụ



Các mô hình phân tầng thường gặp

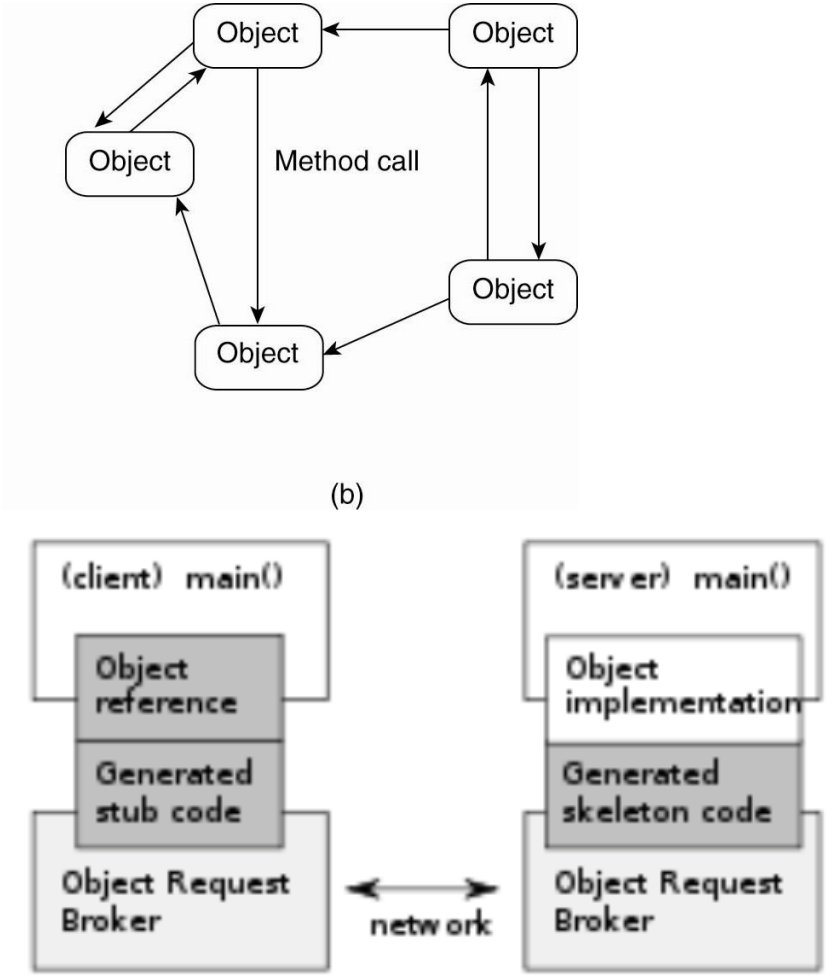
Mô hình OSI



Mô hình Middleware

1.2.2. Kiến trúc hướng đối tượng

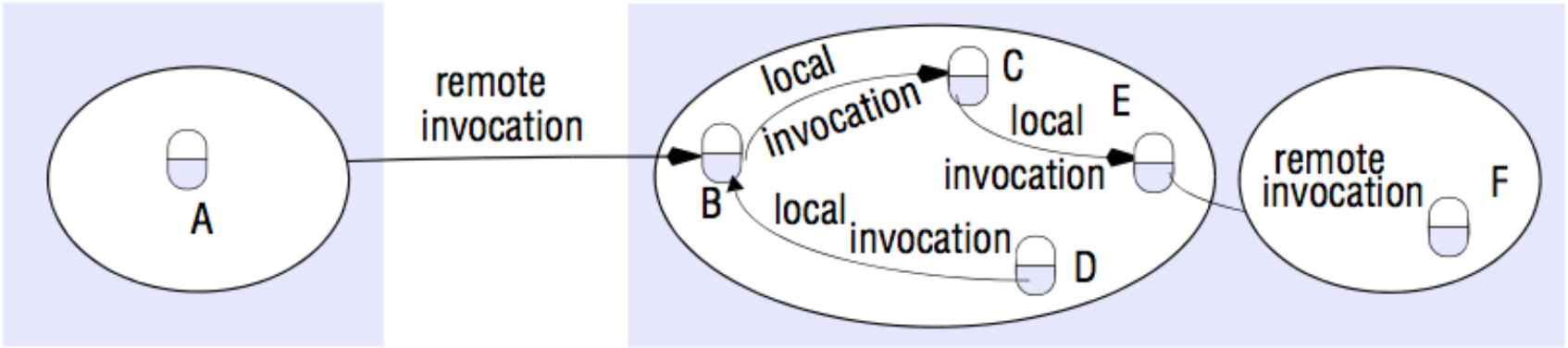
- Thành phần <> đối tượng
- Connector <> Lời gọi phương thức
- Object Client và Object server
- Kết nối lỏng giữa các đối tượng
- Ví dụ: Corba



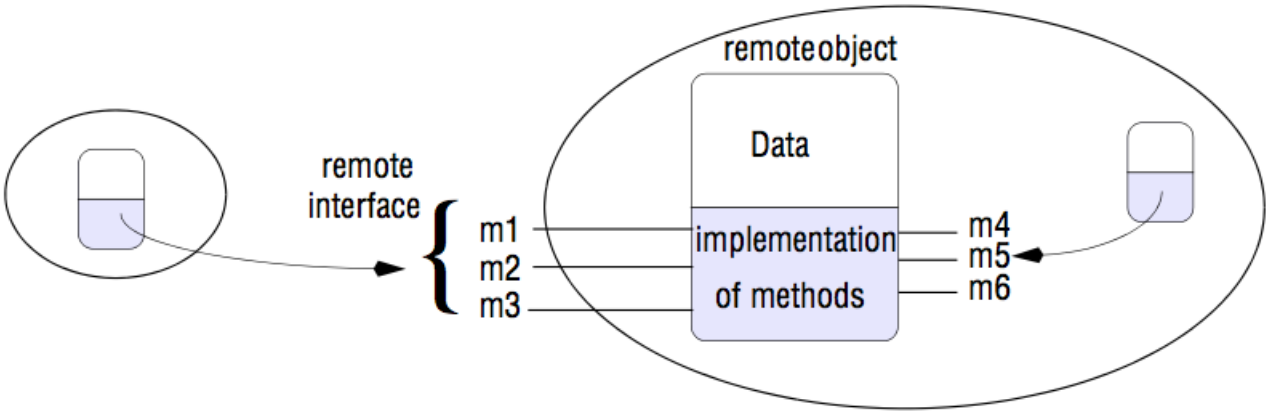
RMI: Lời gọi phương thức từ xa

- Lập trình hướng đối tượng :
  - đối tượng từ xa, ứng dụng phân tán hướng đối tượng
- Các vấn đề cần giải quyết
  - Định vị đối tượng từ xa
  - Trao đổi thông tin với đối tượng
  - Gọi các phương thức của đối tượng
- RMI, T-RMI, DCOM, CORBA

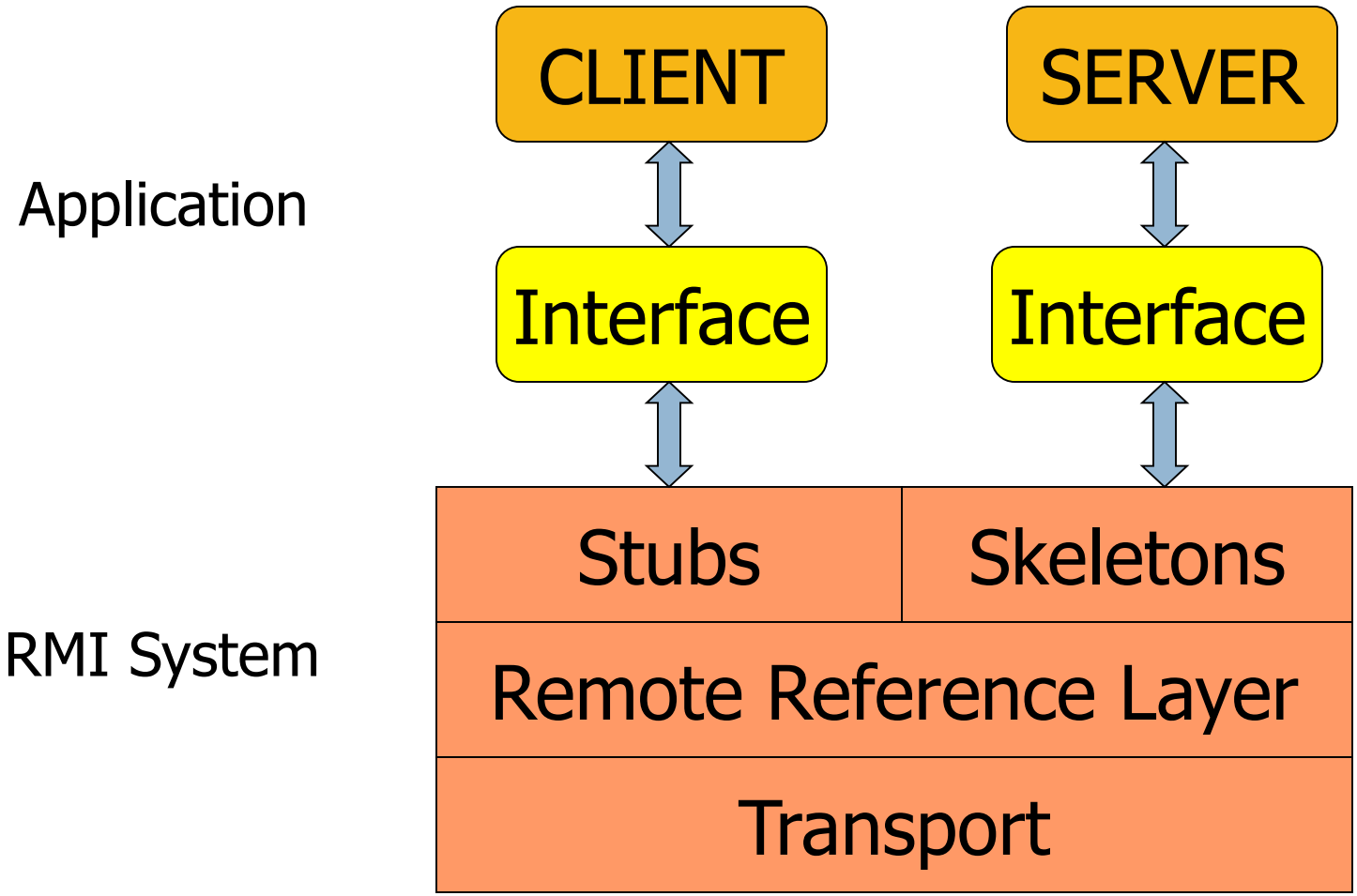
Mô hình đối tượng phân tán



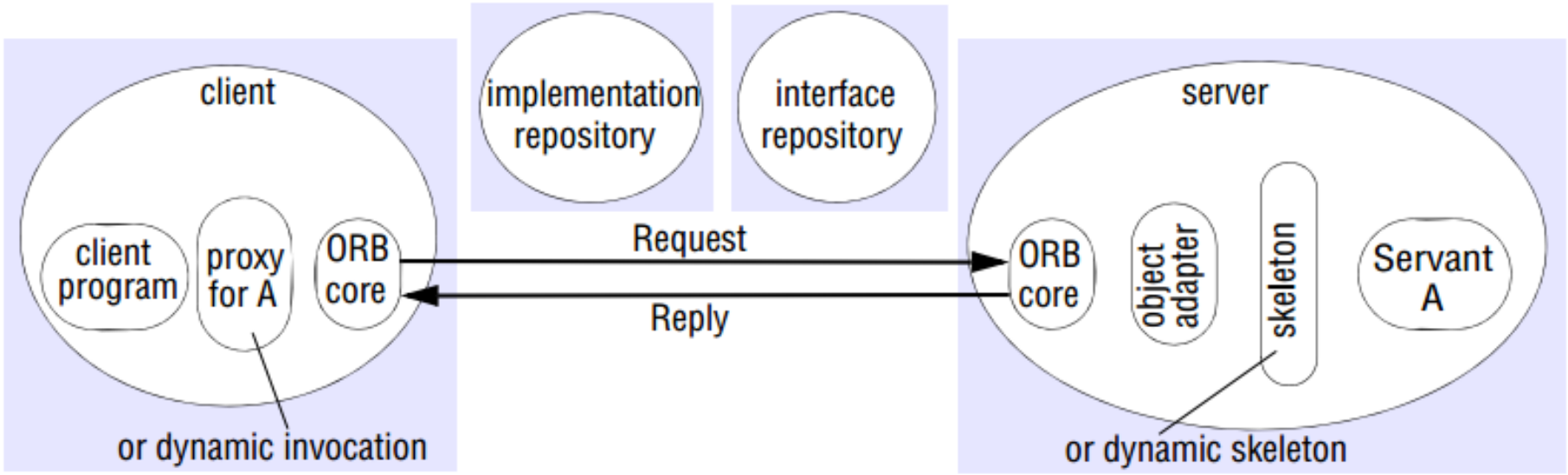
Đối tượng từ xa và giao diện từ xa



Kiến trúc chung RMI



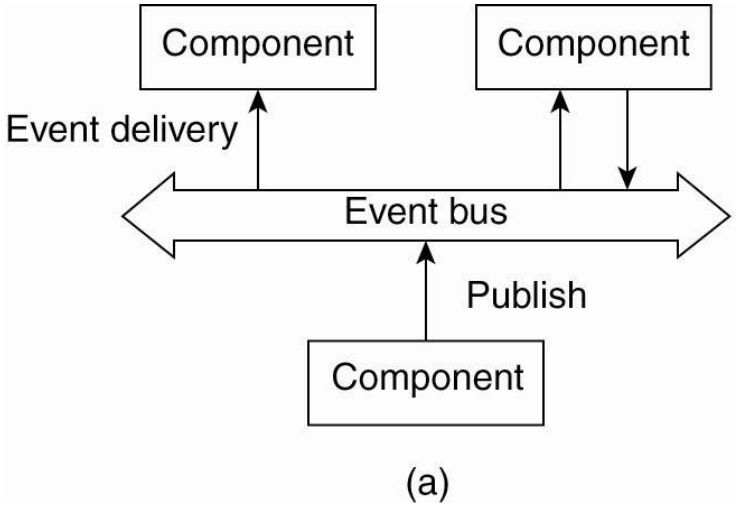
Kiến trúc CORBA



CORBA architecture

1.2.3. Kiến trúc hướng sự kiện

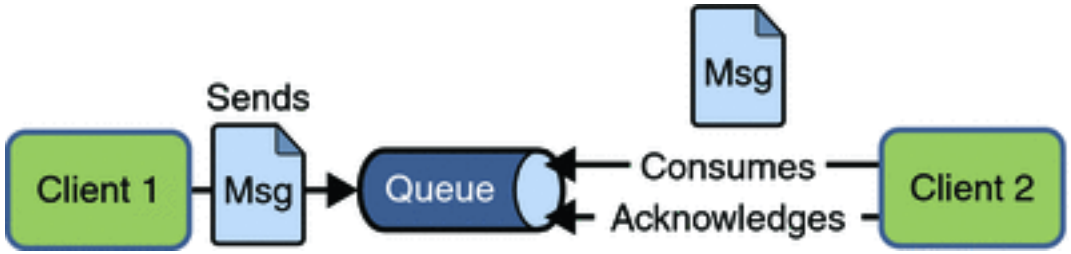
- Thành phần hệ thống trao đổi thông tin với nhau thông qua các sự kiện
- Các sự kiện chứa các thông tin cần trao đổi
- Các sự kiện có thể kích hoạt các thao tác trong các tiến trình
- Có thể thực hiện theo mô hình điểm điểm hoặc mô hình trục quảng bá sự kiện
- Ví dụ
  - mô hình thuê bao/xuất bản
- Liên kết lỏng



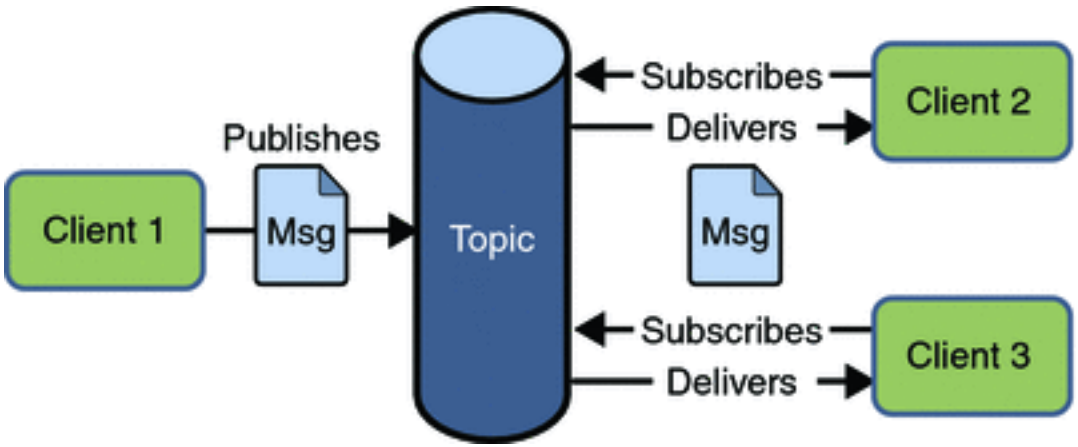
(a)

JMS (Java Message Service)

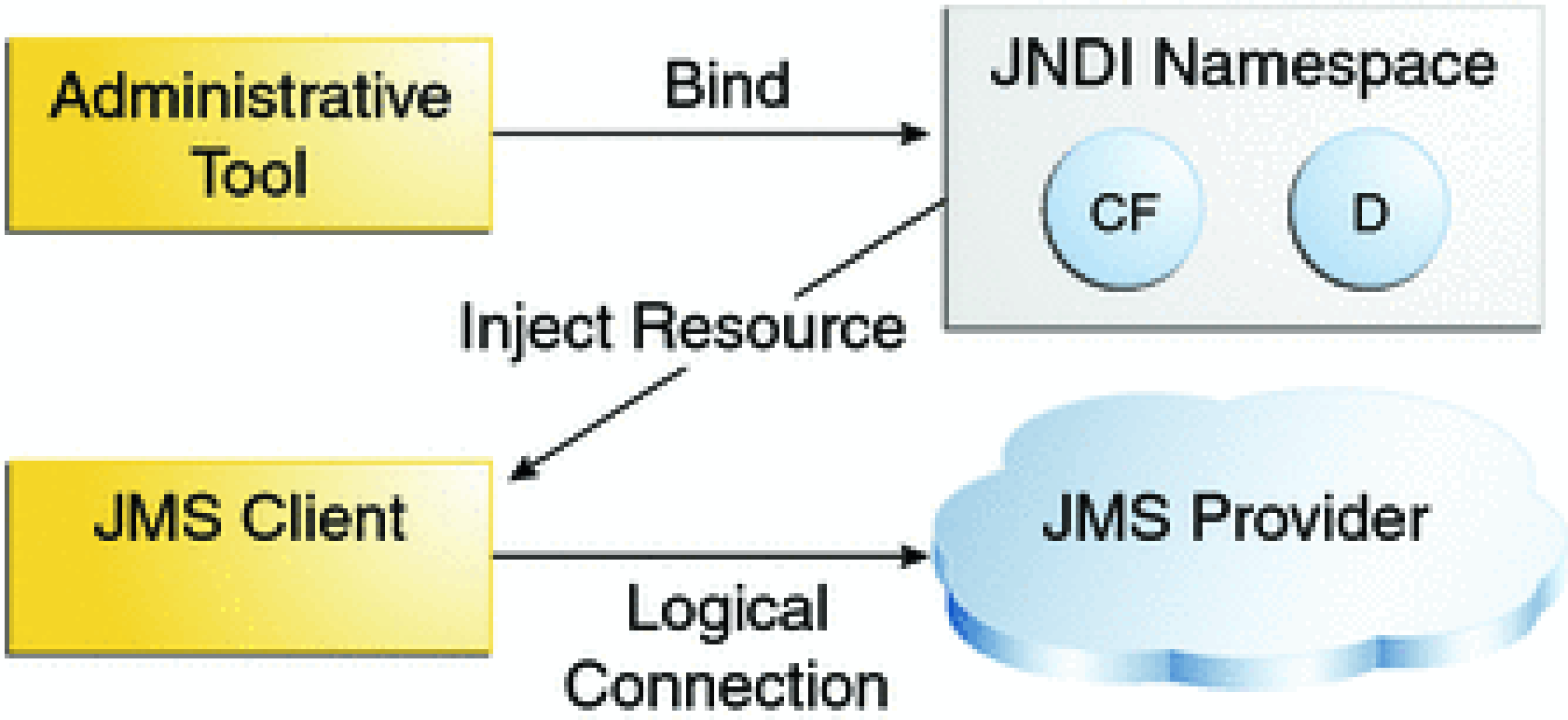
- Point-to-Point Messaging Domain



- Publisher/Subscriber Messaging Domain

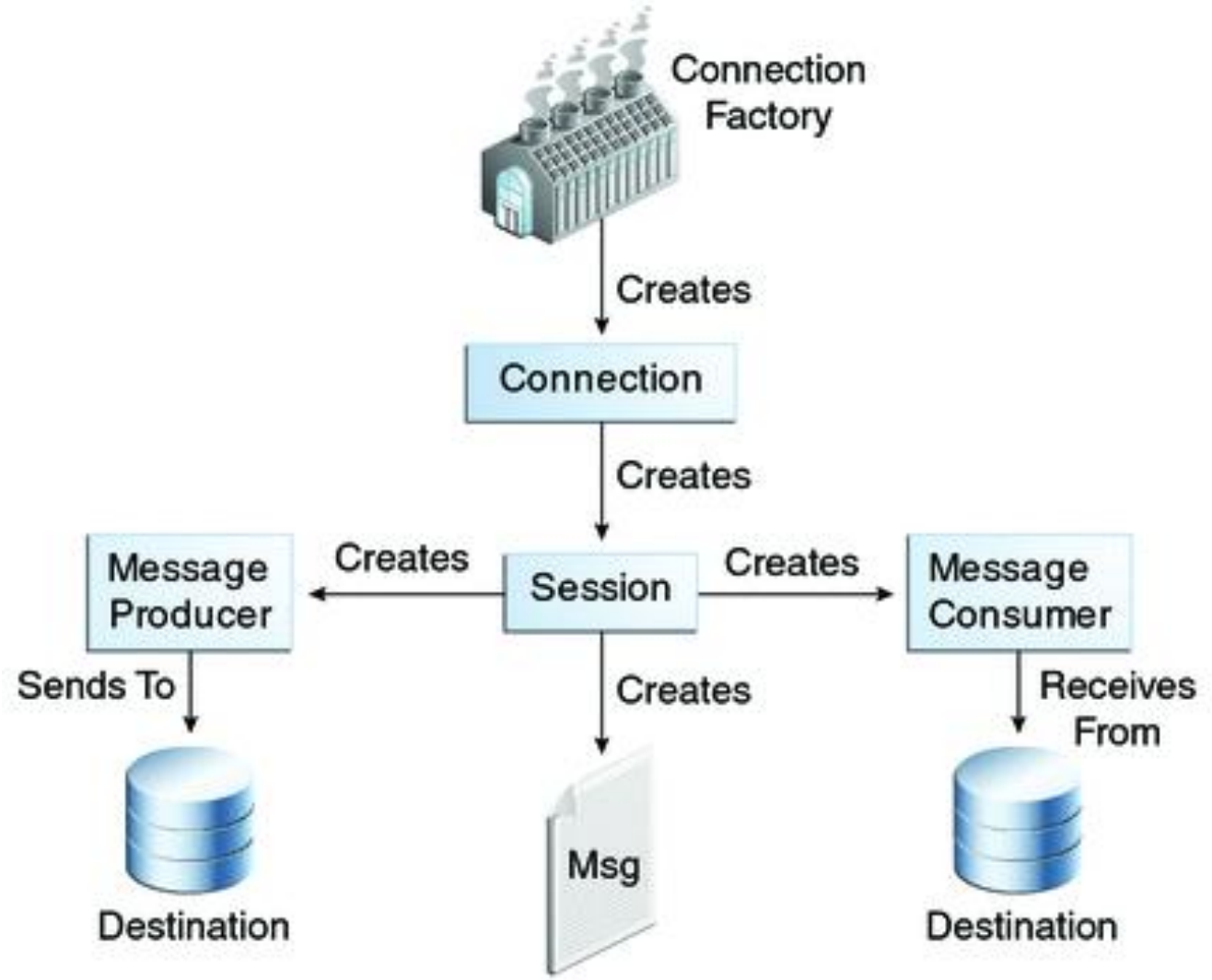


JMS API Architecture

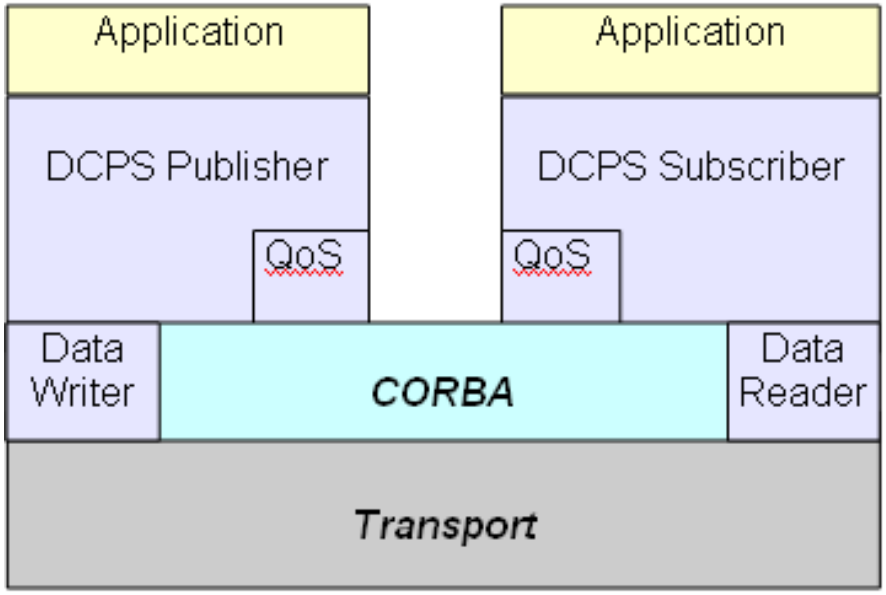


Publish/Subscribe Messaging

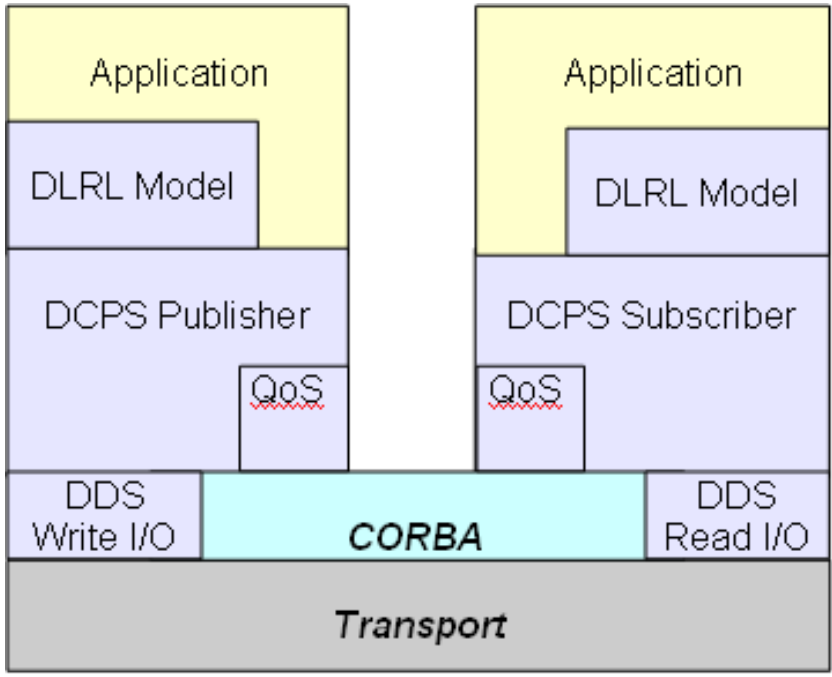
The JMS API Programming Model



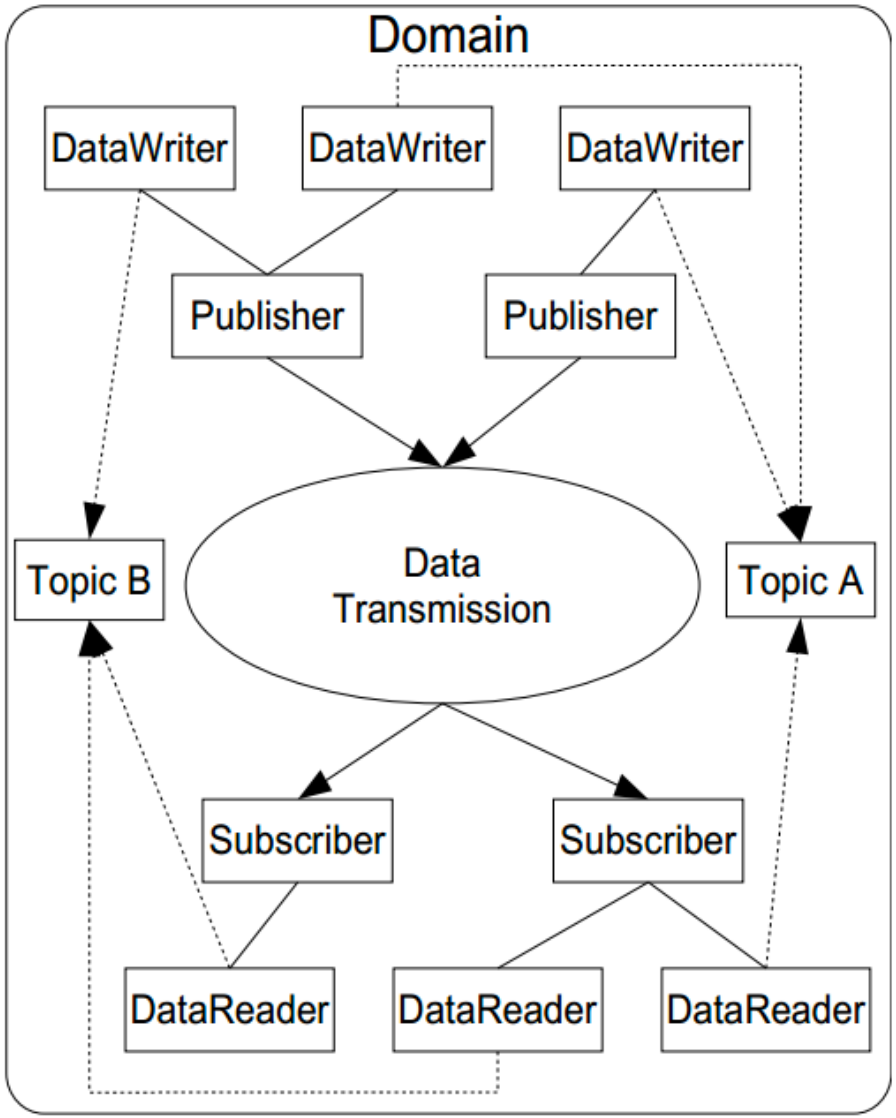
DDS (Data Distribution Service)



DCPS interface



DLRL interface

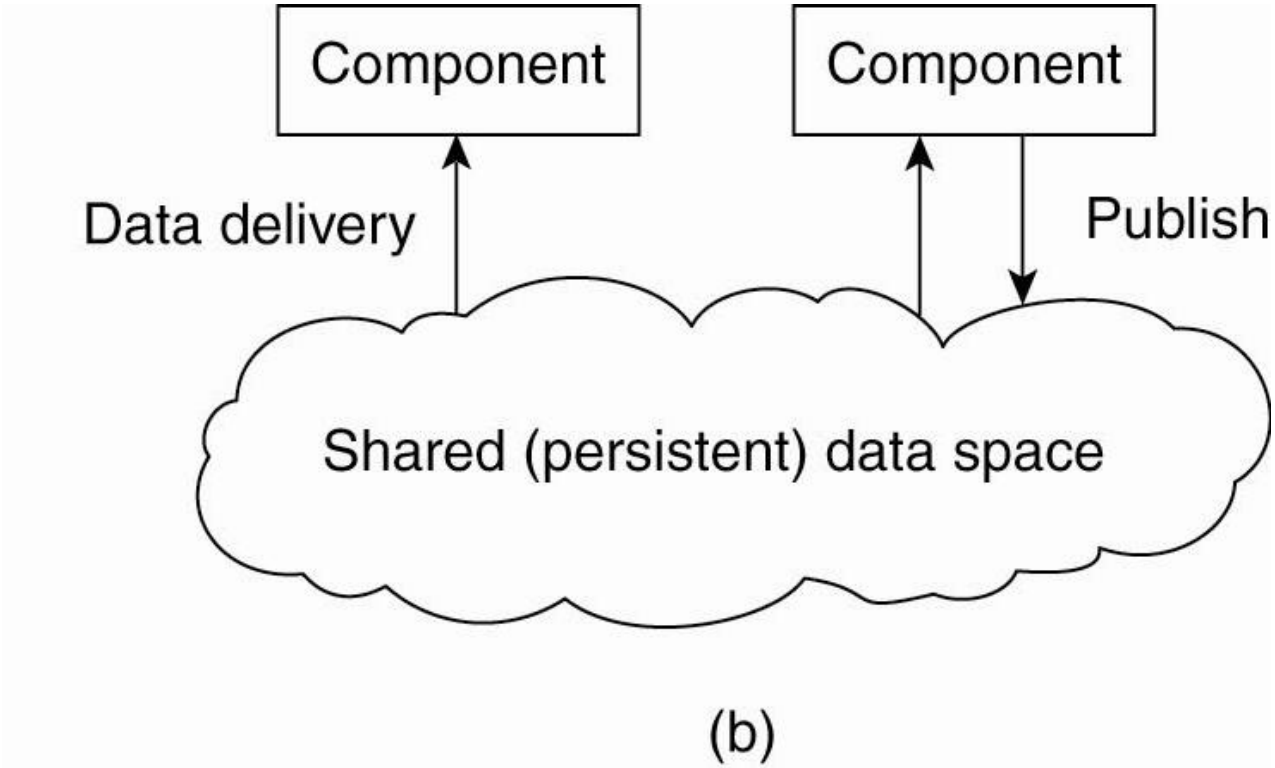

Domain
DataWriter
DataWriter
DataWriter
Publisher
Publisher
Topic B
Data
Transmission
Topic A
Subscriber
Subscriber
DataReader
DataReader
DataReader

Ưu điểm của DDS so với JMS

- Thời gian thực (độ trễ thấp)
- Nhiều ngôn ngữ khác nhau
- Nhiều nền tảng khác nhau

1.2.4. Kiến trúc hướng dữ liệu

- Các thành phần trao đổi thông tin thông qua kho dữ liệu chung



2. Kiến trúc hệ thống

I. Kiến trúc tập trung
II. Kiến trúc không tập trung
III. Kiến trúc hỗn hợp

2.1. Kiến trúc tập trung

2.1.1. Kiến trúc client-server

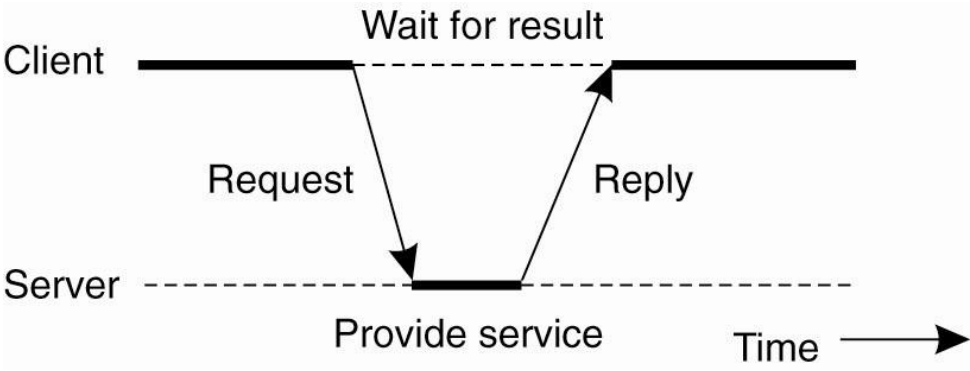
2.1.2. Phân tầng ứng dụng

2.1.3. Kiến trúc đa tầng

2.1.4. Software Agent

2.1.1. Kiến trúc client-server

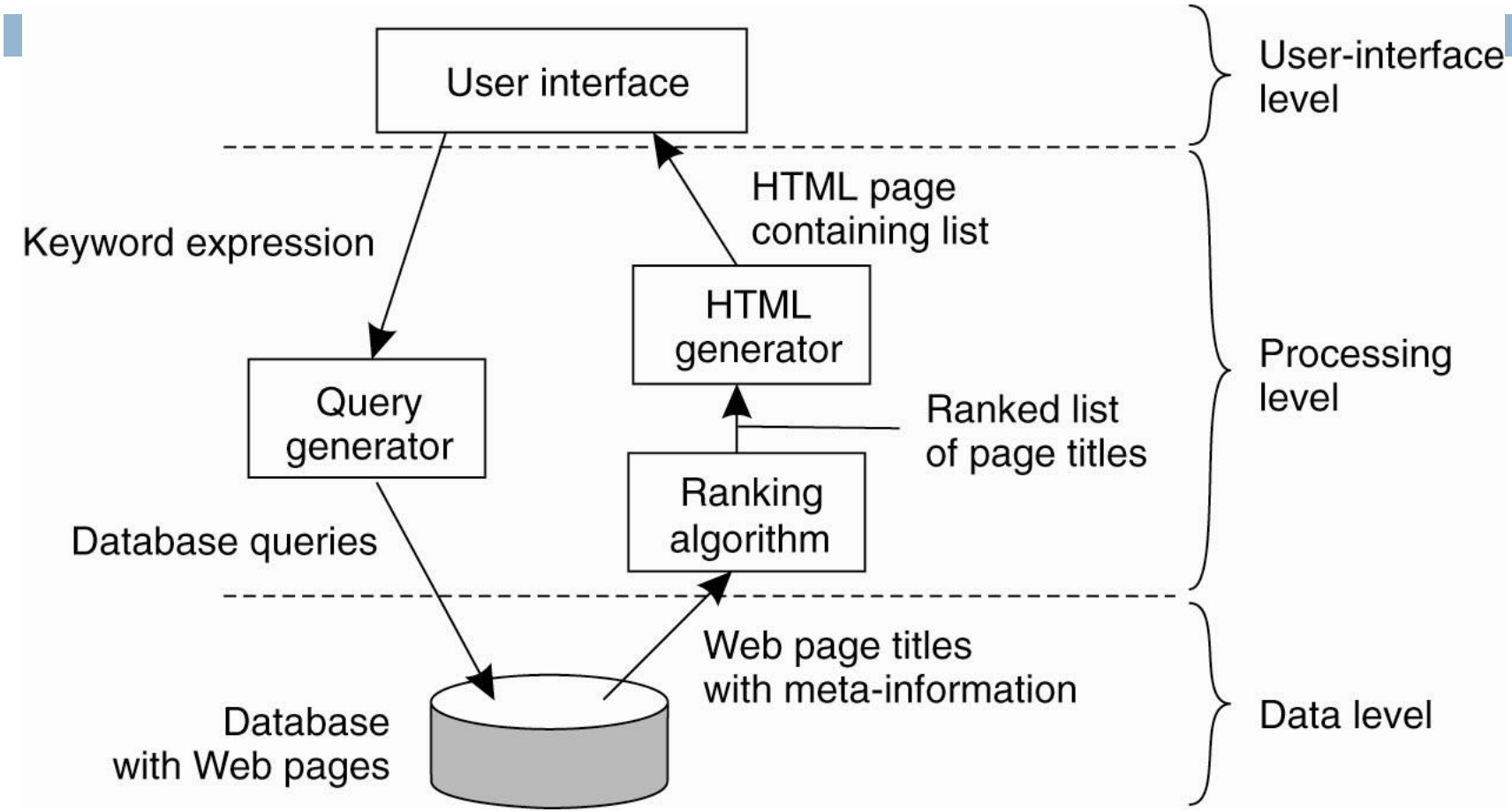
- Client:
  - gửi yêu cầu, nhận kết quả, hiển thị cho NSD
- Server:
  - lắng nghe, nhận yêu cầu, xử lý, trả lời
- Tương tác giữa client và server có thể là hướng kết nối hoặc không hướng kết nối
- Vấn đề
  - Đăng ký server (DNS hoặc dịch vụ thư mục)
  - Có thể lặp lại yêu cầu? (idempotent)
  - Có bộ nhớ trạng thái?



2.1.2. Phân tầng ứng dụng

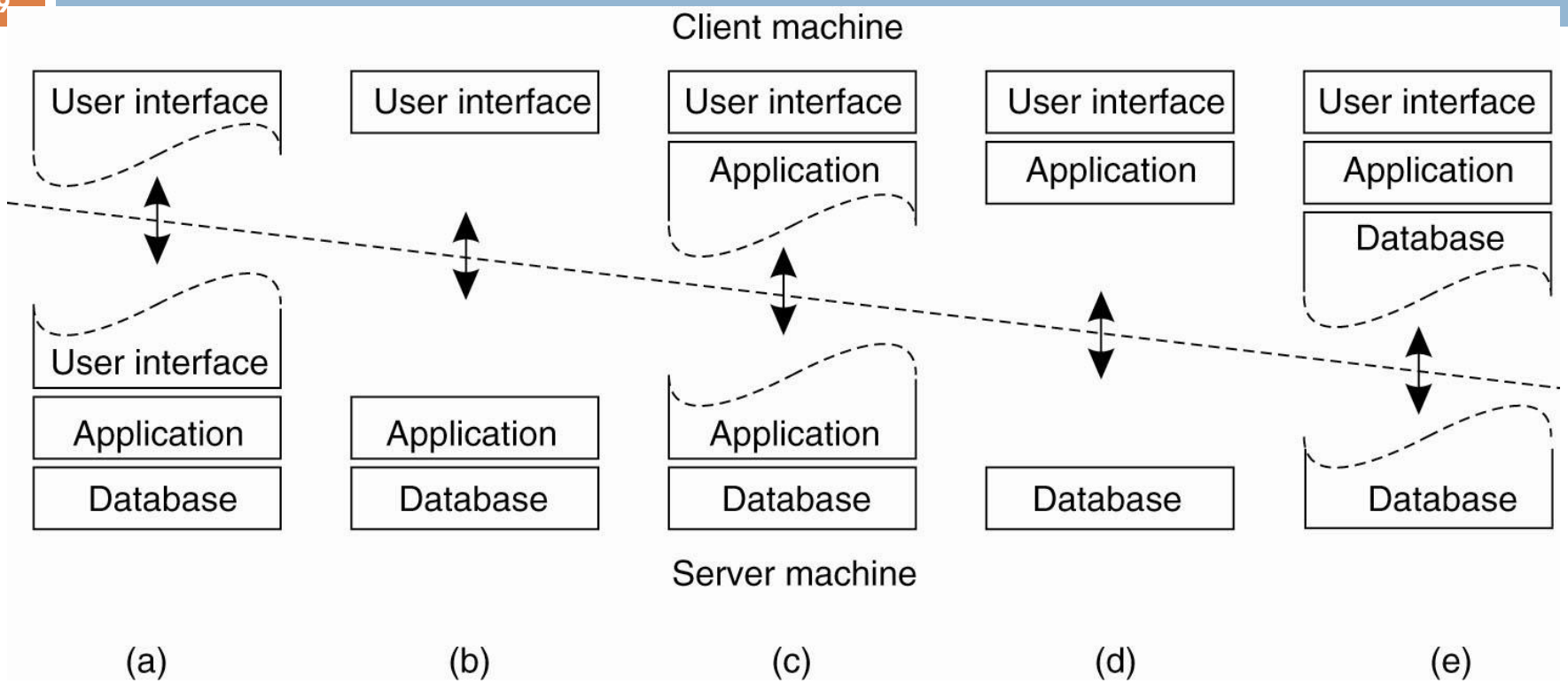
- Các mức phân tầng
  - Giao diện
  - Nghiệp vụ
  - Dữ liệu

Phân tầng ứng dụng tìm kiếm

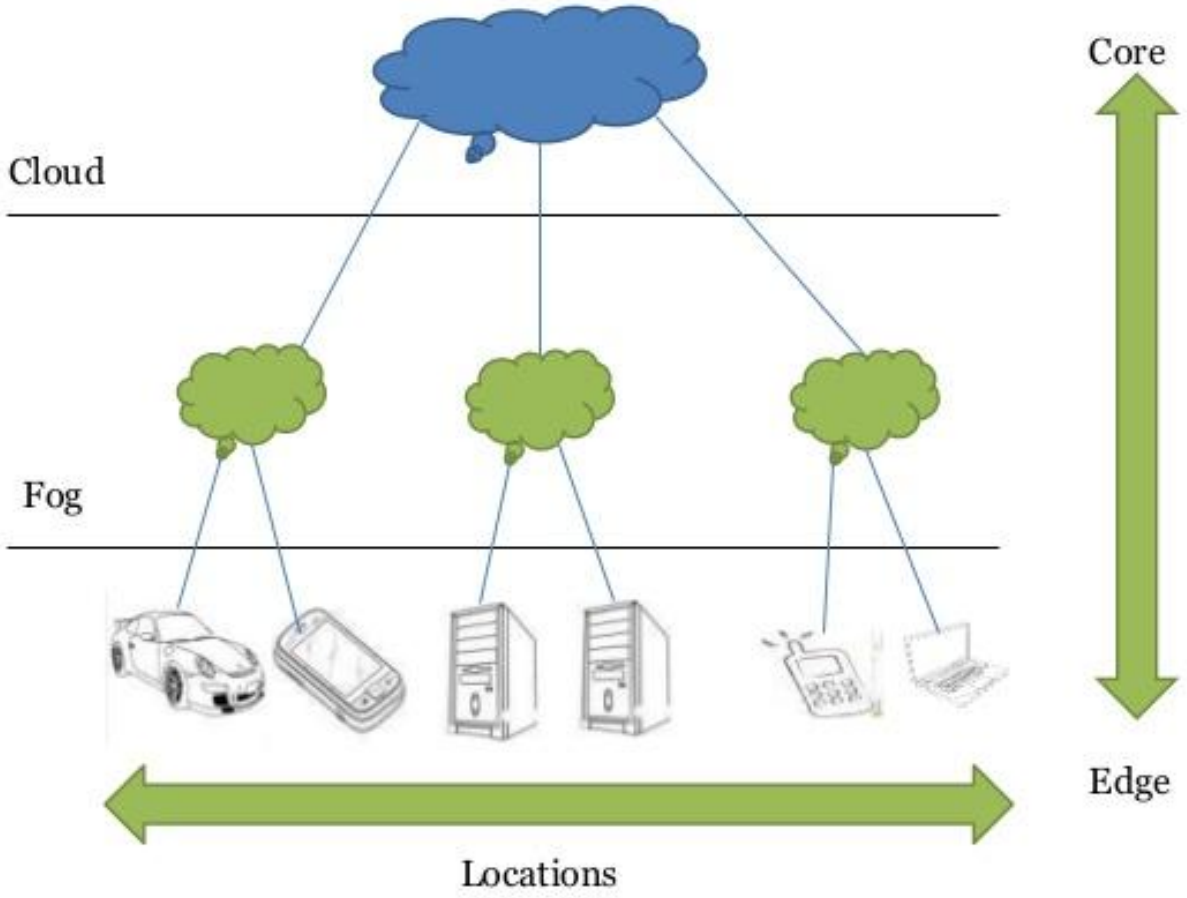


2.1.3. Kiến trúc đa tầng

Các mô hình 2 bên



Cloud & Fog computing



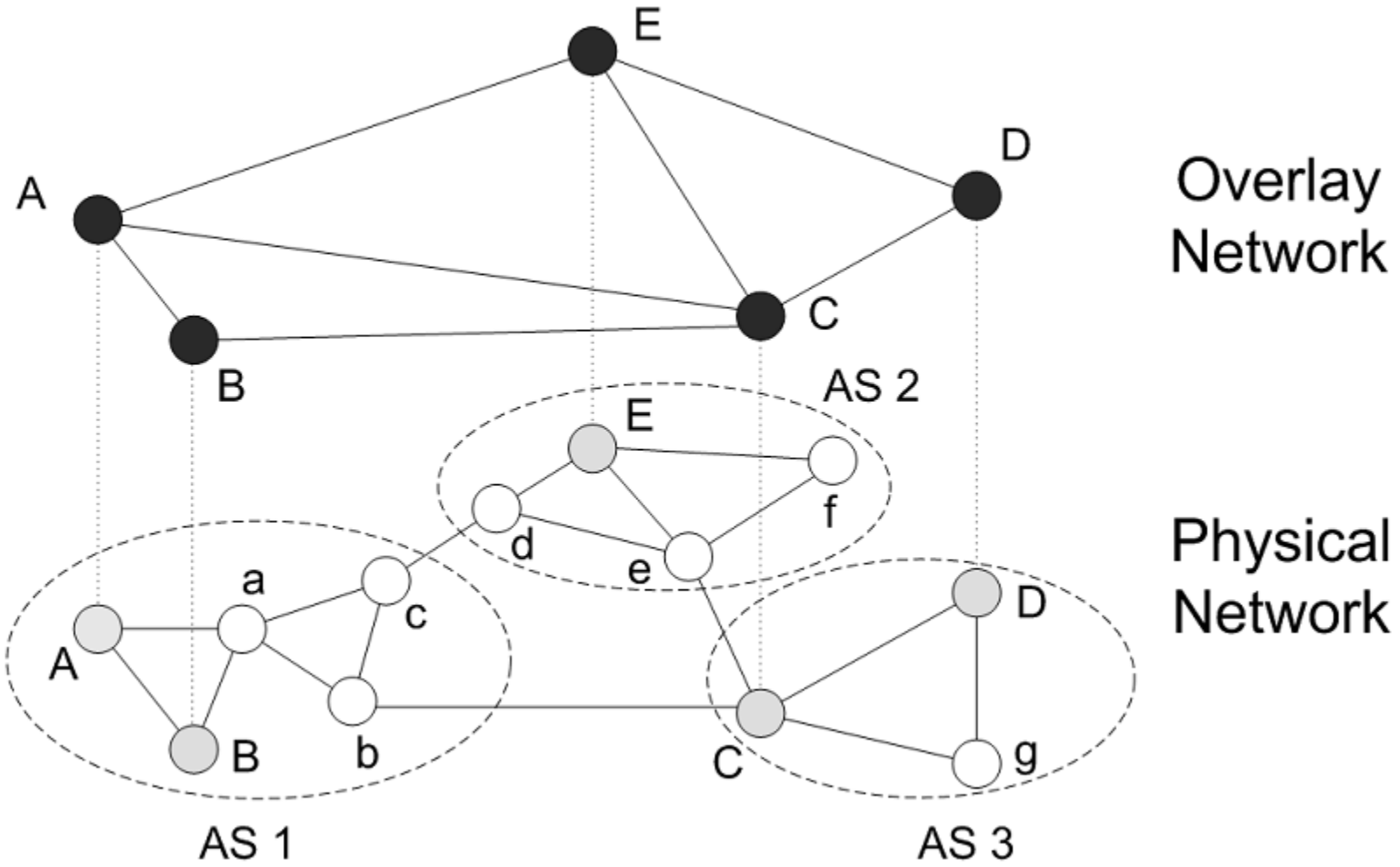
Các mô hình client-server khác

- Hệ thống phân tầng chiều đứng, theo chức năng
- Hệ thống phân tầng theo chiều ngang (theo tải)
  - Proxy, các cơ chế phân tải
- Mã di động (applet, javascript)
- Mobile agent
- Thin client
  Mobile client
  - Kết nối, nhận biết bối cảnh, tương tranh, ..

2.2. Kiến trúc không tập trung

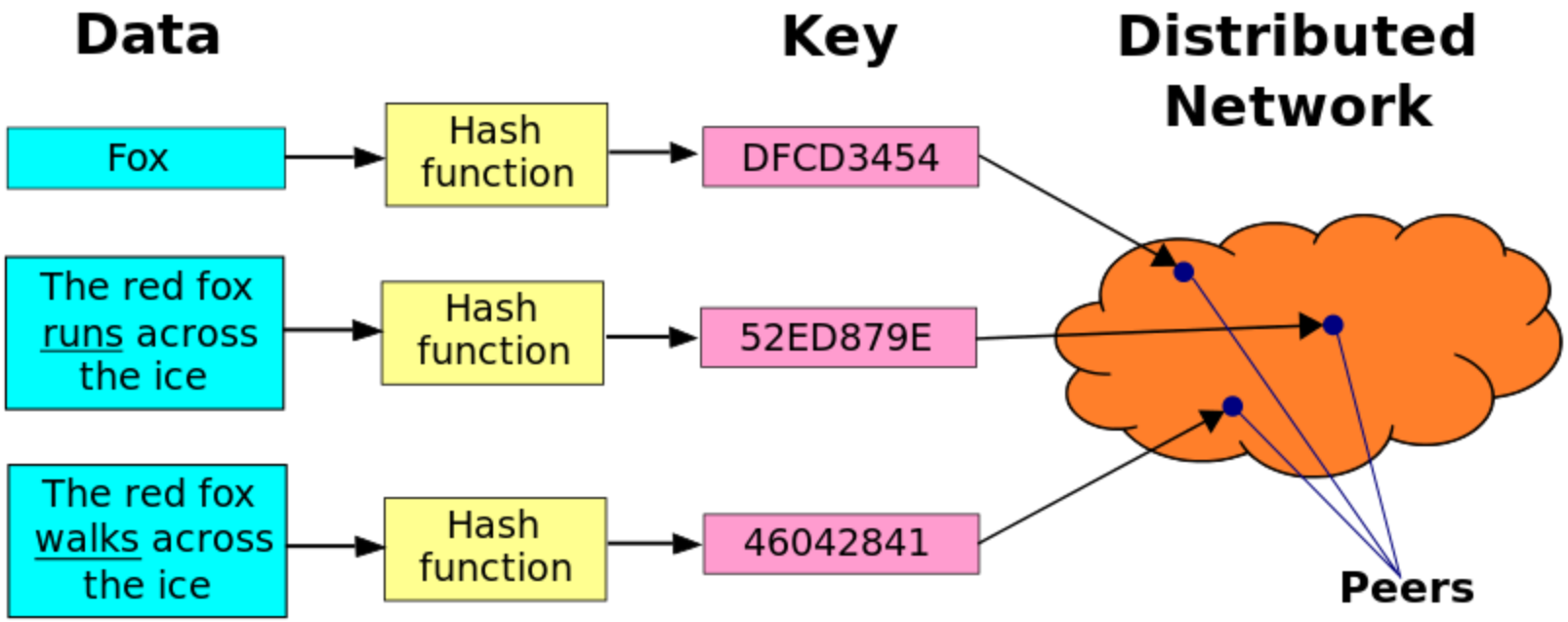
- Client và server không phân biệt vai trò
- Kết nối với nhau bằng một mạng trên mạng hạ tầng (Overlay network)
- Có cấu trúc/Không có cấu trúc
- P2P thuần/P2P hỗn hợp

Overlay network

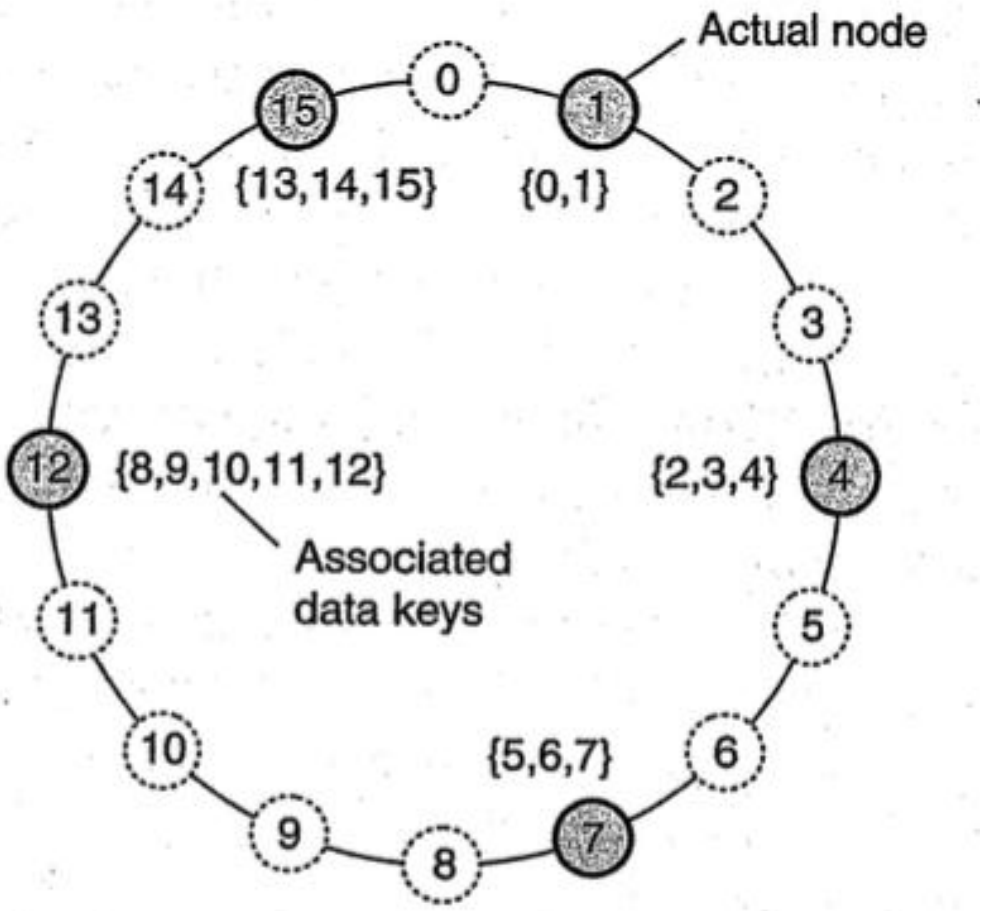


2.2.1. Kiến trúc P2P có cấu trúc

- Mạng overlay được xây dựng dựa trên 1 thủ tục định trước
- DHT (Distributed Hash Table)

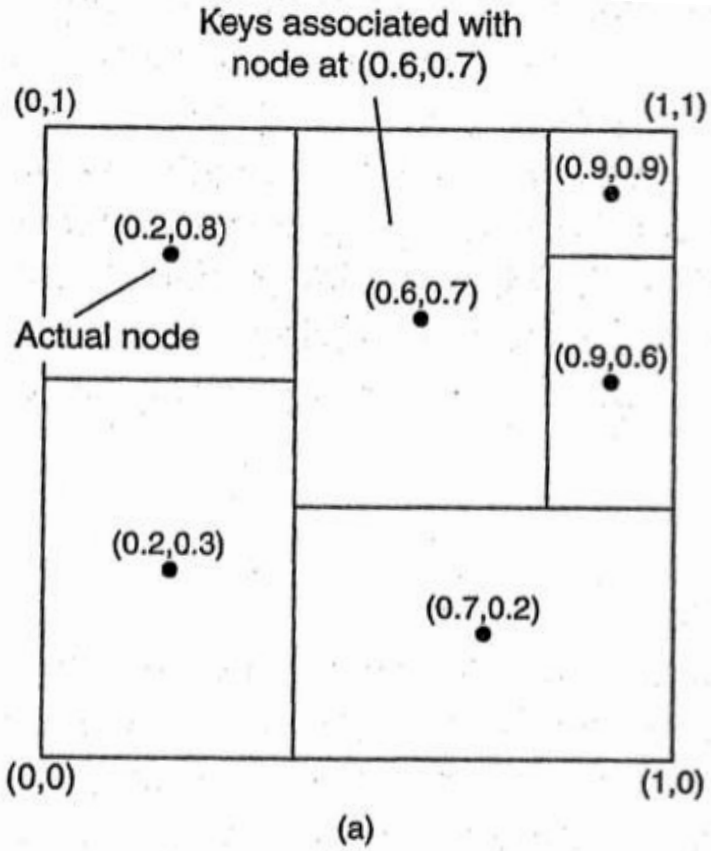


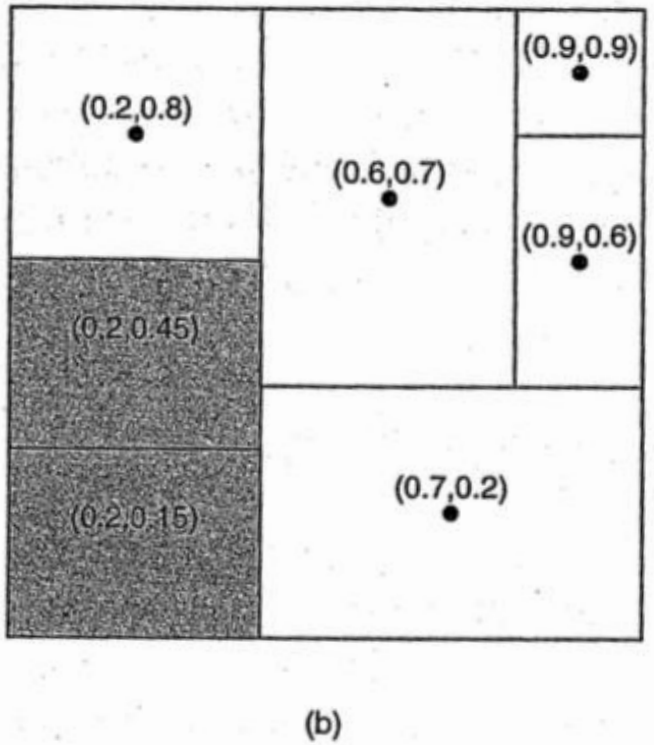
Hệ thống Chord



- Mạng dạng vòng
- Succ(k)
- Hàm LOOKUP(k)
- Một node muốn join hệ thống
- Một node muốn rời hệ thống

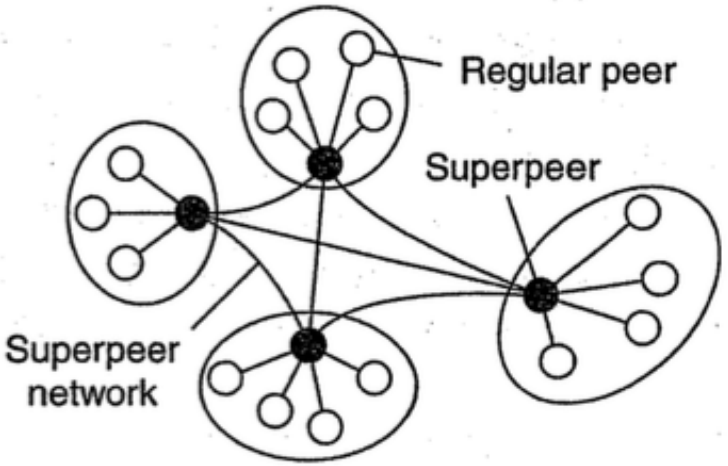
Hệ thống CAN (Content Addressable Network)





2.2.2. Kiến trúc P2P không có cấu trúc

- Thuật toán ngẫu nhiên để xây dựng mạng overlay (random graph).
- Mỗi node duy trì một danh sách hàng xóm (partial view).
- Dữ liệu được đưa vào hệ thống 1 cách ngẫu nhiên
- => Mỗi lần cần lấy dữ liệu ra, cần thực hiện duyệt toàn bộ hệ thống (flooding)
- =>superpeers



2.3. Kiến trúc hỗn hợp

- Hệ thống máy chủ biên (edge-server system)
- Hệ phân tán hợp tác

Hệ thống máy chủ biên

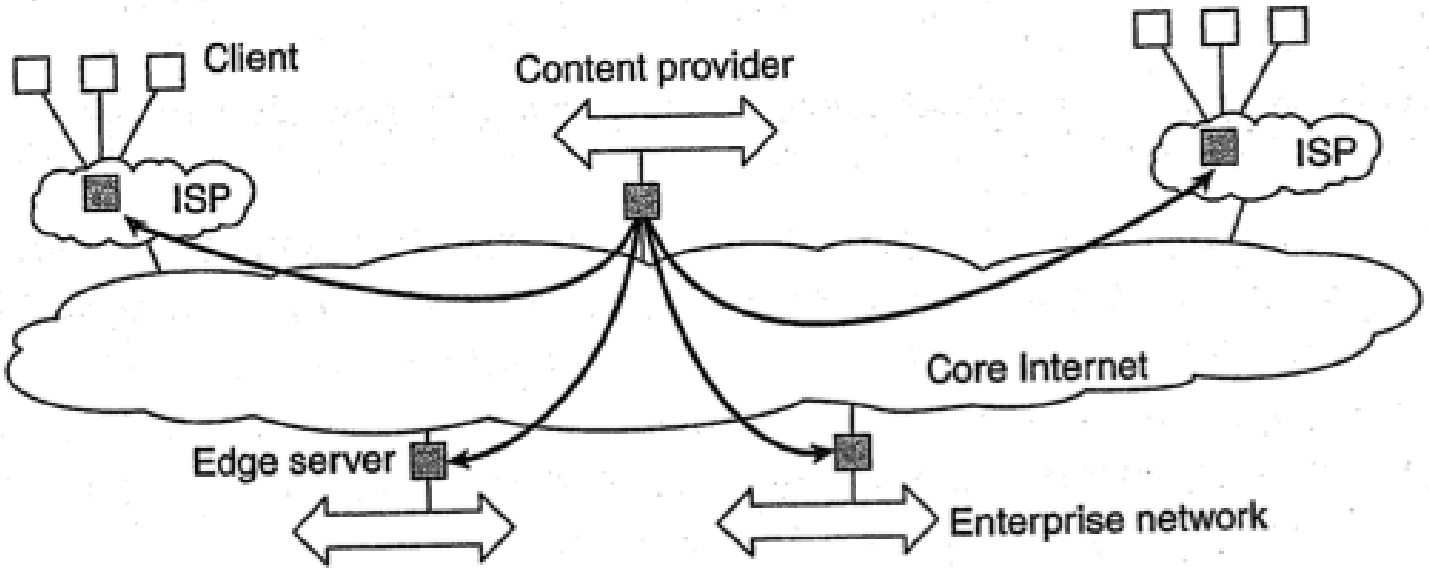
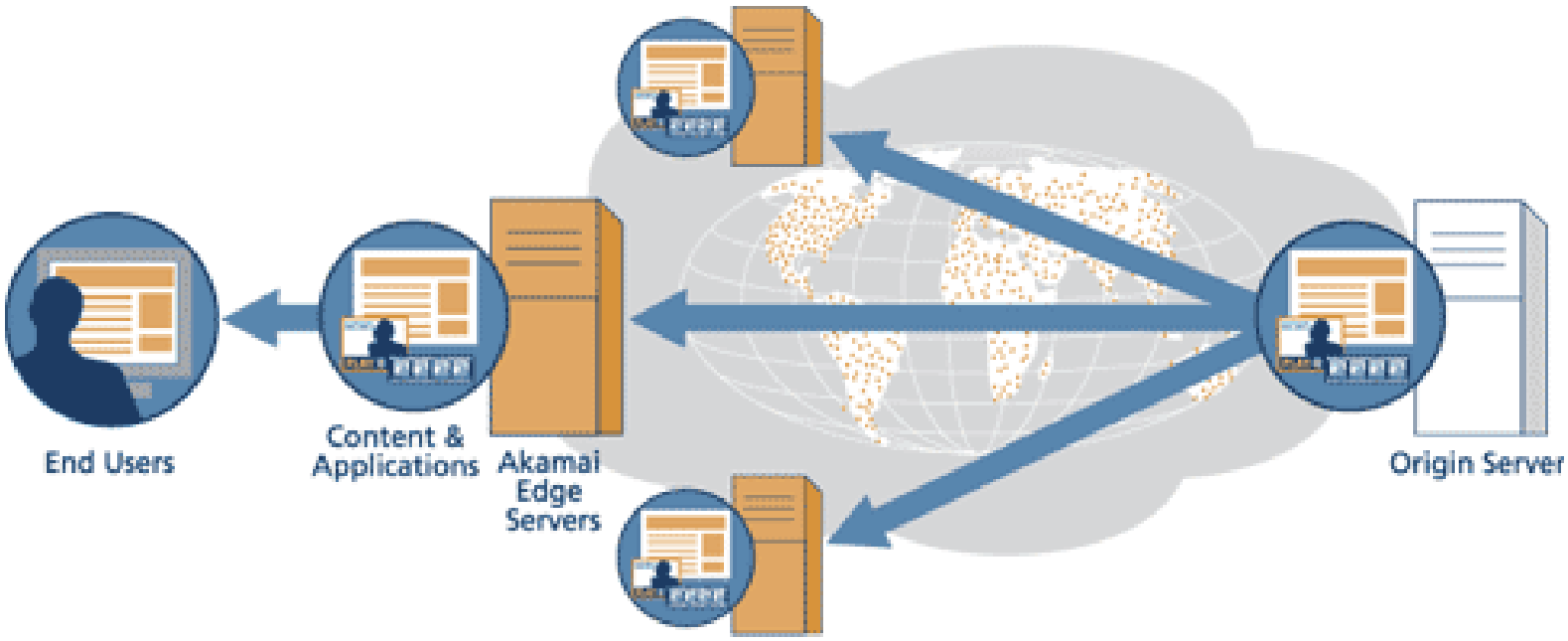
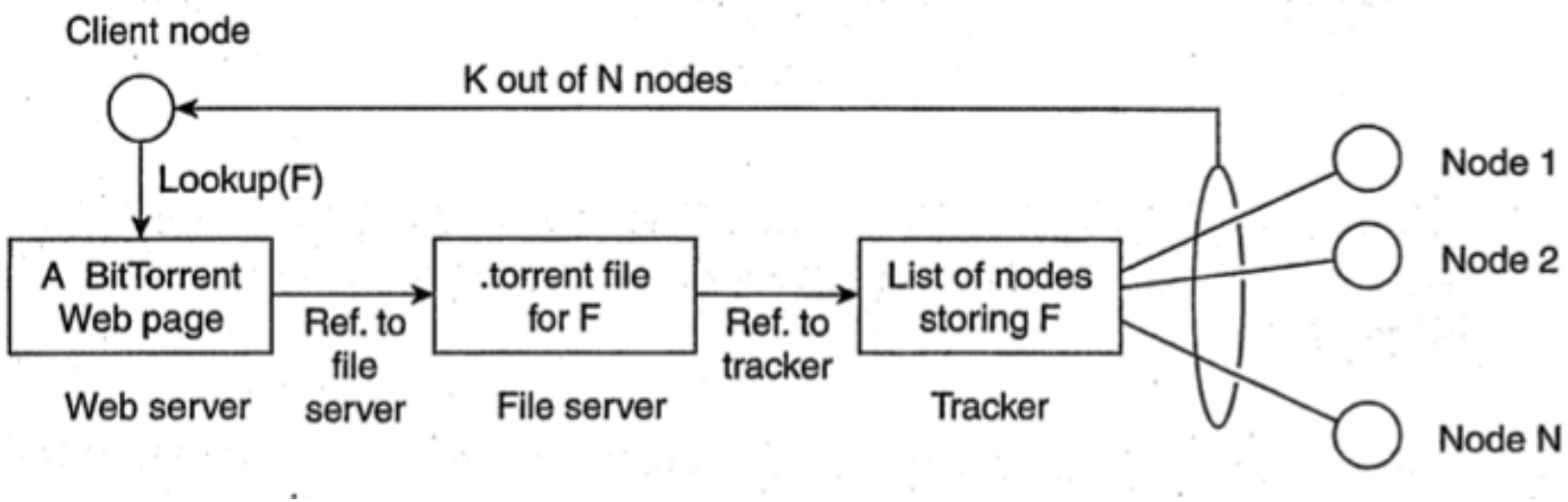


Figure 2-13. Viewing the Internet as consisting of a collection of edge servers.

VD: Content Delivery Network

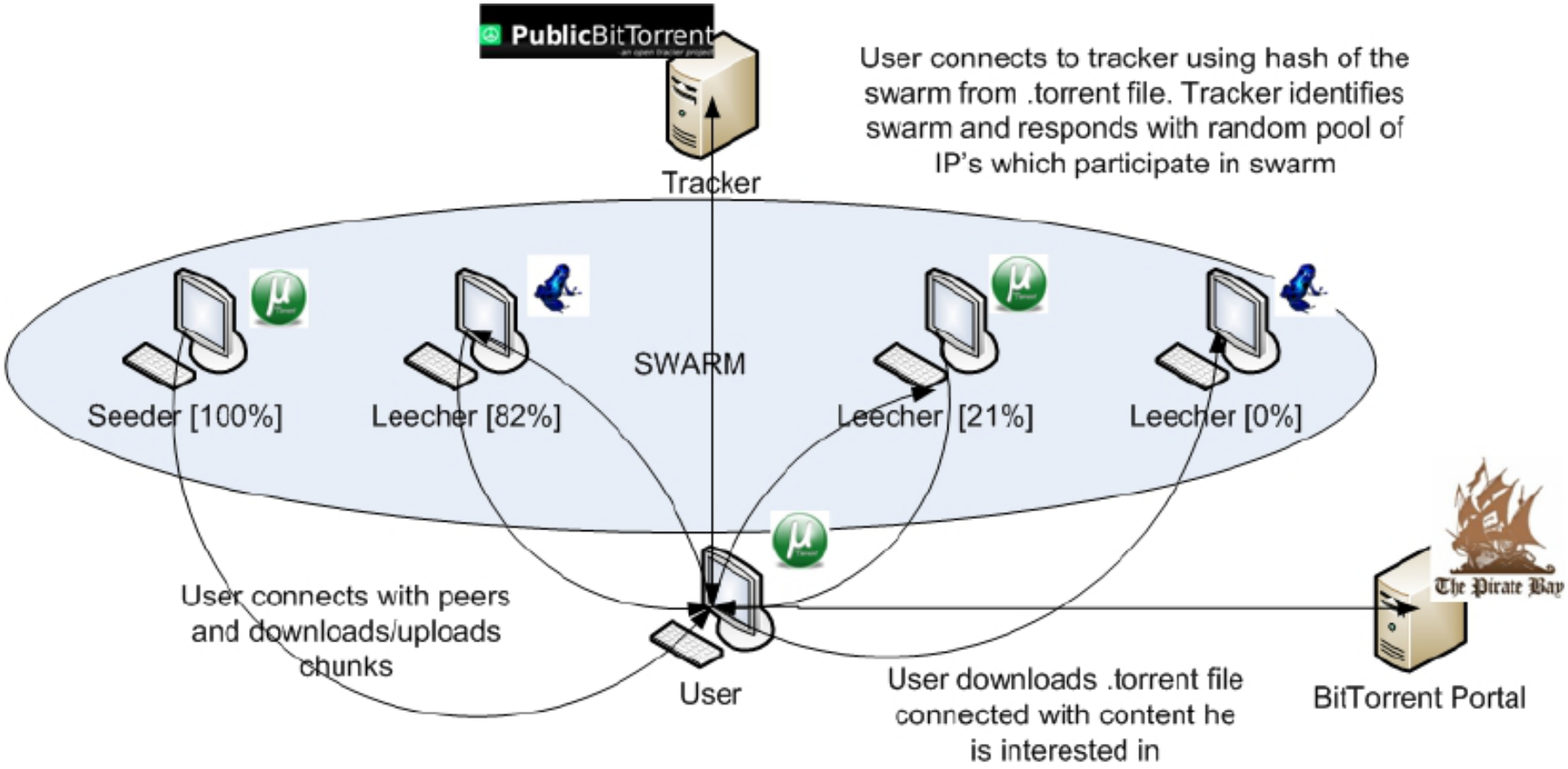


Hệ phân tán hợp tác



Hệ thống chia sẻ file BitTorrent

VD: Hệ thống BitTorrent



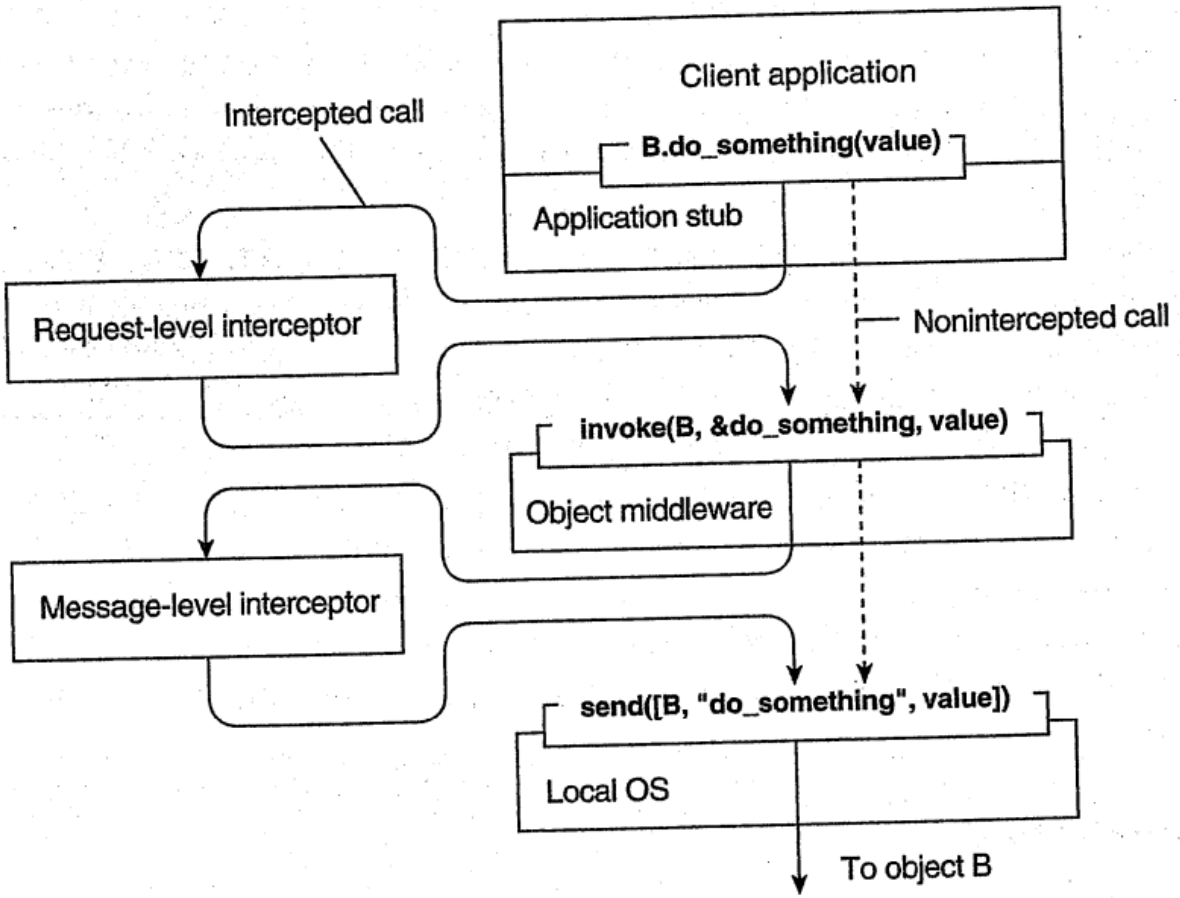
3. Middleware trong các kiến trúc

Các kiểu kiến trúc Middleware

- Vị trí của middleware
- VD: CORBA, TIB/Rendezvous
- Ưu điểm: dễ dàng hơn cho thiết kế ứng dụng.
- Nhược điểm: không tối ưu cho mỗi nhà phát triển ứng dụng.
- Giải pháp:
  - Sử dụng nhiều phiên bản khác nhau của middleware.
  - Tách biệt cơ chế và chính sách → dễ dàng cấu hình, thích nghi và tùy chỉnh.

Interceptors

- Cấu trúc phần mềm, cho phép chặn các dòng điều khiển thông thường, cho phép các đoạn mã khác được thực thi.

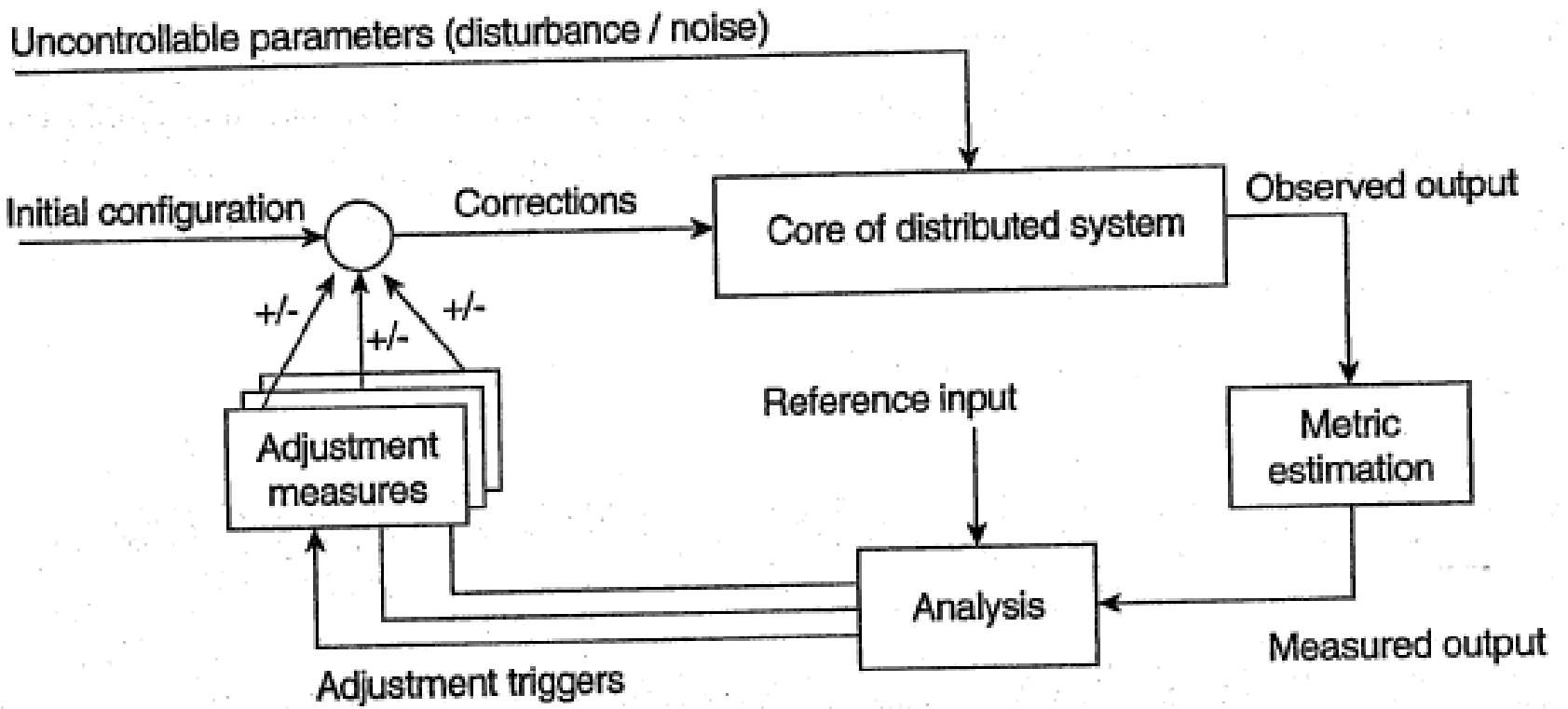


Những hướng tiếp cận chung cho phần mềm thích nghi

- Môi trường các ứng dụng phân tán luôn luôn thay đổi.
- "Phần mềm thích nghi" là yếu tố quan trọng trong thiết kế HPT.
- Các kỹ thuật:
  - Tách biệt các vấn đề
  - Phản ánh tính toán
  - Thiết kế dựa trên thành phần

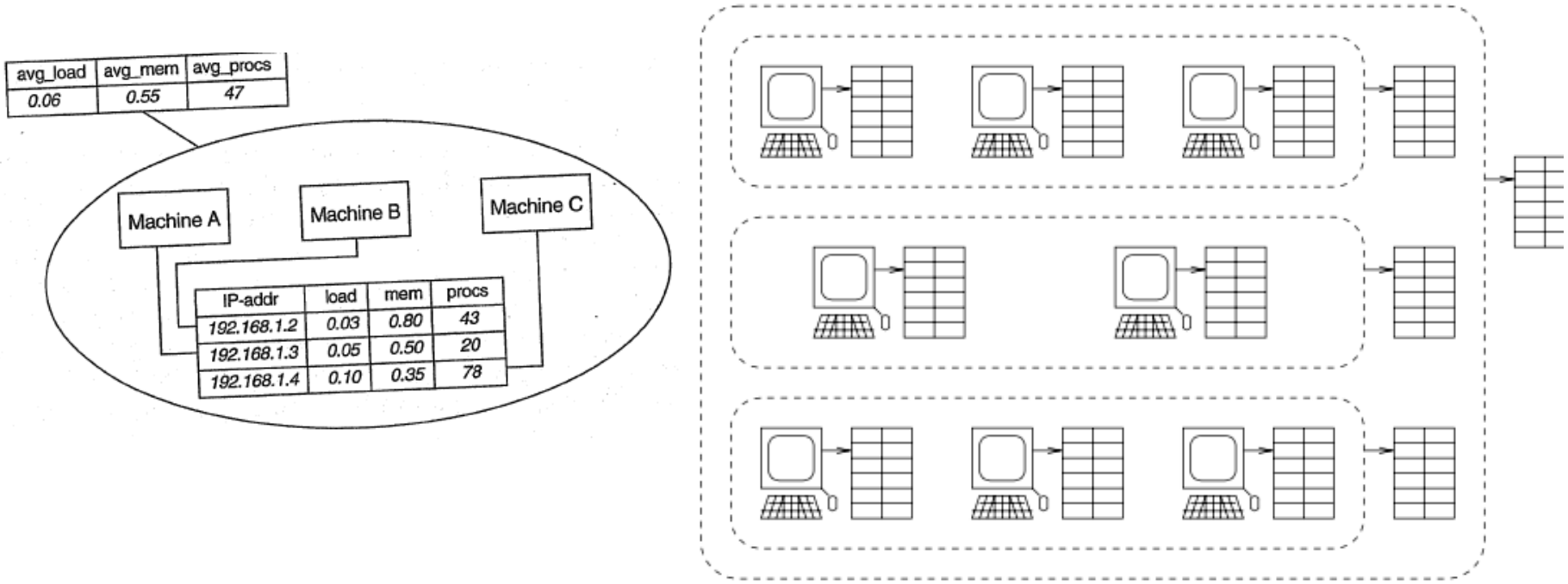
4. Quản lý tự động trong hệ phân tán

Mô hình điều khiển dựa trên phản hồi

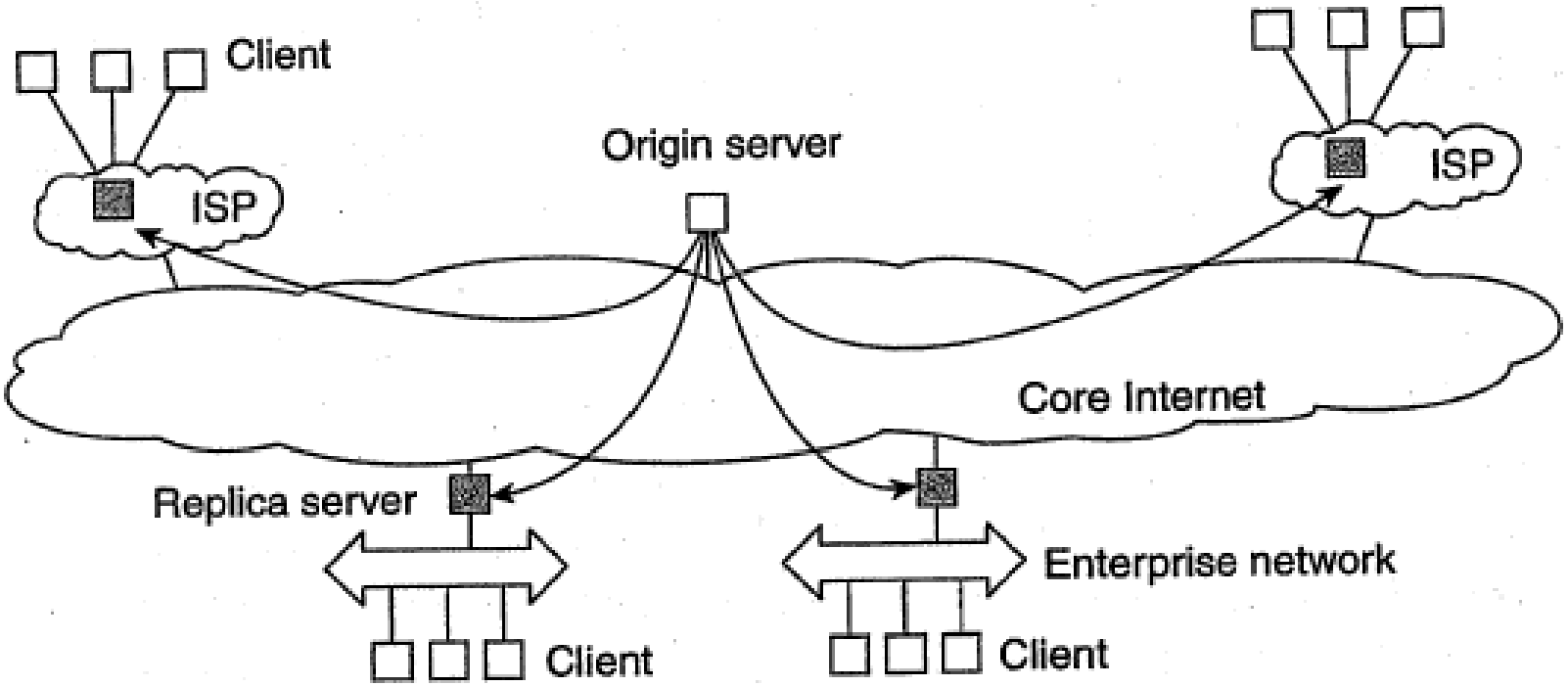


Tổ chức logic của hệ thống điều khiển dựa trên phản hồi

Ví dụ: Hệ thống giám sát Astrolabe



Ví dụ: Globule



$$cost = (w_1 \times m_1) + (w_2 \times m_2) + \cdots + (w_n \times m_n)$$

Ví dụ: hệ thống quản lý sửa chữa Jade

- Mục đích: phát hiện các nodes bị hỏng và tự động thay thế.
- Mô hình Fractal
- Miền quản lý sửa chữa (*repair management domain*).
- Node manager
- Các bước sửa lỗi:
  - Tắt liên kết
  - Khởi động và thêm node mới vào domain
  - Cấu hình lại node mới
  - Thiết lập lại liên kết với các node trước