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 và các kiểu kiến trúc
2. Kiến trúc hệ thống
3. Middleware trong các kiến trúc

1. Khái niệm kiến trúc

1.1. Kiến trúc

- Xem xét tổ chức của một Hệ Phân Tán → tách biệt giữa *tổ chức logic* và *thực thi vật lý*.
- Tổ chức logic: các thành phần phần mềm, cách thức kết nối, kiểu dữ liệu trao đổi → **Kiến trúc phần mềm**
- Thực thi vật lý: cách thức xếp đặt/cài đặt các thành phần phần mềm lên các thiết bị vật lý → **Kiến trúc hệ thống**

1.2. Các kiểu 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
- Kiến trúc Microservices

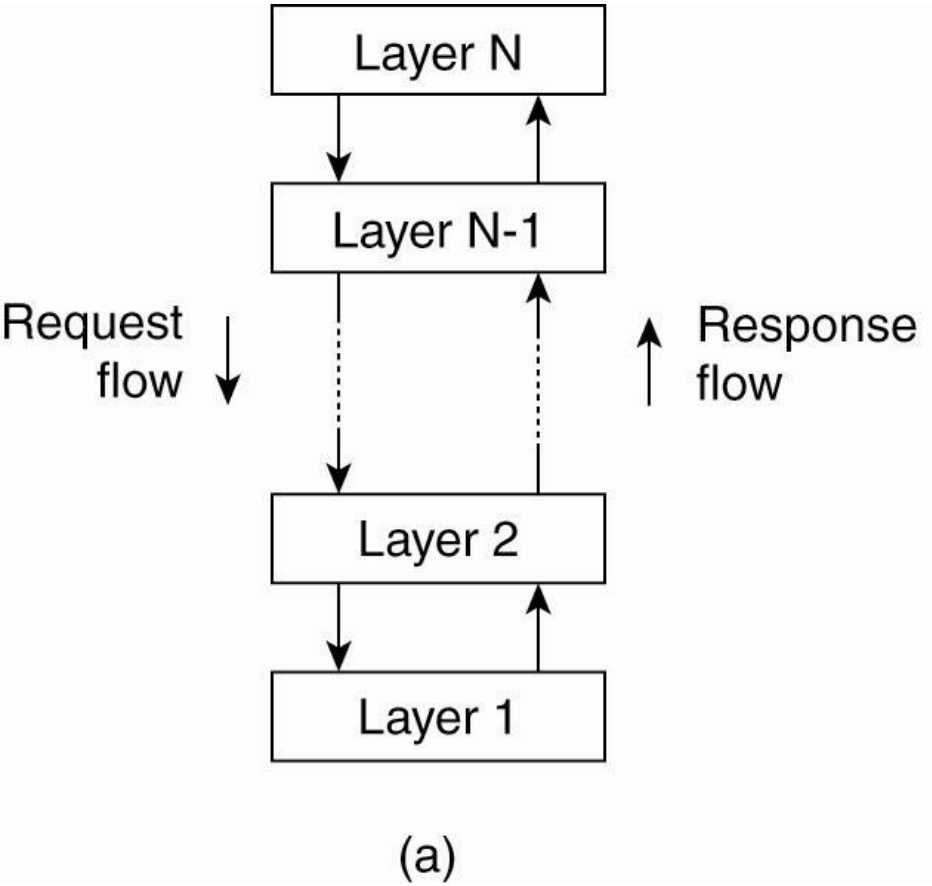
1.2.1. Kiến trúc phân tầng

- 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

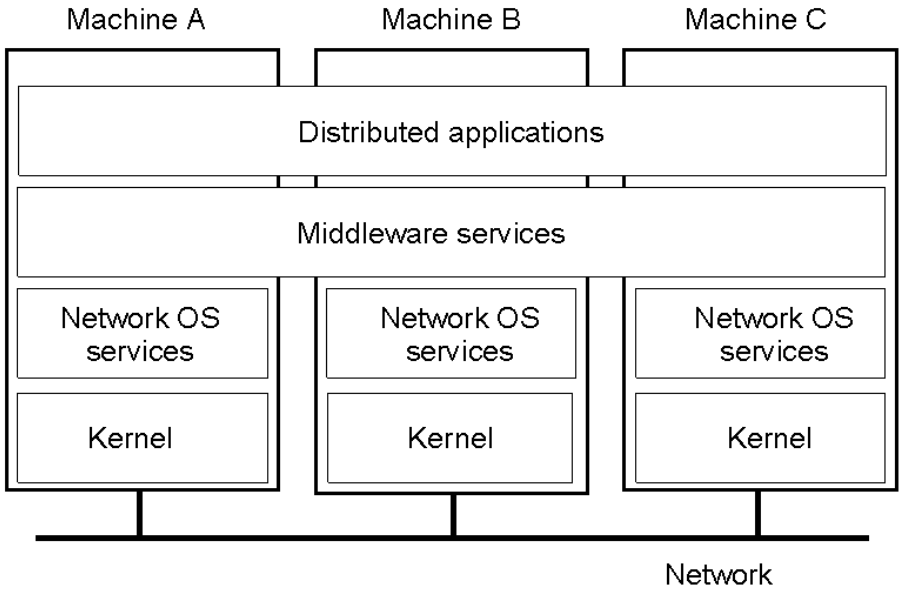
Tầng N

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



Các mô hình phân tầng thường gặp

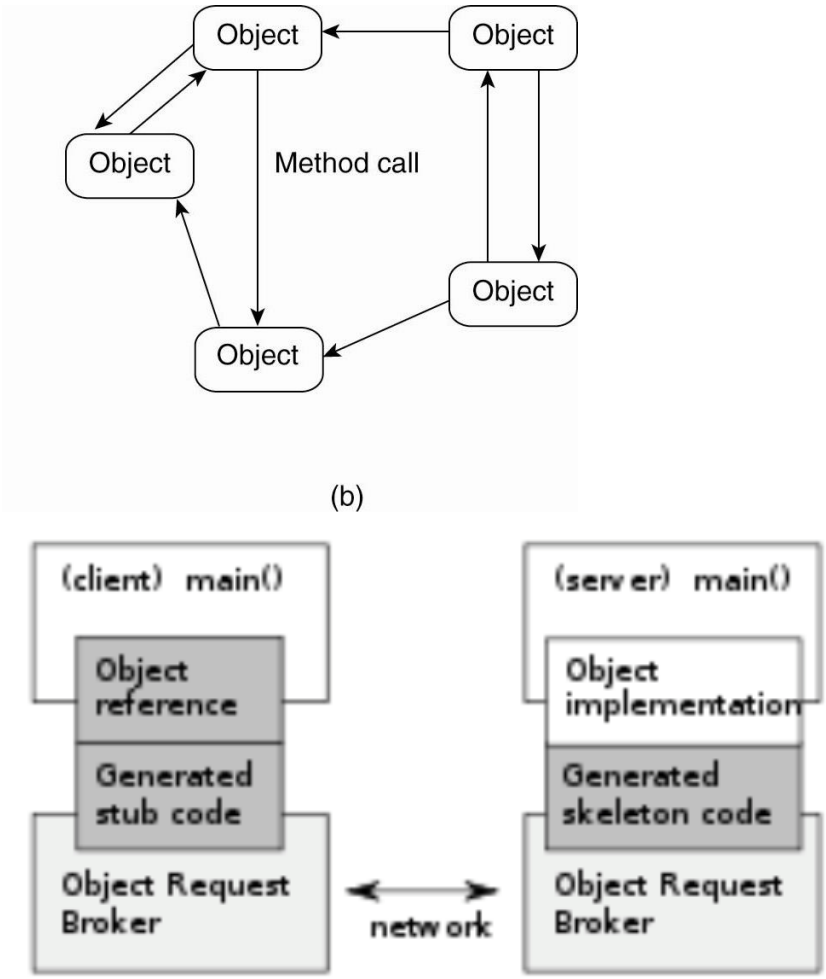
Mô hình OSI



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

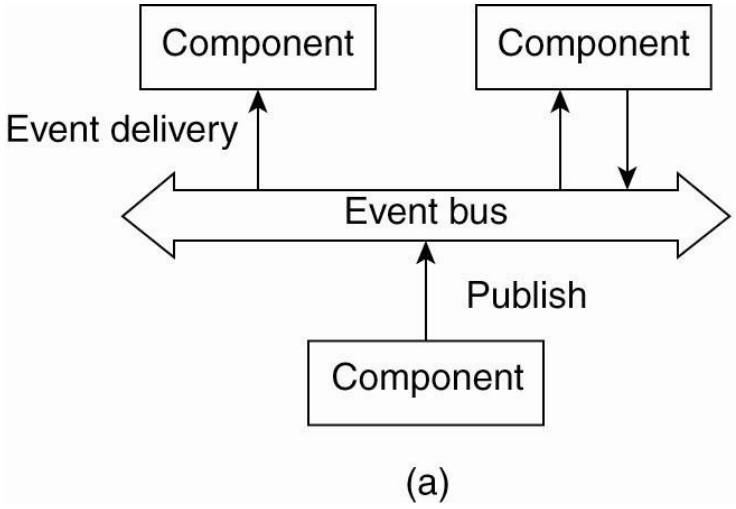


Ưu nhược điểm

- Ưu
  - Ánh xạ vào các đối tượng trong thế giới thật → dễ hiểu
  - Dễ dàng bảo trì và nâng cấp
  - Tính tái sử dụng (Polymorphism & Abstraction)
  - Kiểm soát lỗi
  - Mở rộng chức năng mà không ảnh hưởng hệ thống
  - Dễ dàng kiểm thử với encapsulation
  - Giảm thời gian và chi phí phát triển
- Nhược
  - Khó khăn trong việc xác định các lớp, các đối tượng
  - Kích cỡ chương trình lớn
  - Chương trình chạy chậm hơn (so với procedure programs)
  - Không phải phù hợp với mọi bài toán

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



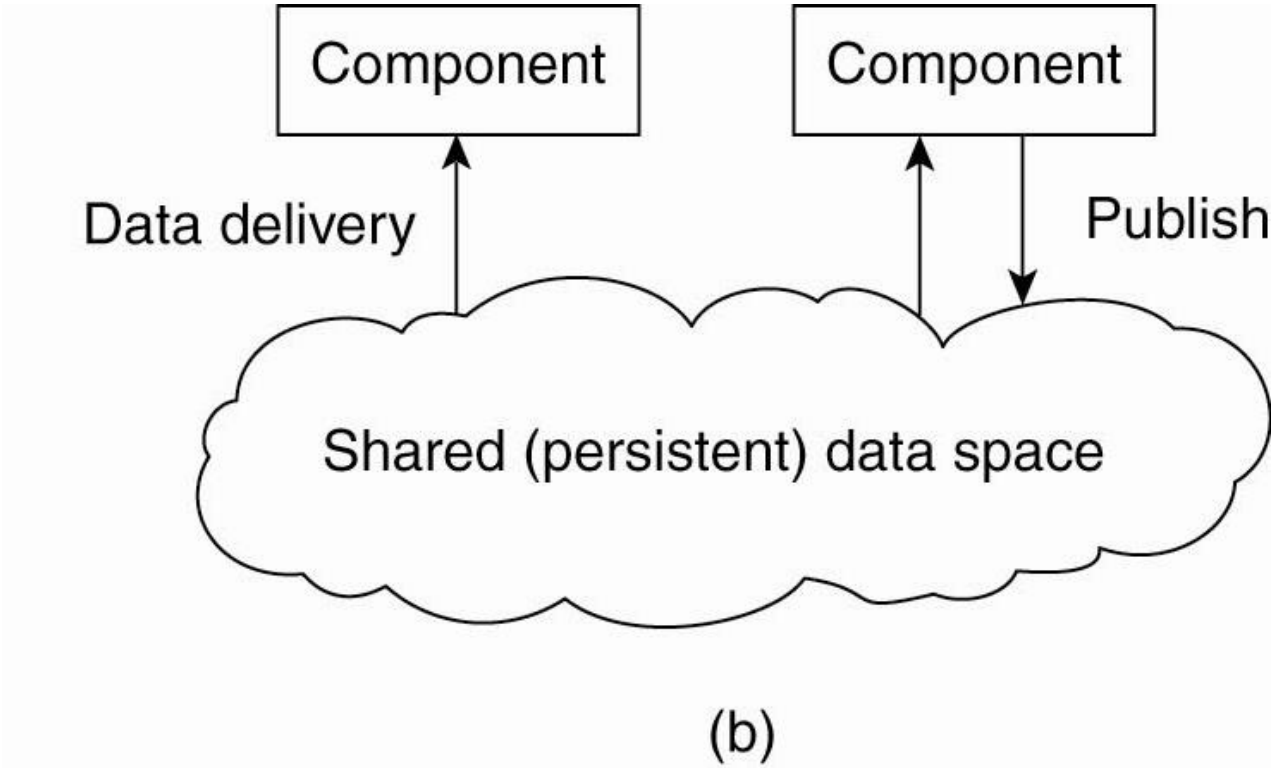
(a)

Ưu điểm của DDS so với JMS

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

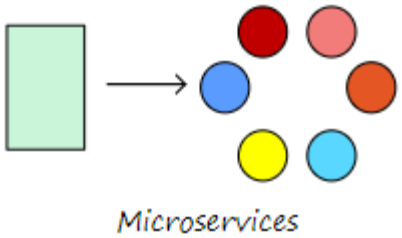
1.2.4. Kiến trúc hướng dữ liệu

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

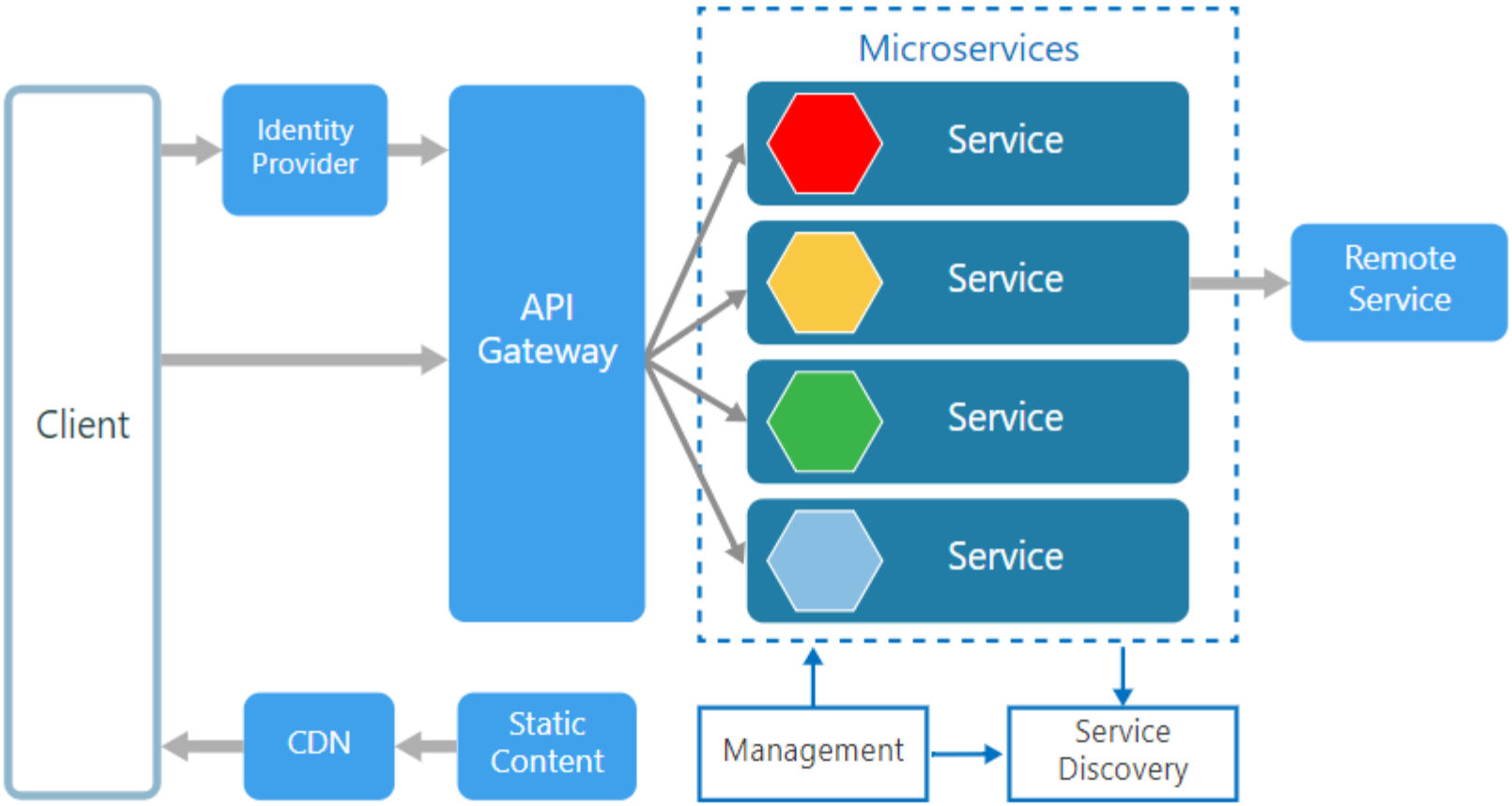


1.2.5. Microservices

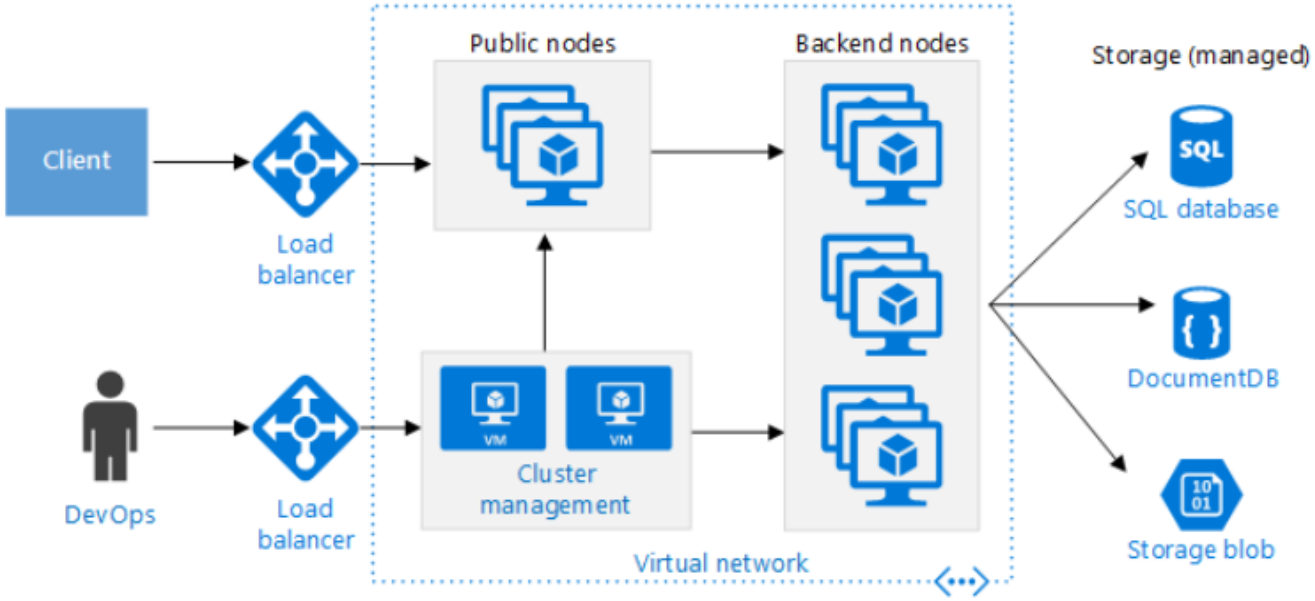
- Chuyển đổi monolithic → microservices
- Xây dựng ứng dụng dựa trên số lượng nhỏ các services, mỗi services chạy trên tiến trình riêng và hoàn toàn triển khai độc lập được.
- Ưu điểm:
  - Đơn giản triển khai
  - Đơn giản để hiểu
  - Tái sử dụng
  - Nhanh chóng cách ly thành phần hỏng
  - Giảm thiểu nguy cơ khi thực hiện thay đổi



Microservices



Microservices

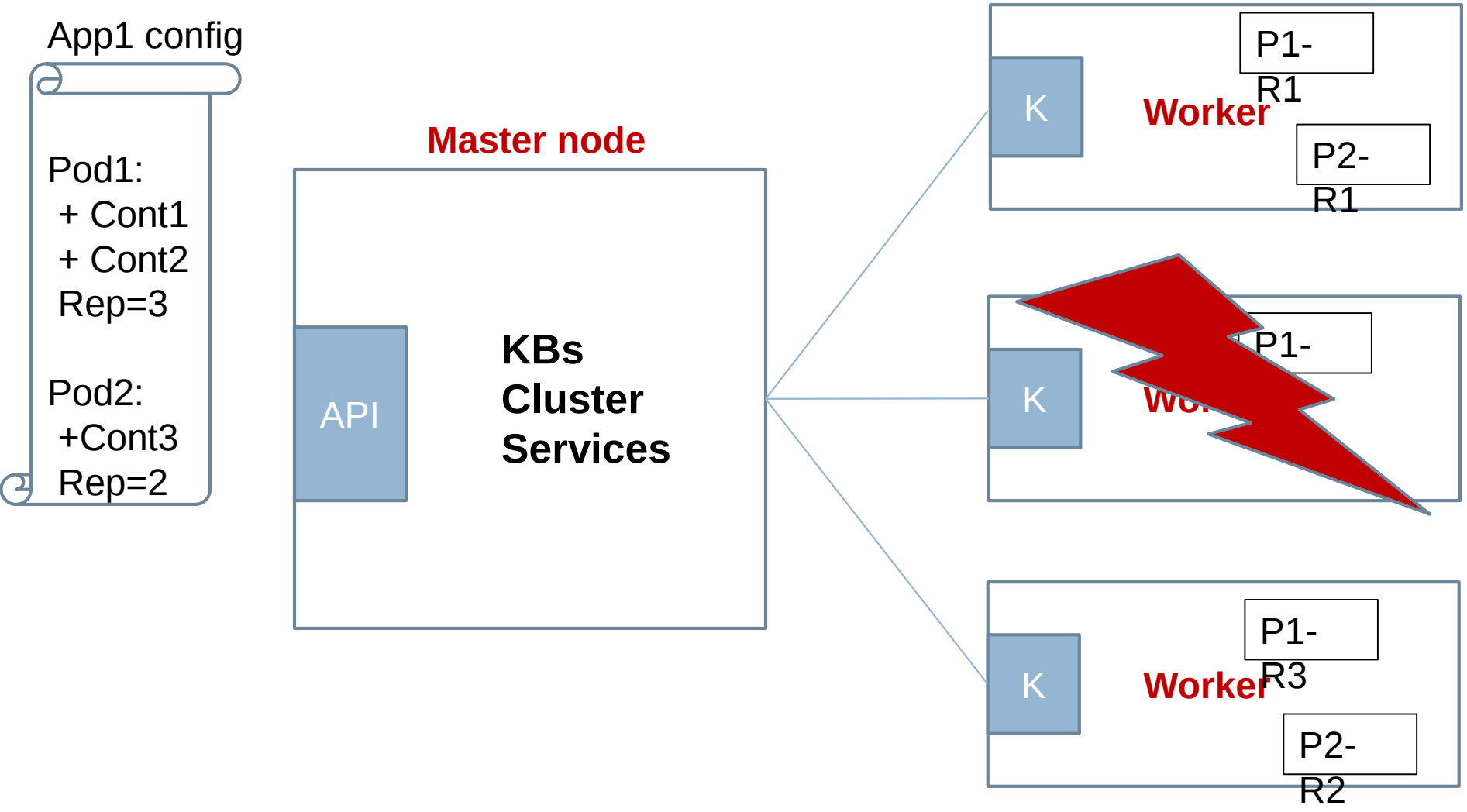


Vấn đề!!!

Container Orchestration tools

- Amazon ECS (EC2 Container Service)
- Azure Container Service (ACS)
- Cloud Foundry's Diego
- CoreOS Fleet
- Docker Swarm
- Kubernetes

Kubernetes


App1 config
Pod1:
+ Cont1
+ Cont2
Rep=3
Pod2:
+Cont3
Rep=2
Master node
API
KBs
Cluster
Services
K
Worker
P1-
R1
P2-
R1
K
P1-
K
P1-
R3
Worker
P2-
R2

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

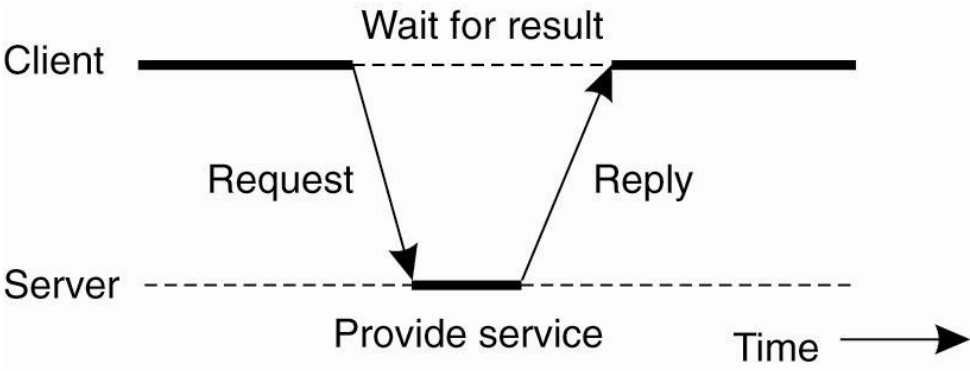
2.1.2. Phân tầng ứng dụng

2.1.3. Kiến trúc đa tầng

2.1.4. Software Agent

2.1.1. Kiến trúc client-server

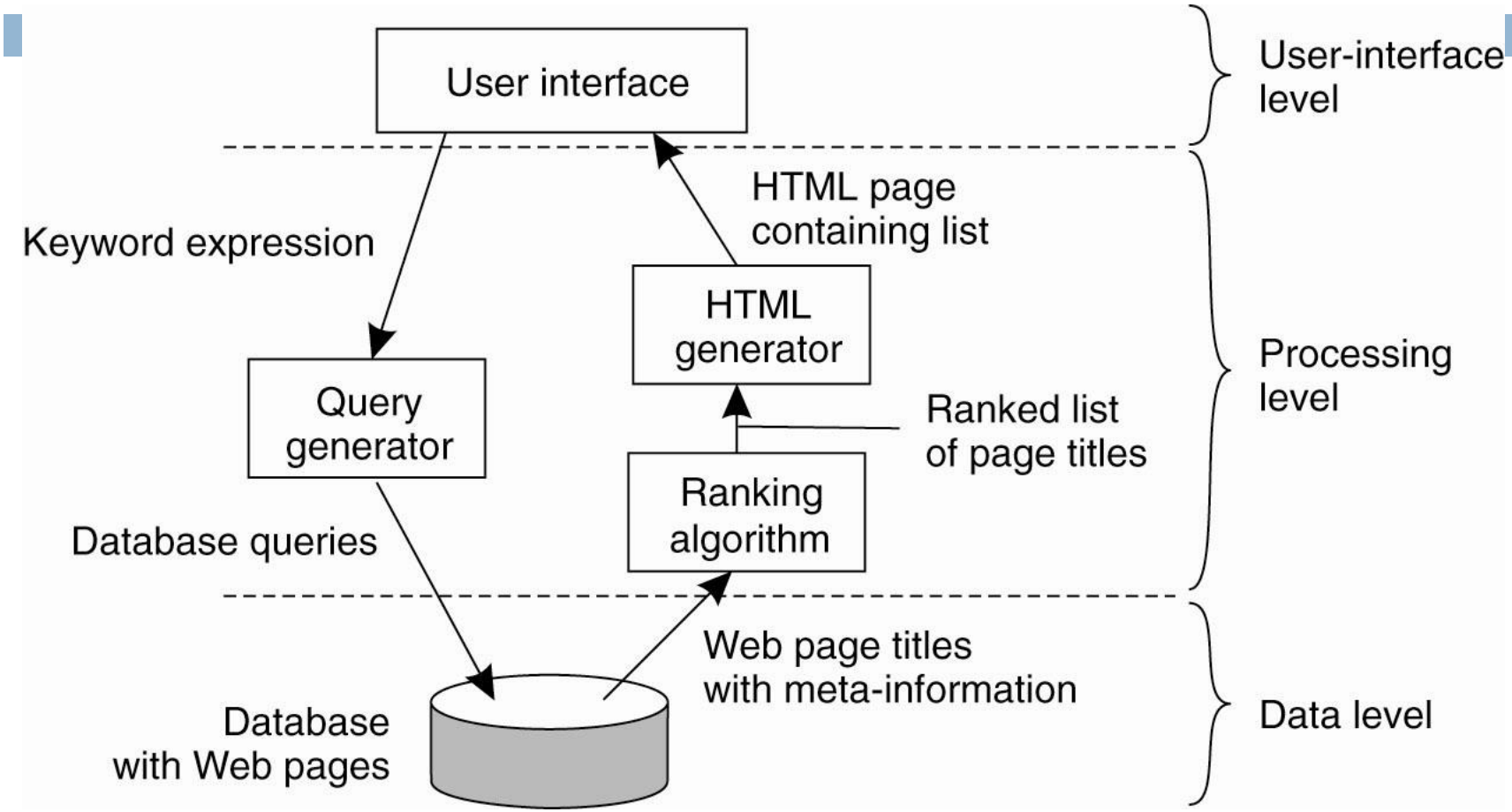
- Client:
  - gửi yêu cầu, nhận kết quả, hiển thị cho NSD
- Server:
  - lắng nghe, nhận yêu cầu, xử lý, trả lời
- Tương tác giữa client và server có thể là hướng kết nối hoặc không hướng kết nối
- Vấn đề
  - Đăng ký server (DNS hoặc dịch vụ thư mục)
  - Có thể lặp lại yêu cầu? (idempotent)
  - Có bộ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



2.1.2. Phân tầng ứng dụng

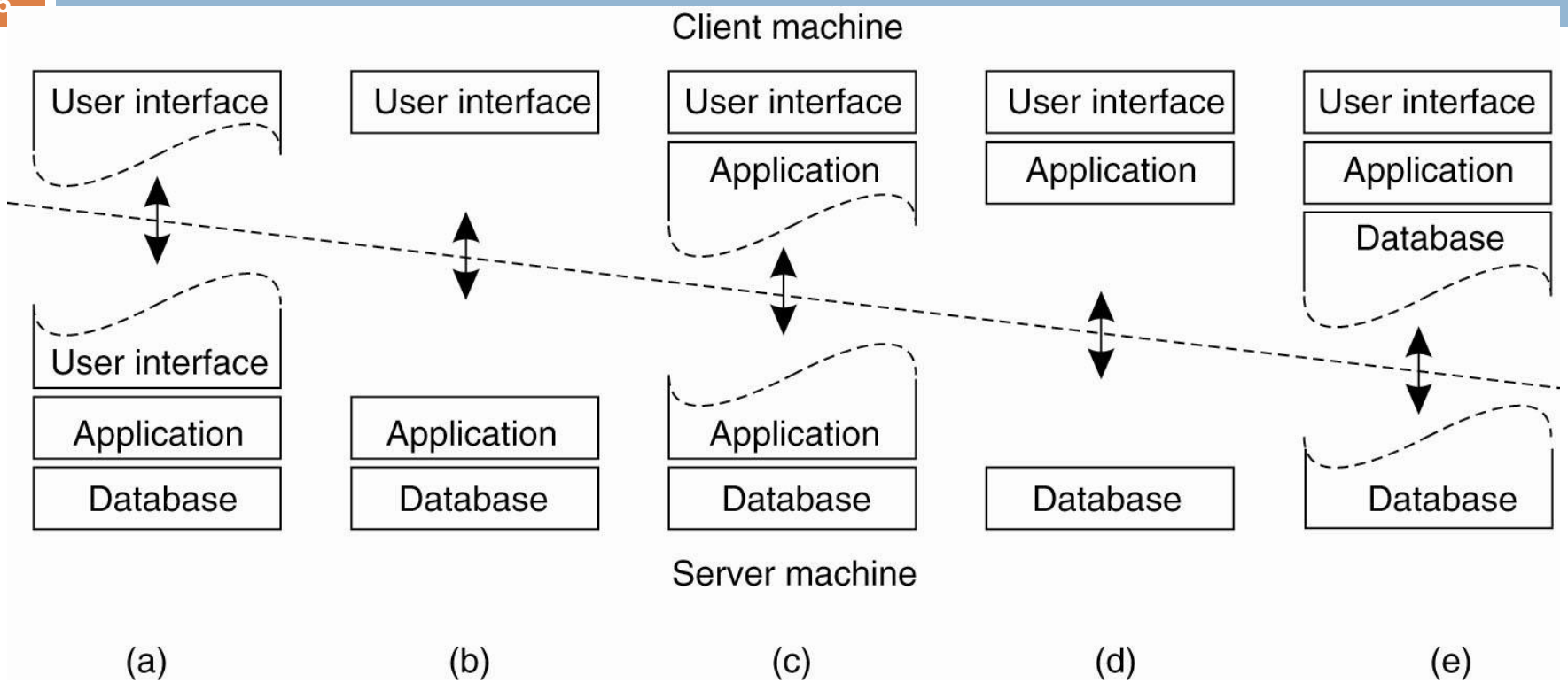
- Các mức phân tầng
  - Giao diện
  - Nghiệp vụ
  - Dữ liệu

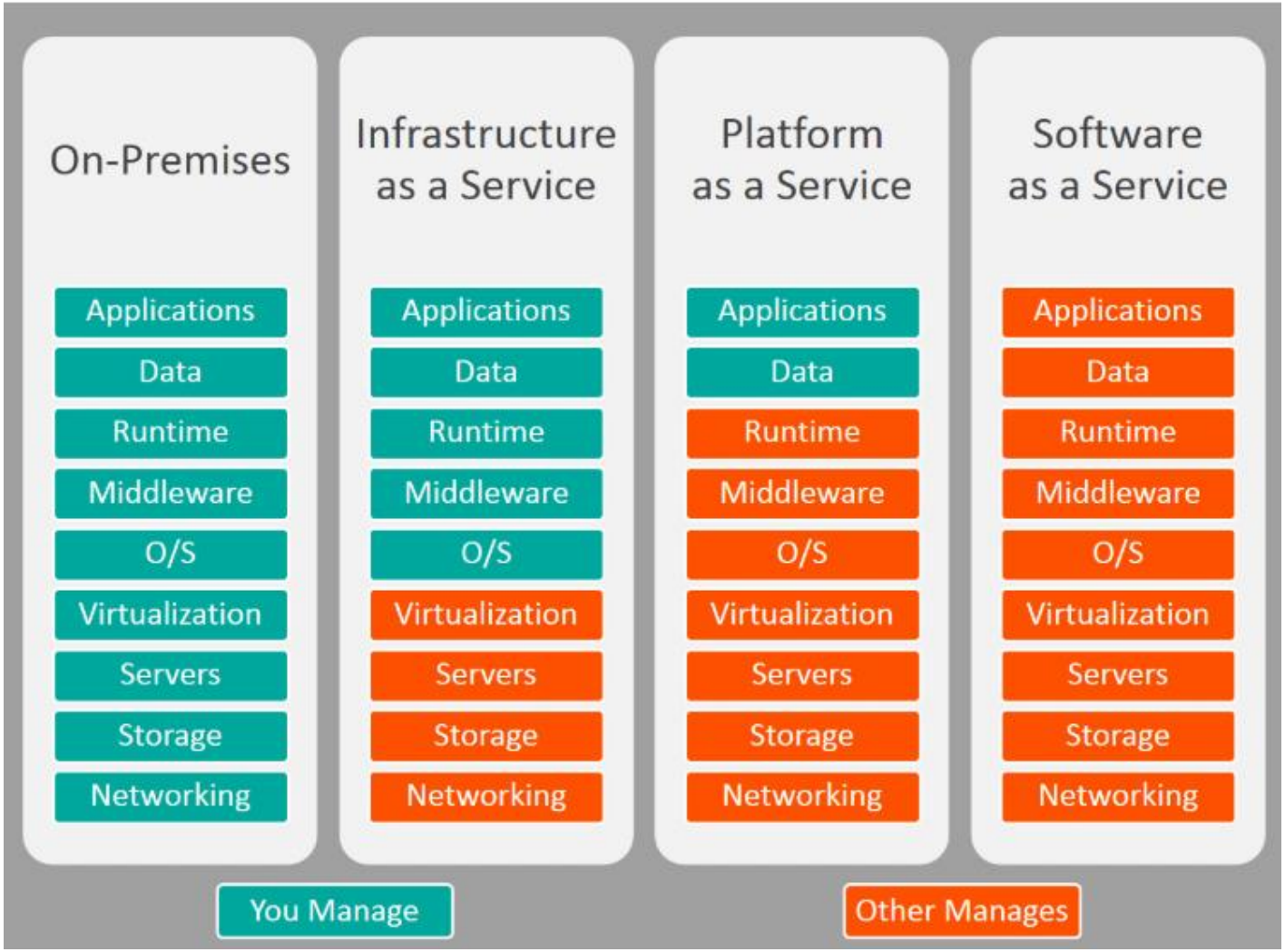
Phân tầng ứng dụng tìm kiếm



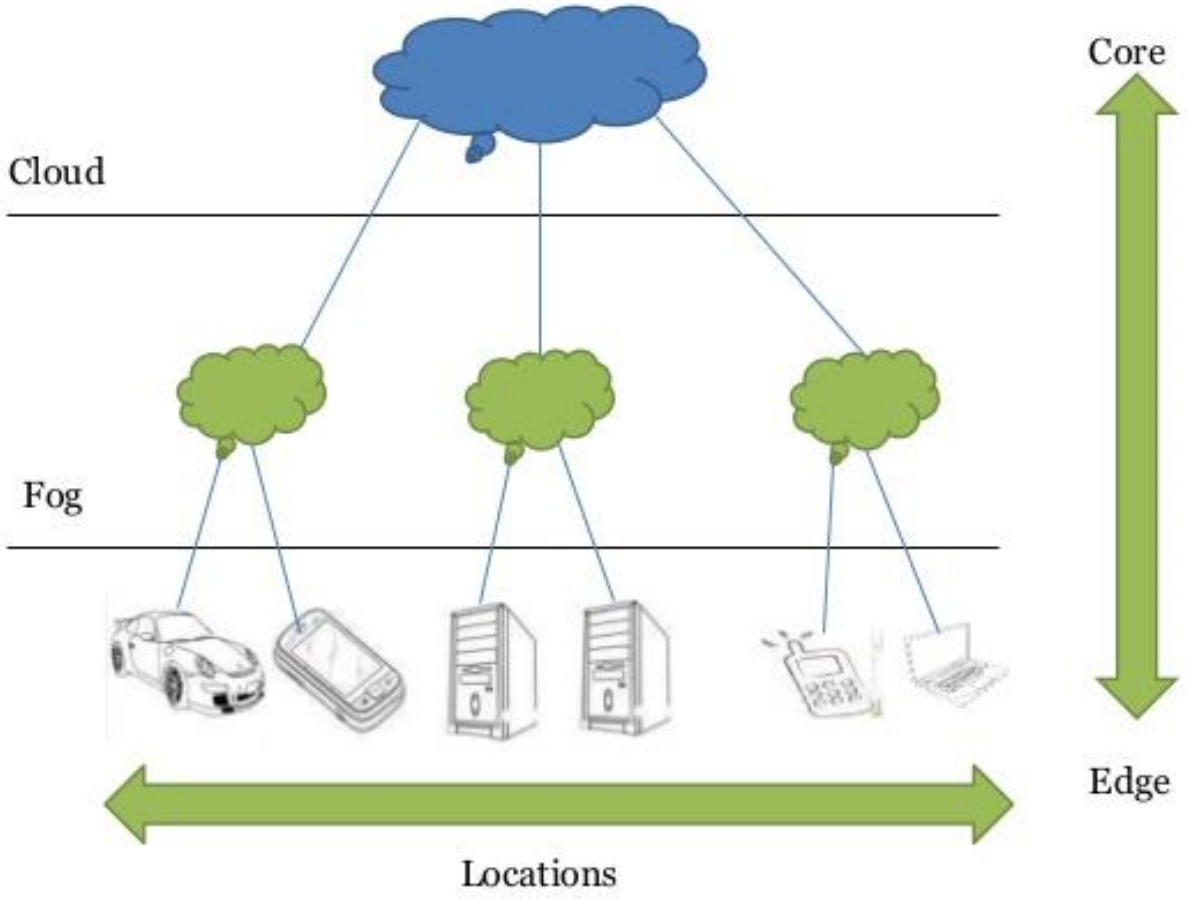
2.1.3. Kiến trúc đa tầng

Các mô hình 2 bên




On-Premises
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
Infrastructure as a Service
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
Platform as a Service
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
Software as a Service
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
You Manage
Other Manages

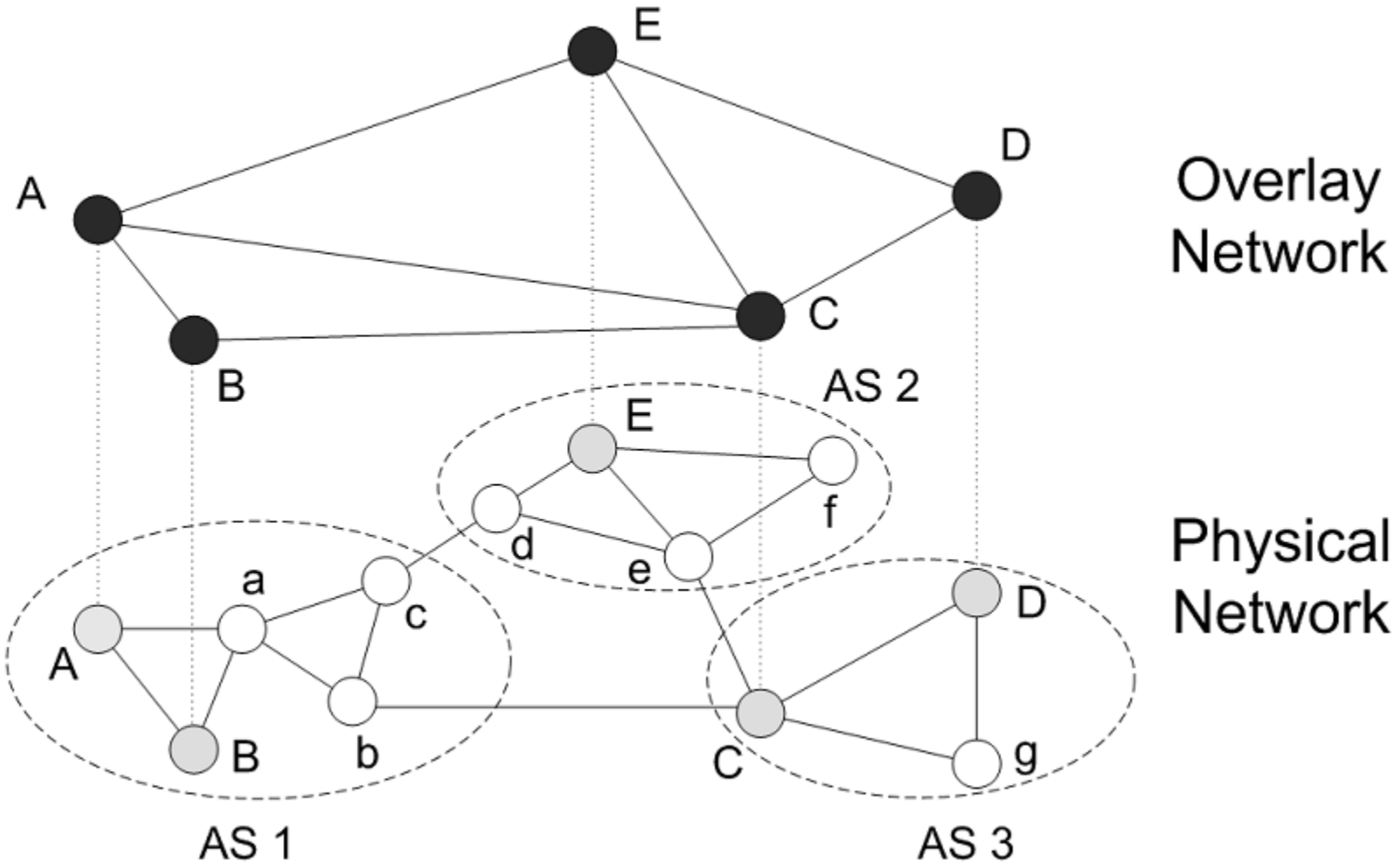
Cloud & Fog computing



2.2. Kiến trúc không tập trung

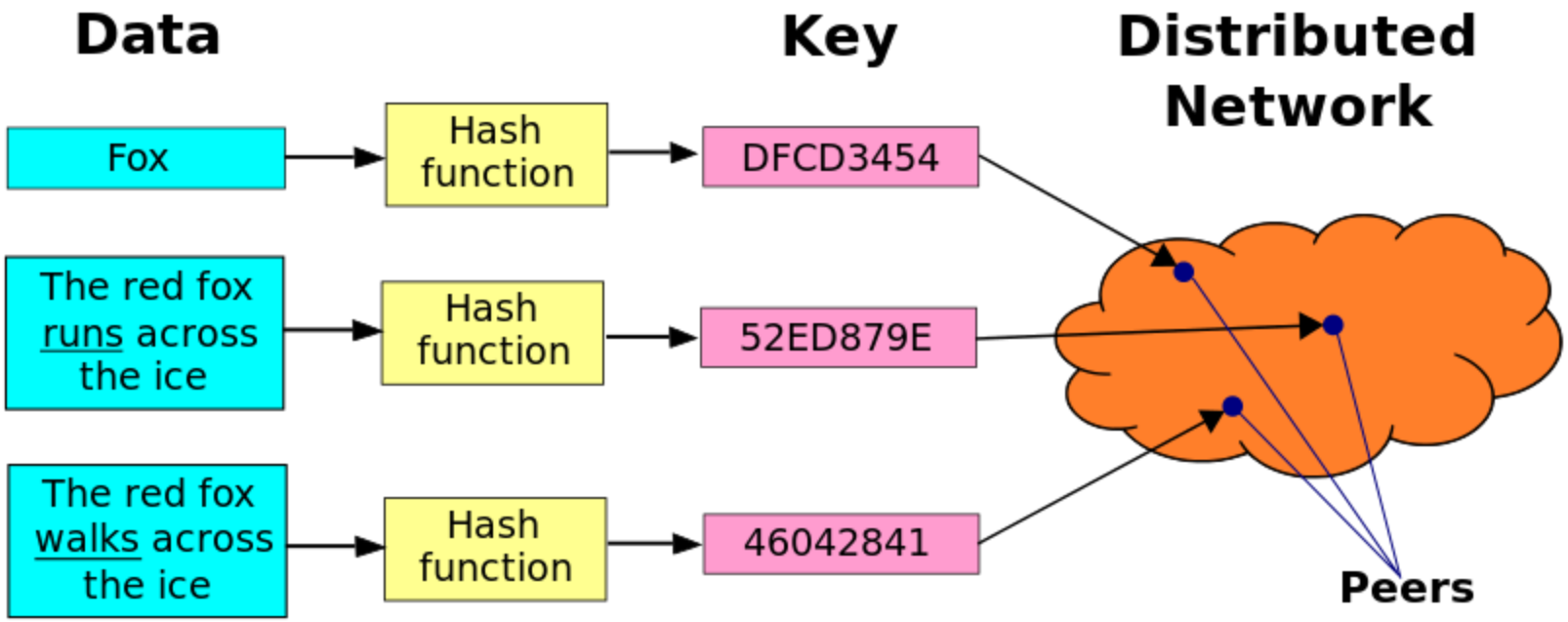
- Client và server không phân biệt vai trò
- Kết nối với nhau bằng một mạng trên mạng hạ tầng (Overlay network)
- Có cấu trúc/Không có cấu trúc
- P2P thuần/P2P hỗn hợp

Overlay network

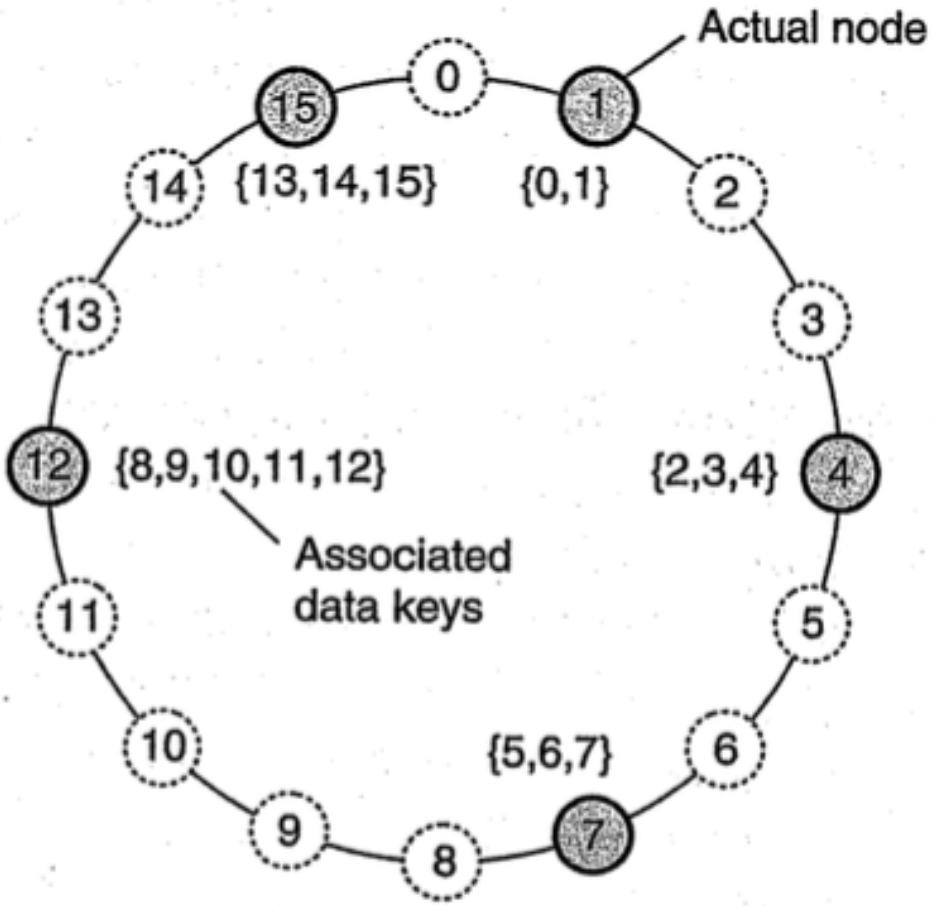


2.2.1. Kiến trúc P2P có cấu trúc

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

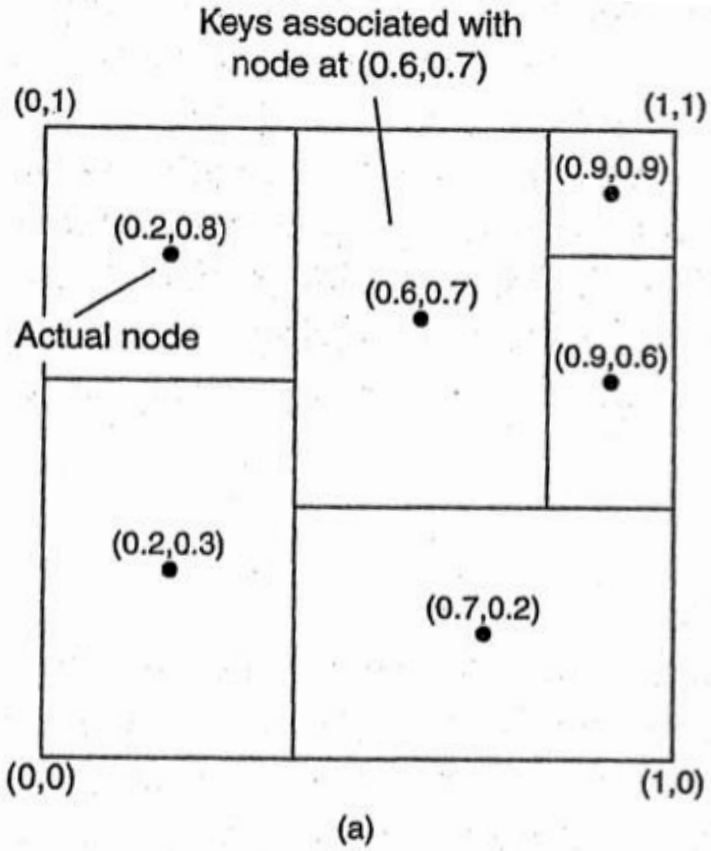


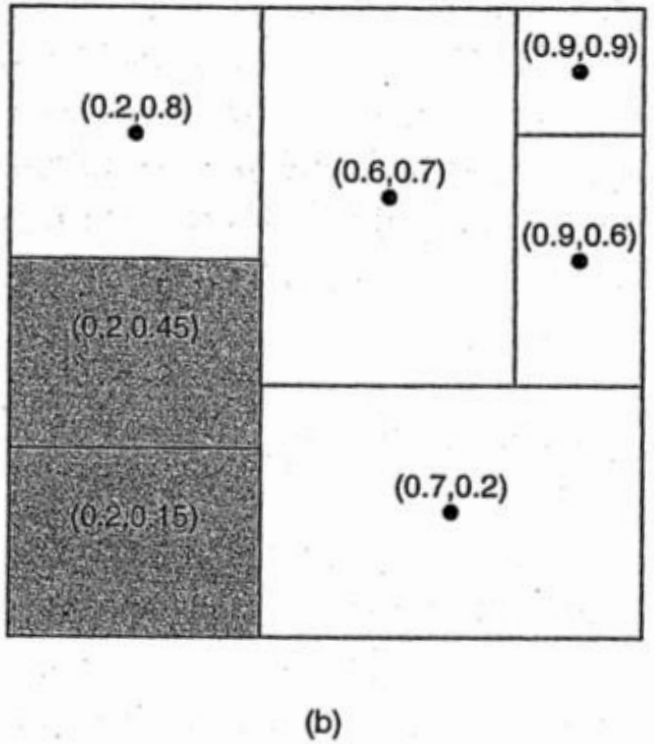
Hệ thống Chord



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

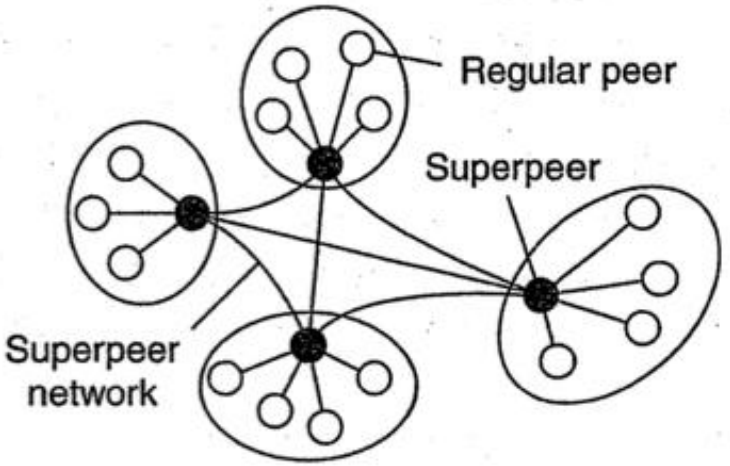
Hệ thống CAN (Content Addressable Network)





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

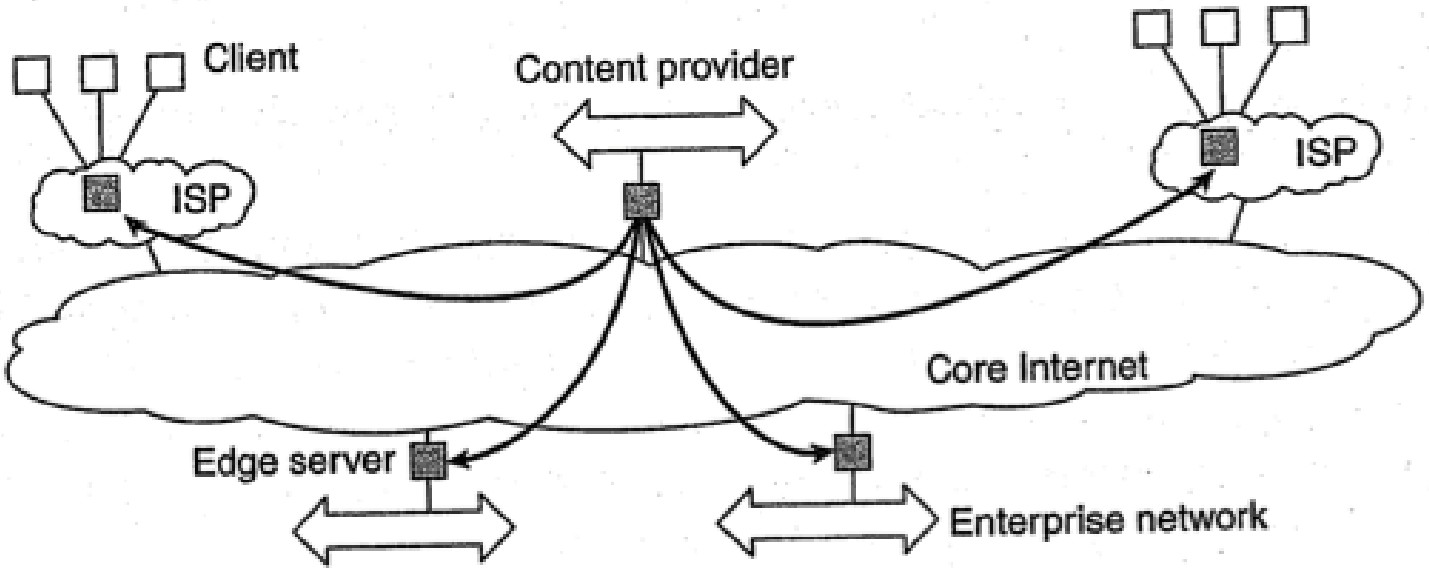
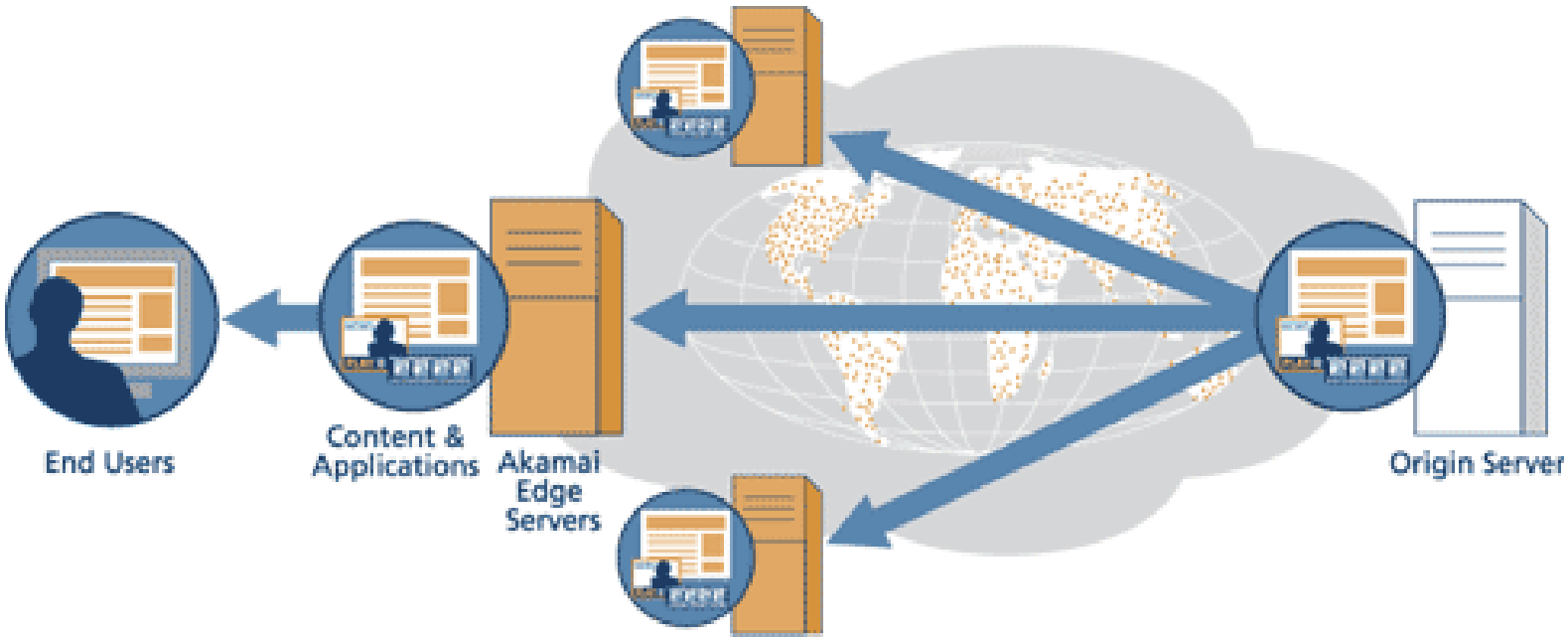
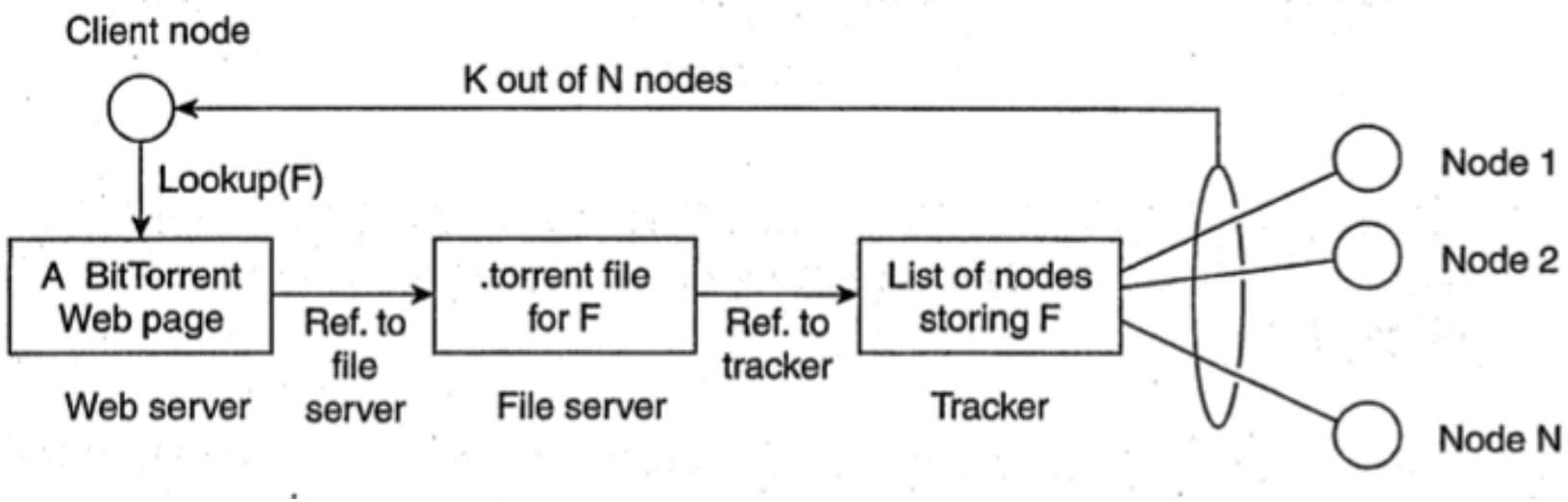


Figure 2-13. Viewing the Internet as consisting of a collection of edge servers.

VD: Content Delivery Network

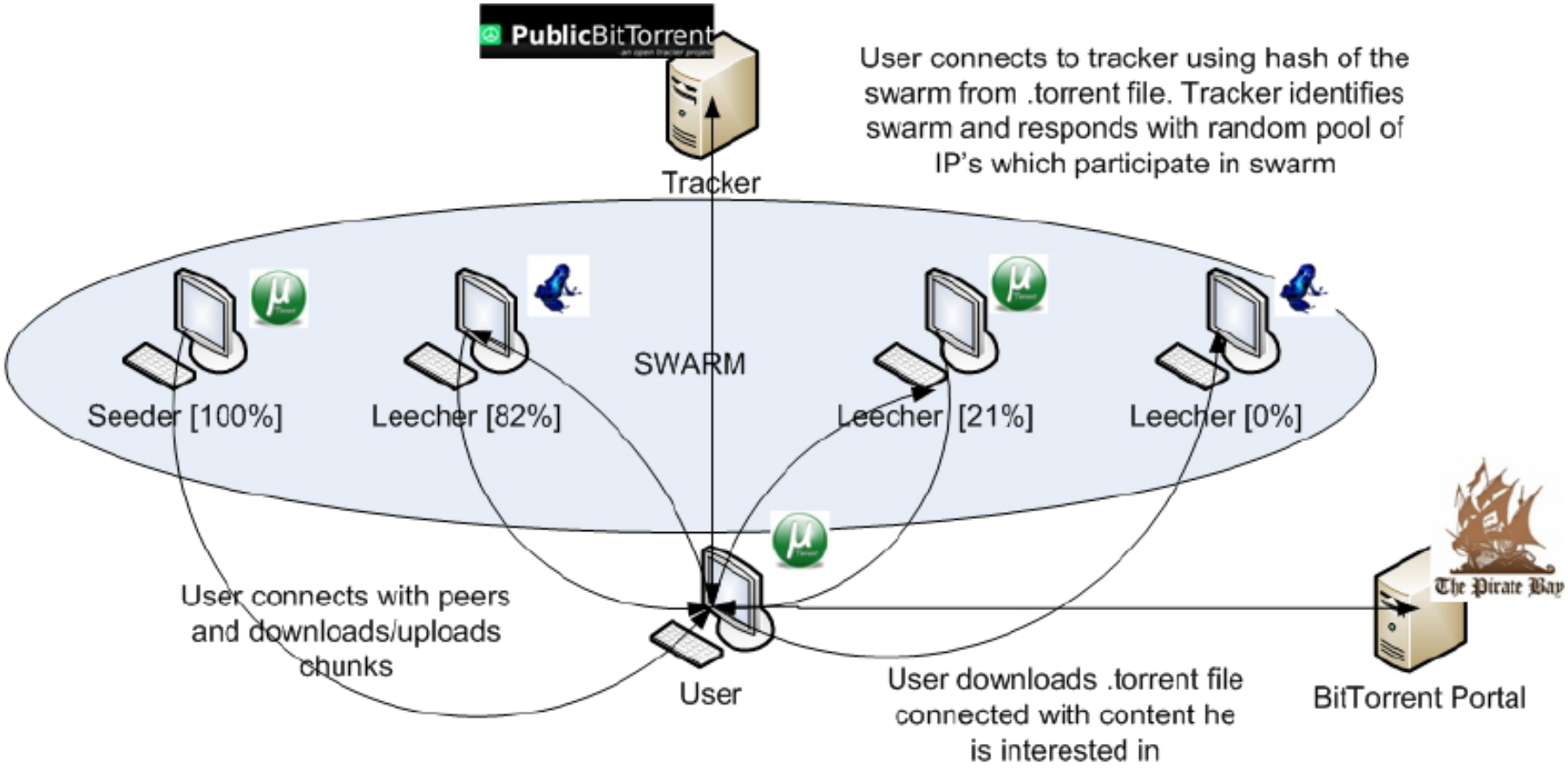


Hệ phân tán hợp tác



Hệ thống chia sẻ file BitTorrent

VD: Hệ thống BitTorrent



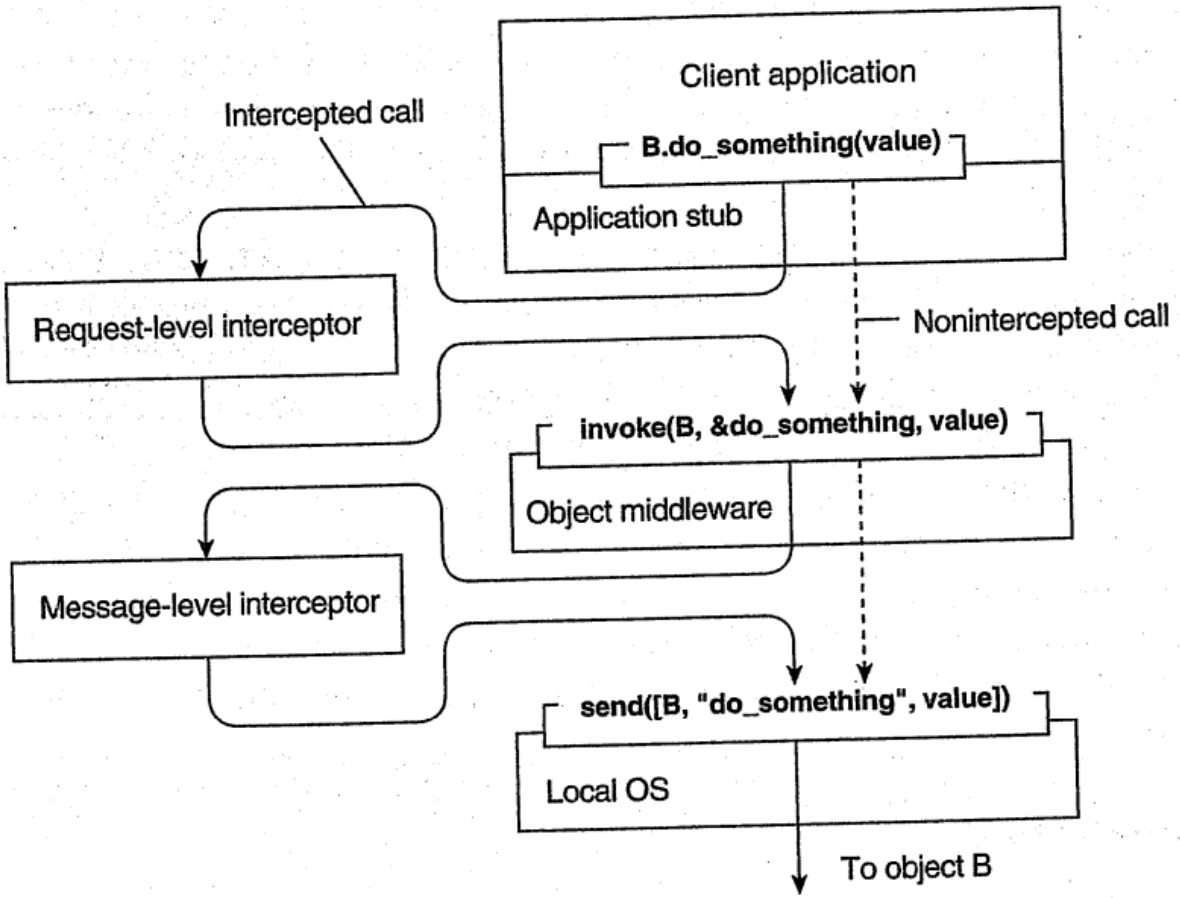
3. Middleware trong các kiến trúc

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