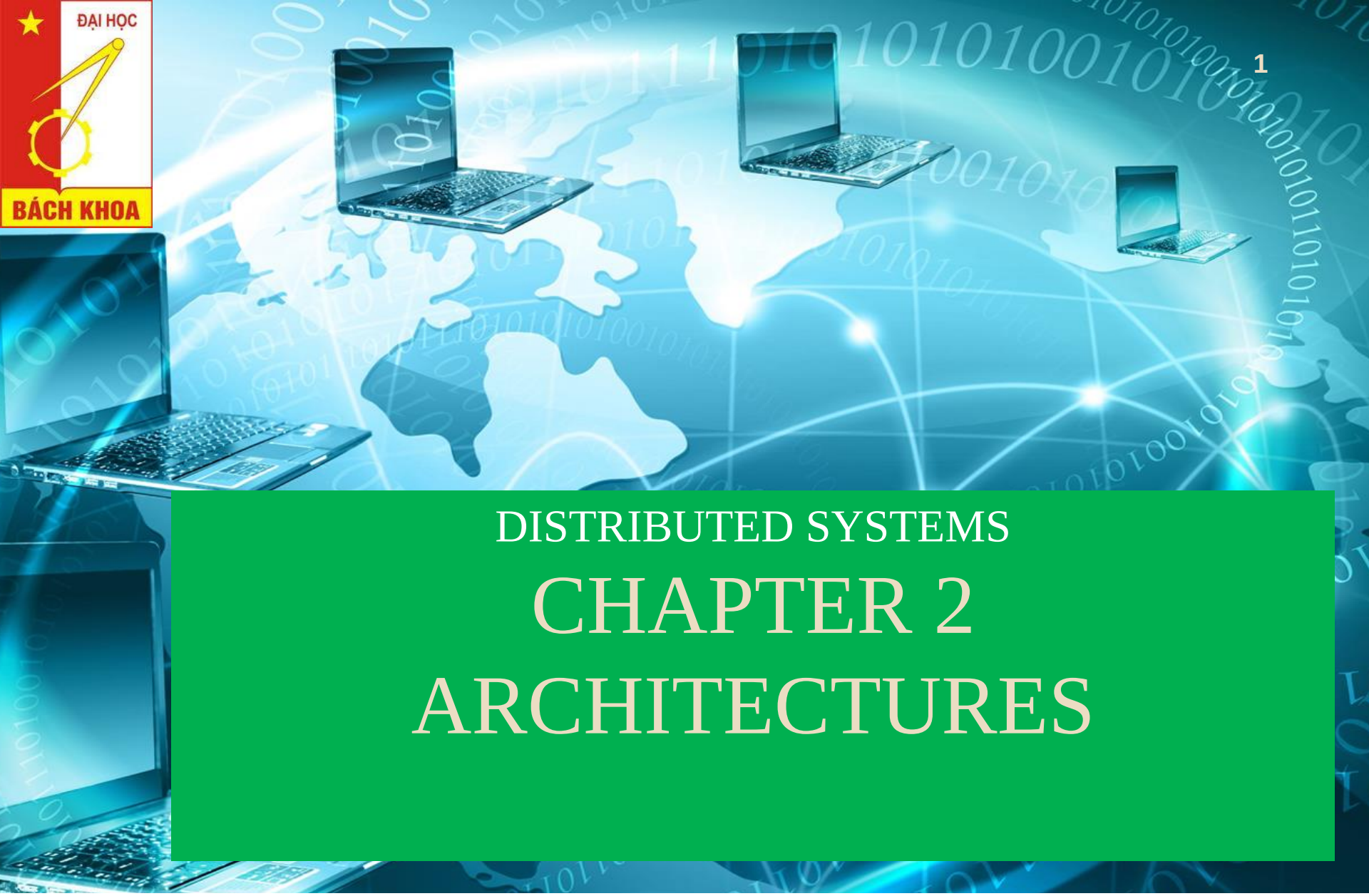


DISTRIBUTED SYSTEMS

CHAPTER 2
ARCHITECTURES

DR. TRẦN HẢI ANH

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

- Organization of a distributed system: → distinction between *the logical organization* and *the physical realization*
- The logical organization: the collection of software components that constitute the system → **software architecture**
- The physical realization: instantiate and place software components on real machines → **system architecture**

Outline

1. Architectural styles
2. System architectures
3. Architectures versus Middleware

1. Architectural styles

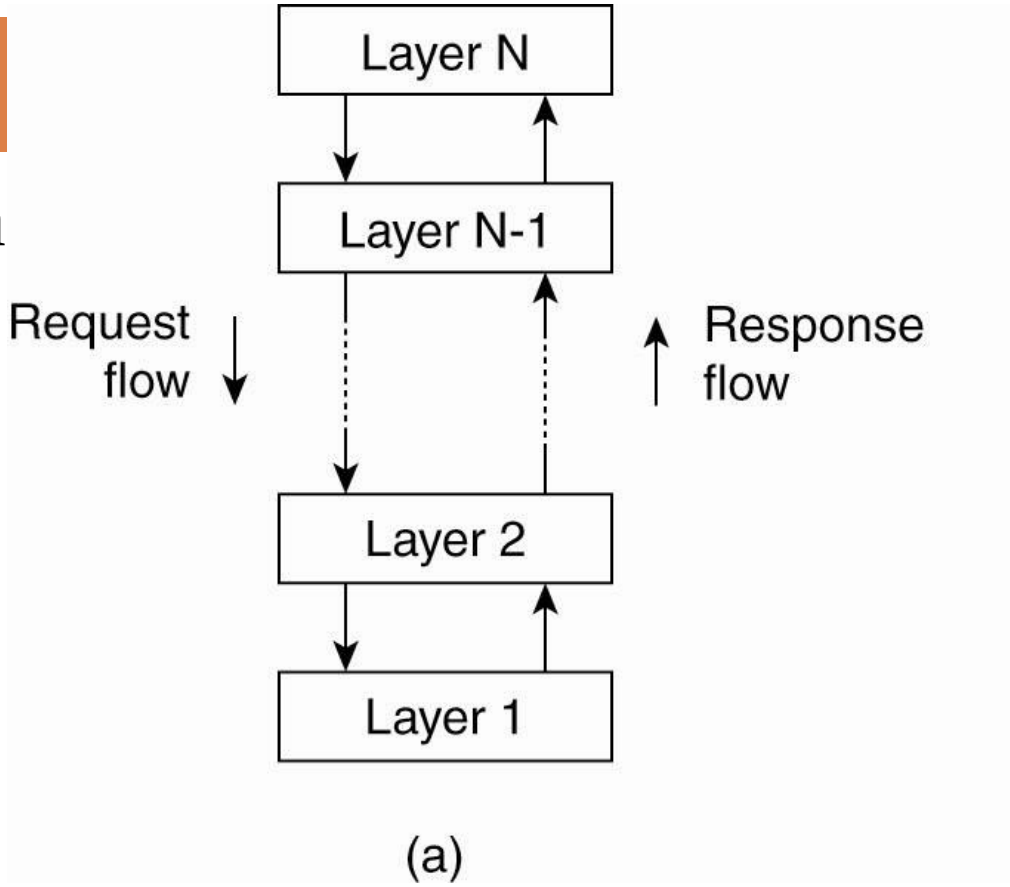
Architectural styles

- Layered architectures
- Object-based architectures
- Data-centered architectures
- Event-based architectures

1.1. Layered achitectures

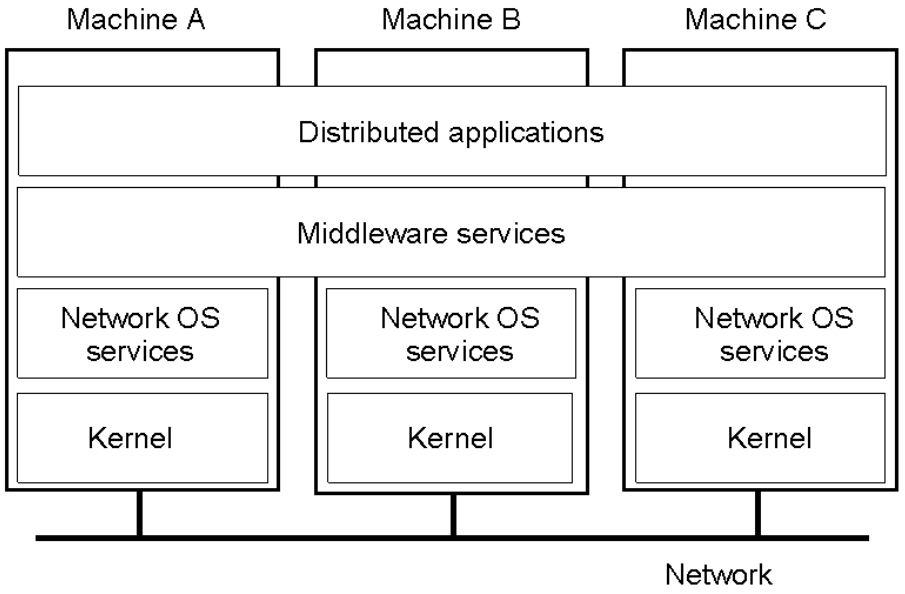
Layered fashion

- Each layer has its own task
- Transparency



Layered architectures (con't)

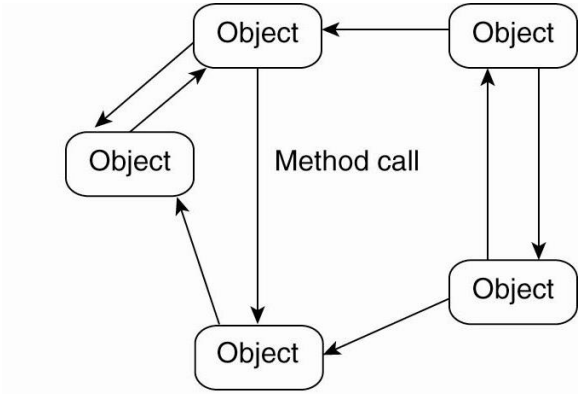
OSI model



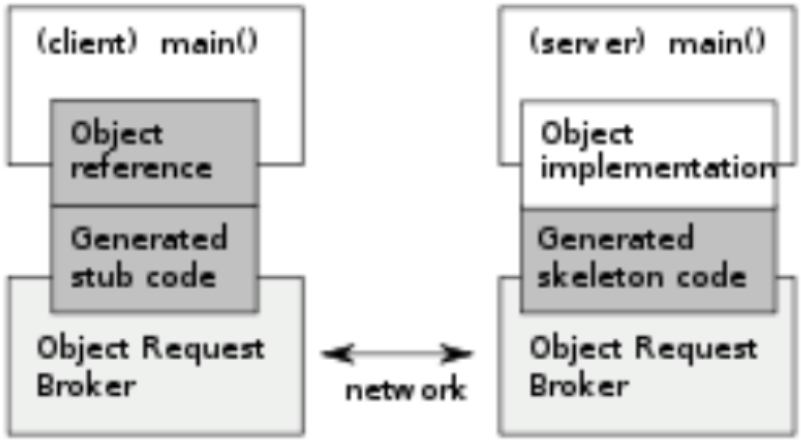
General DS architecture with Middleware

1.2. Object-based architectures

- Component: Object
- Connector: (Remote) Procedure call
- Object Client & Object server
- E.g. CORBA

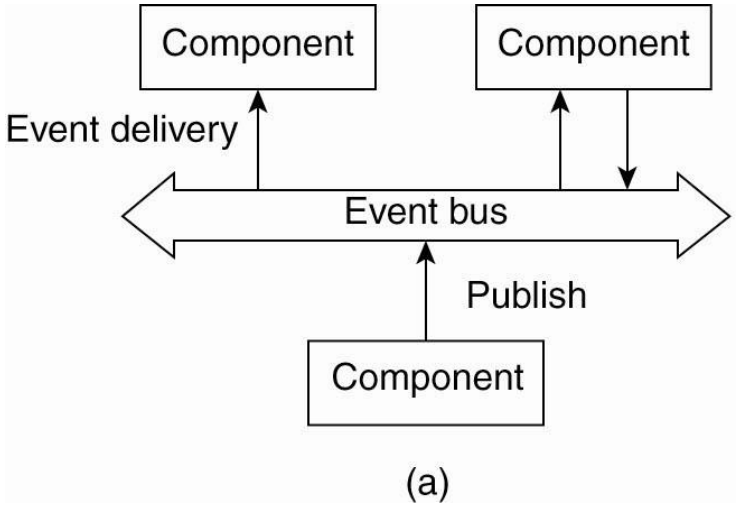


(b)



1.3. Event-based architectures

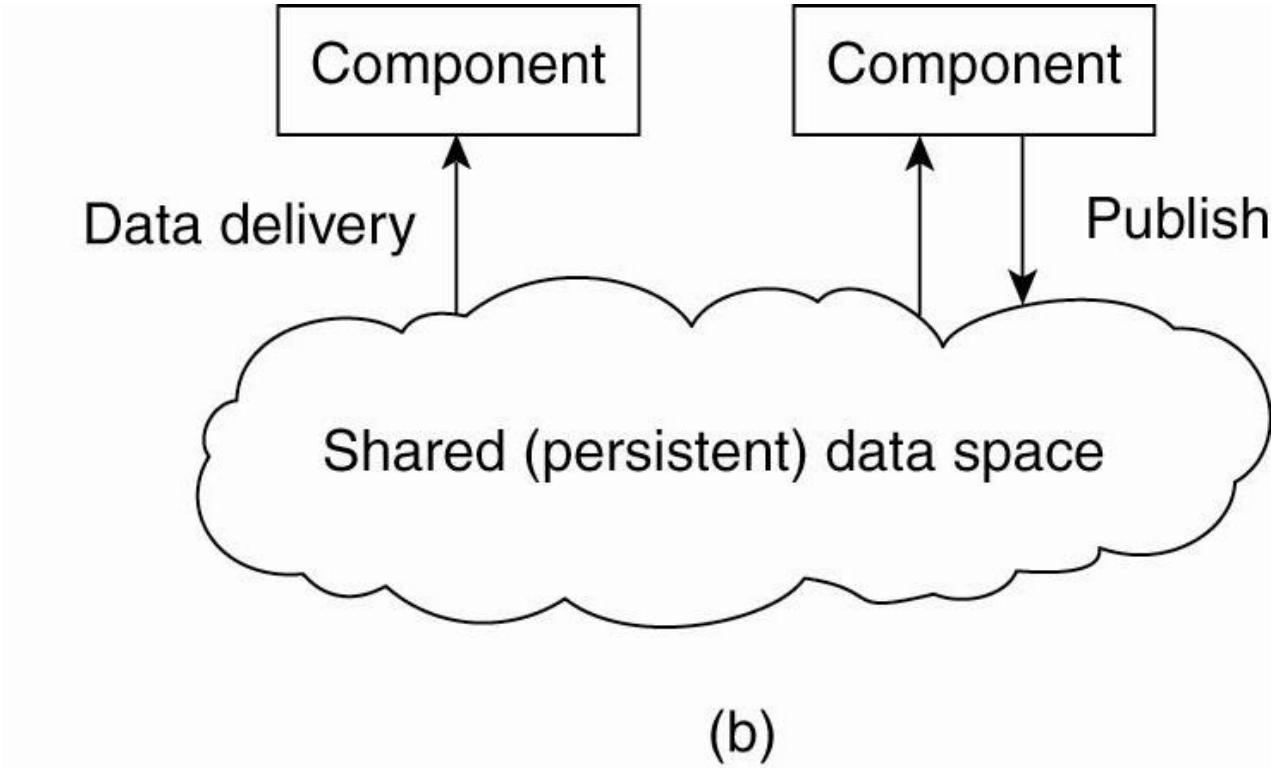
- Communicate through the propagation of events (optionally carry data)
- Publish/Subscribe systems
- Processes are loosely coupled



(a)

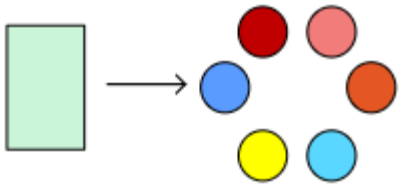
1.4. Data-centered architecture

- Communicate through a common repository

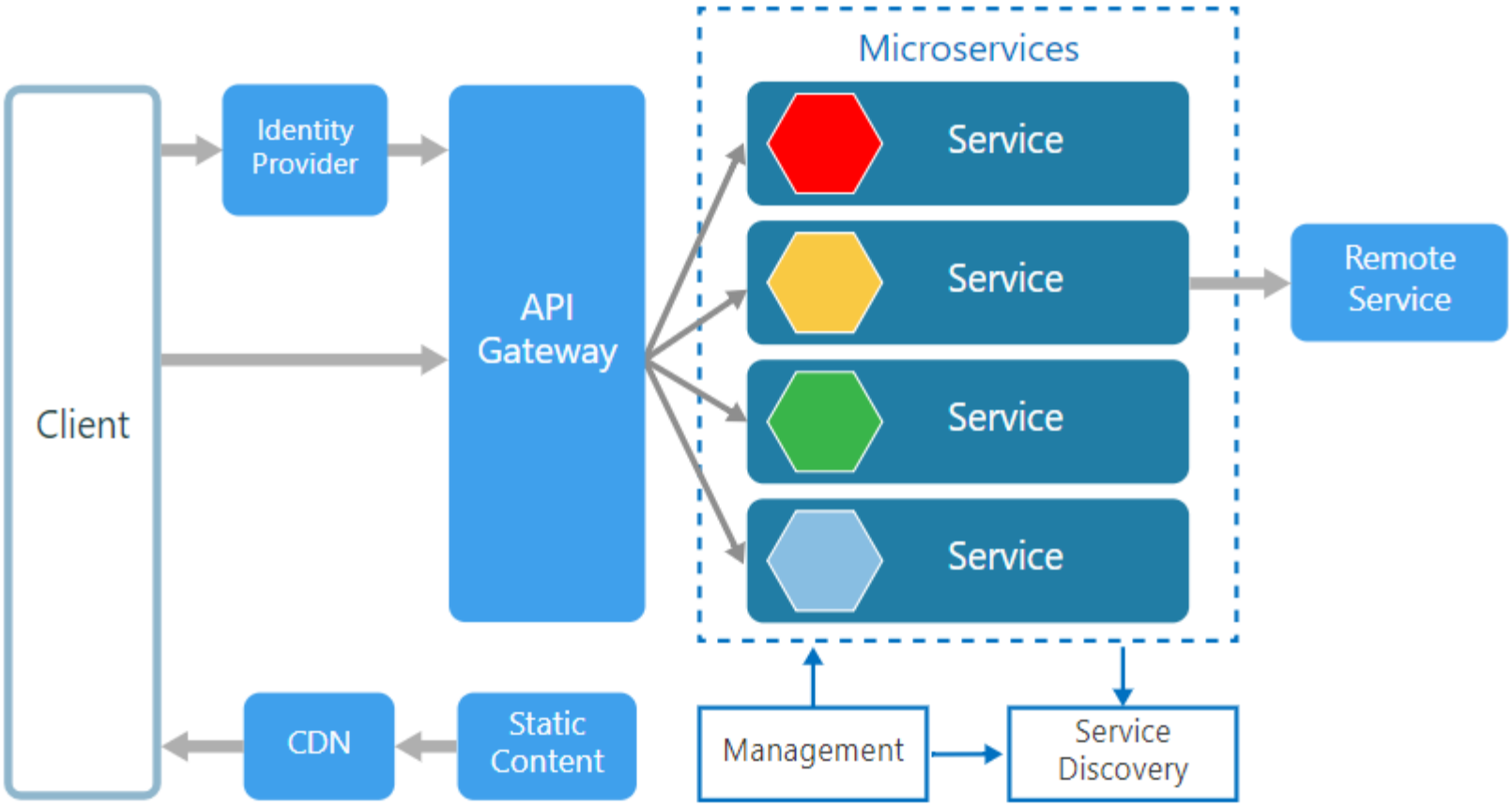


1.5. Microservices

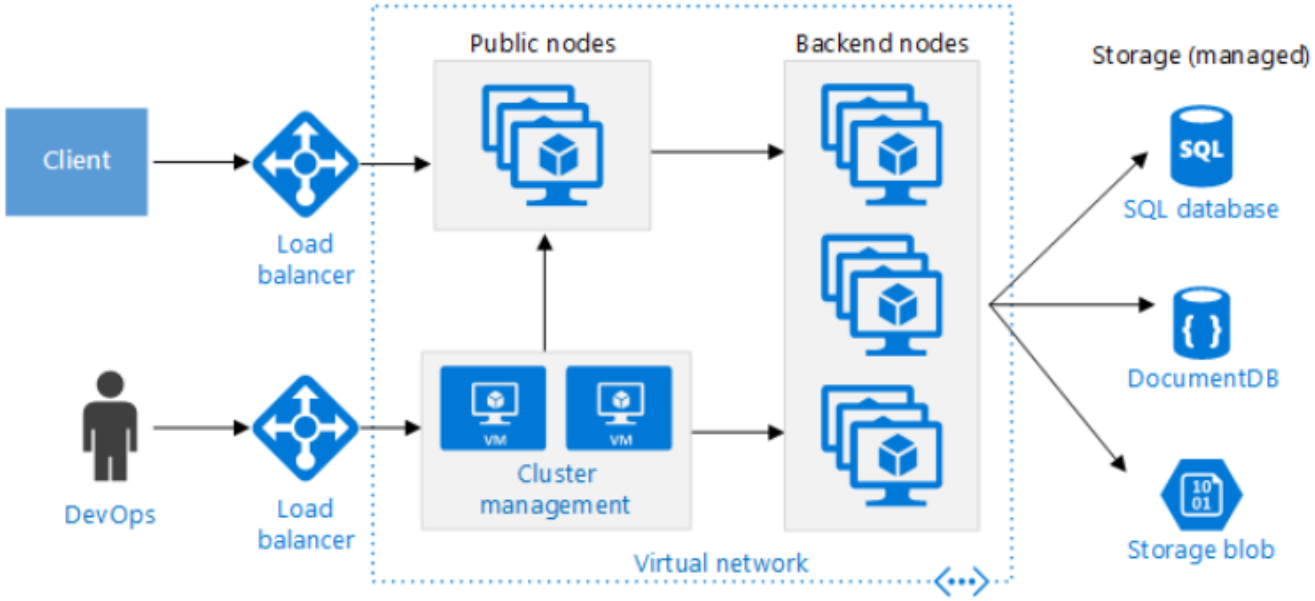
- Monolithic → microservices
- build an application as a suite of small services, each running in its own process and are independently deployable.
- Benefits:
  - Simpler To Deploy
  - Simpler To Understand
  - Reusability Across Business
  - Faster Defect Isolation
  - Minimized Risk Of Change



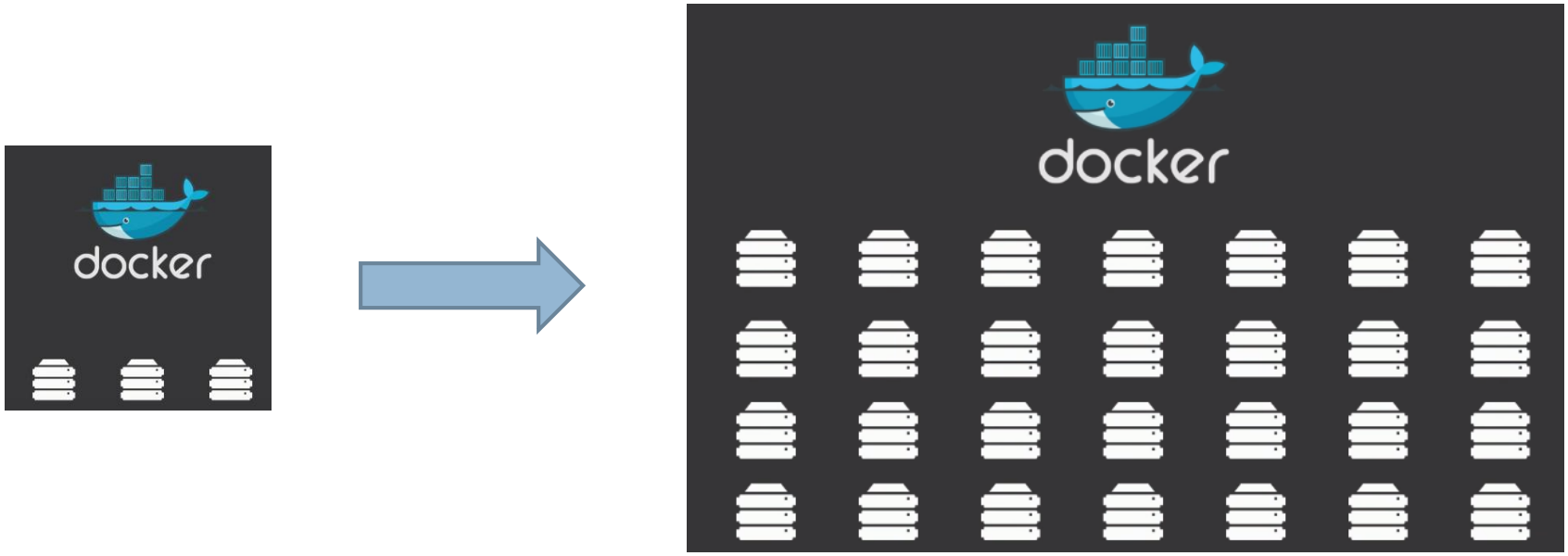
Microservices



Microservices



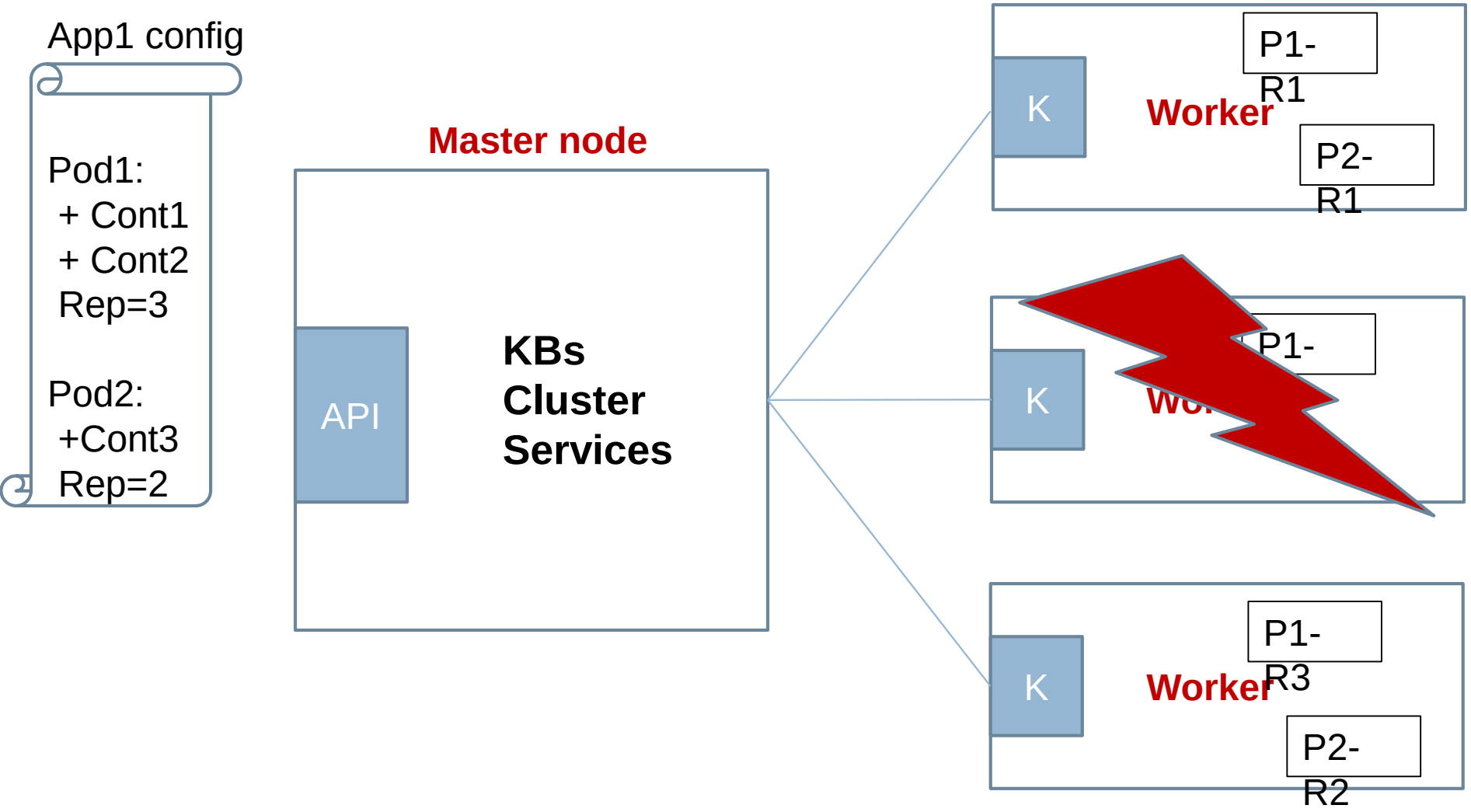
Problem



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. System architectures

I. Centralized architectures
II. Decentralized architectures
III. Hybrid architectures

2.1. Centralized architectures

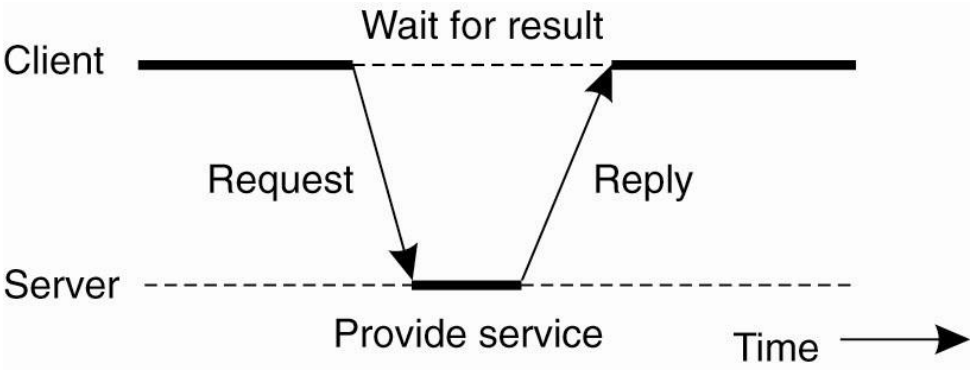
2.1.1. Client-server architectures

2.1.2. Application layering

2.1.3. Multitiered architectures

2.1.1. Client-server architecture

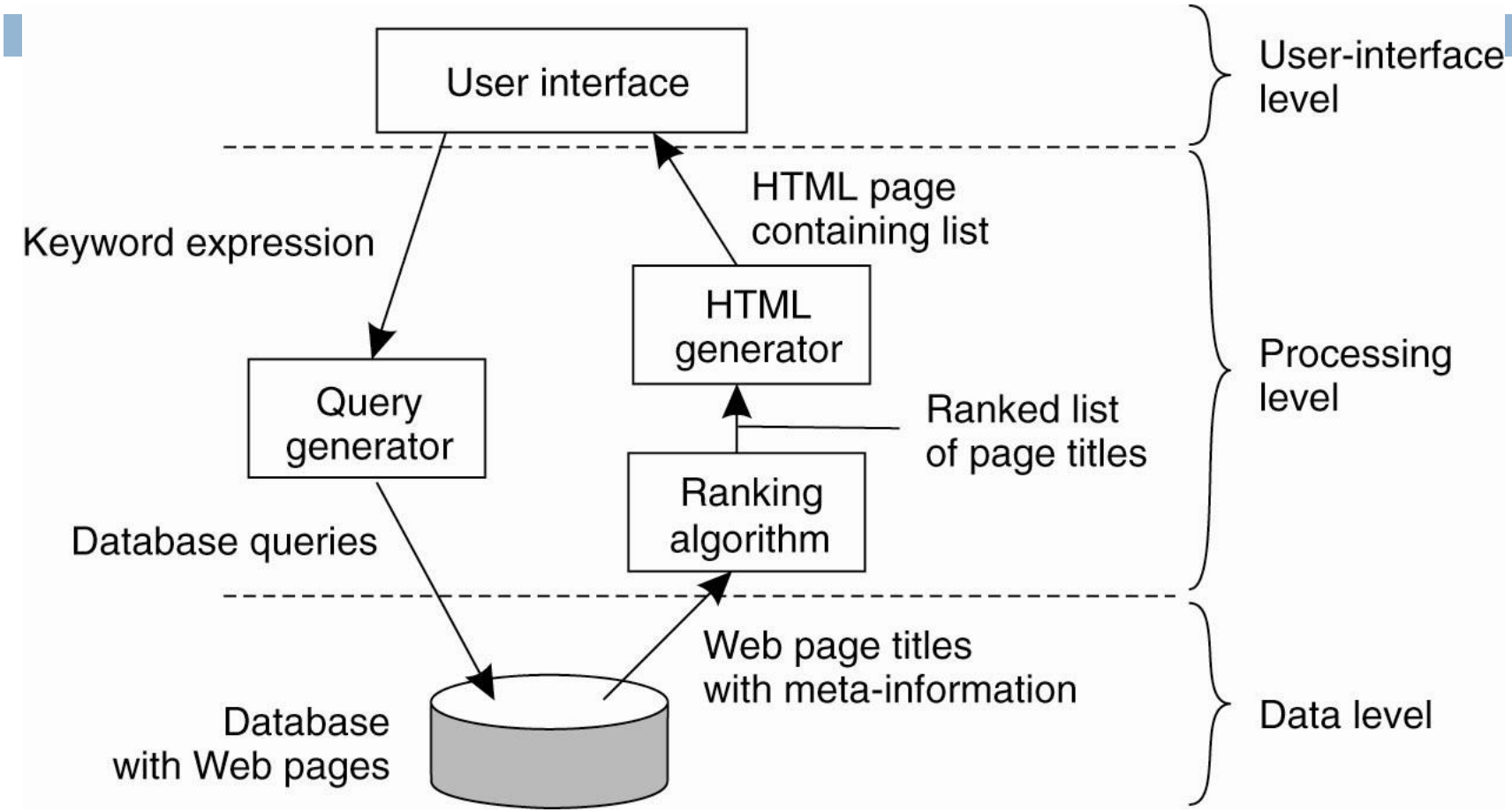
- Client:
  - Send the requests, receive the results, show to the users
- Server:
  - Listen; receive the request, processing, reply
- Connected or unconnected
- Issues:
  - Register the server
  - Idempotent
  - Stateful of Stateless server



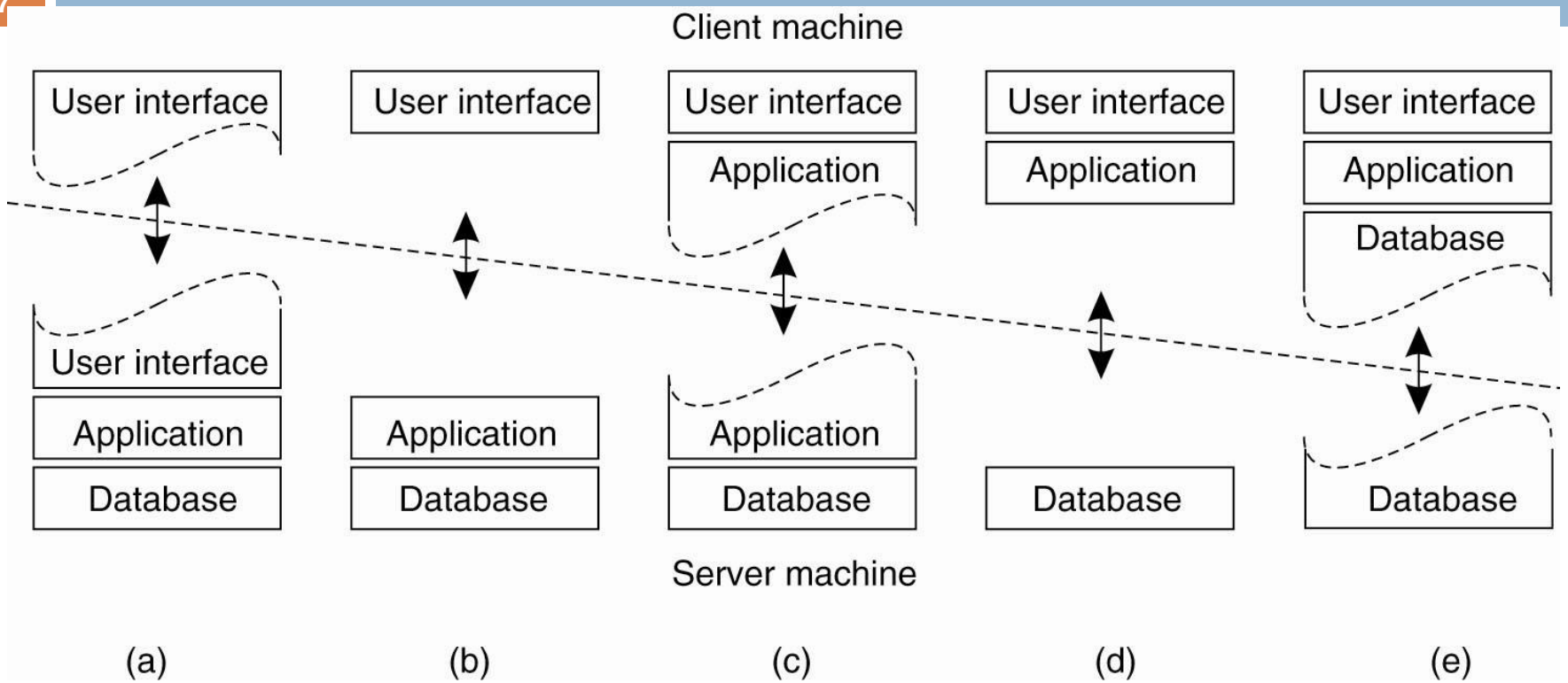
2.1.2. Application layering

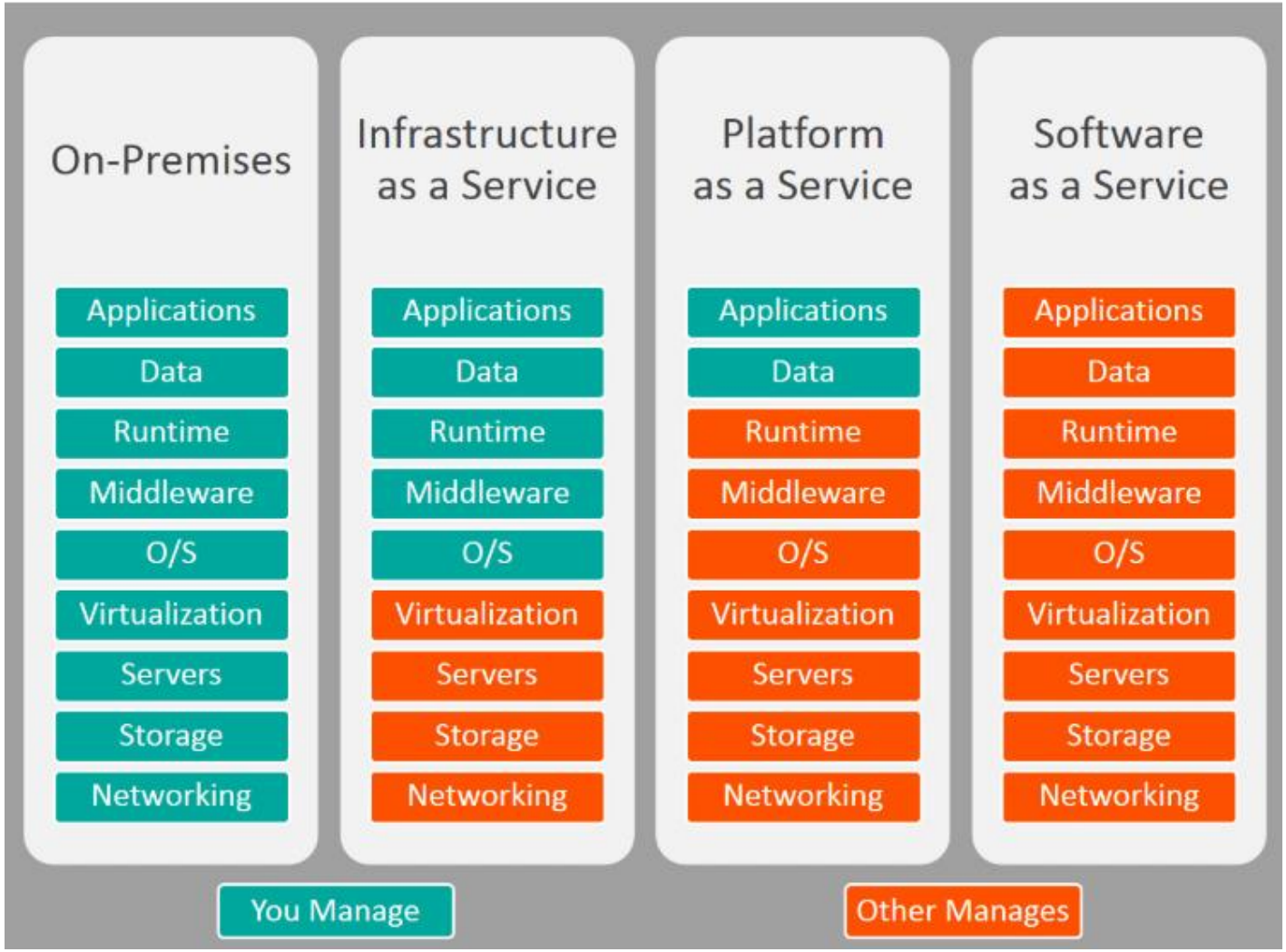
- The user-interface level
- The processing level
- The data level

Organization of a search engine

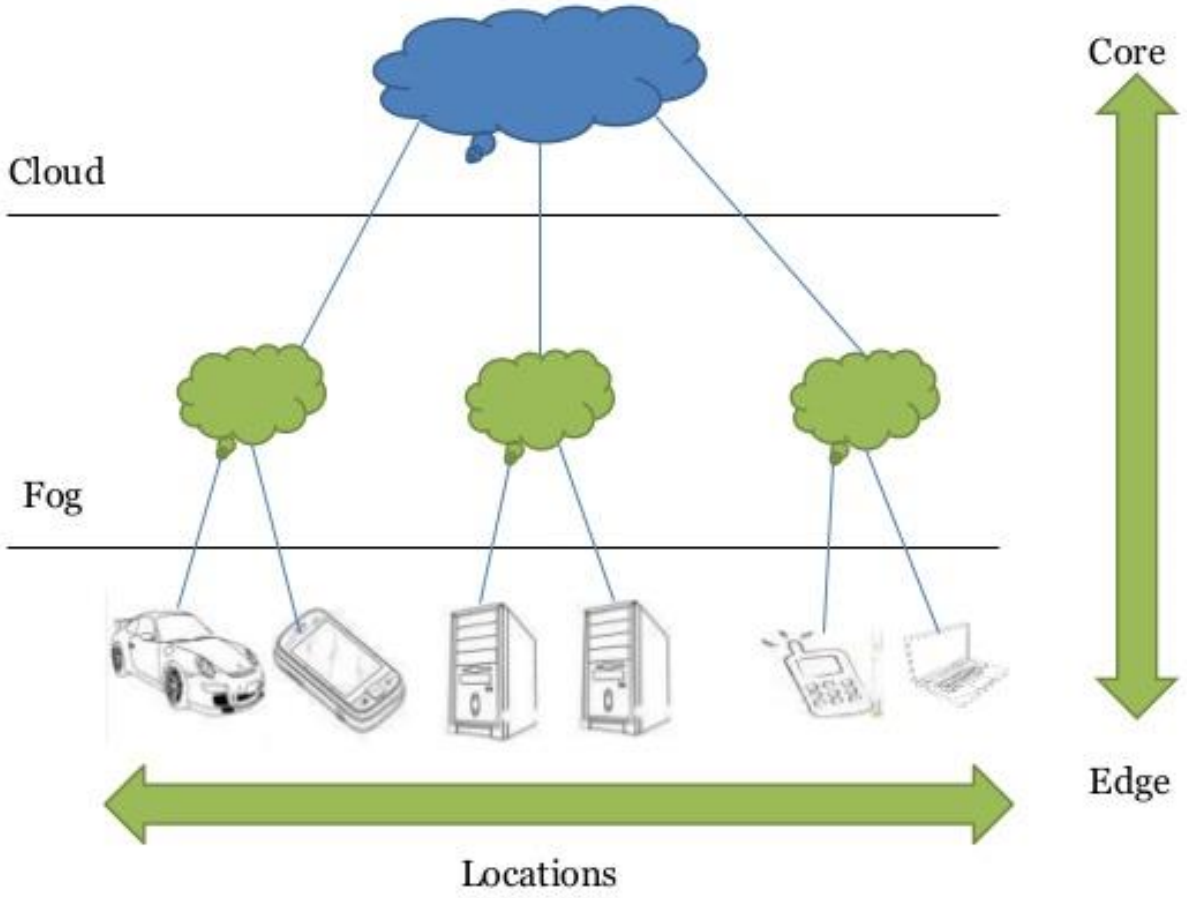


2.1.3. Multitiered architectures




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

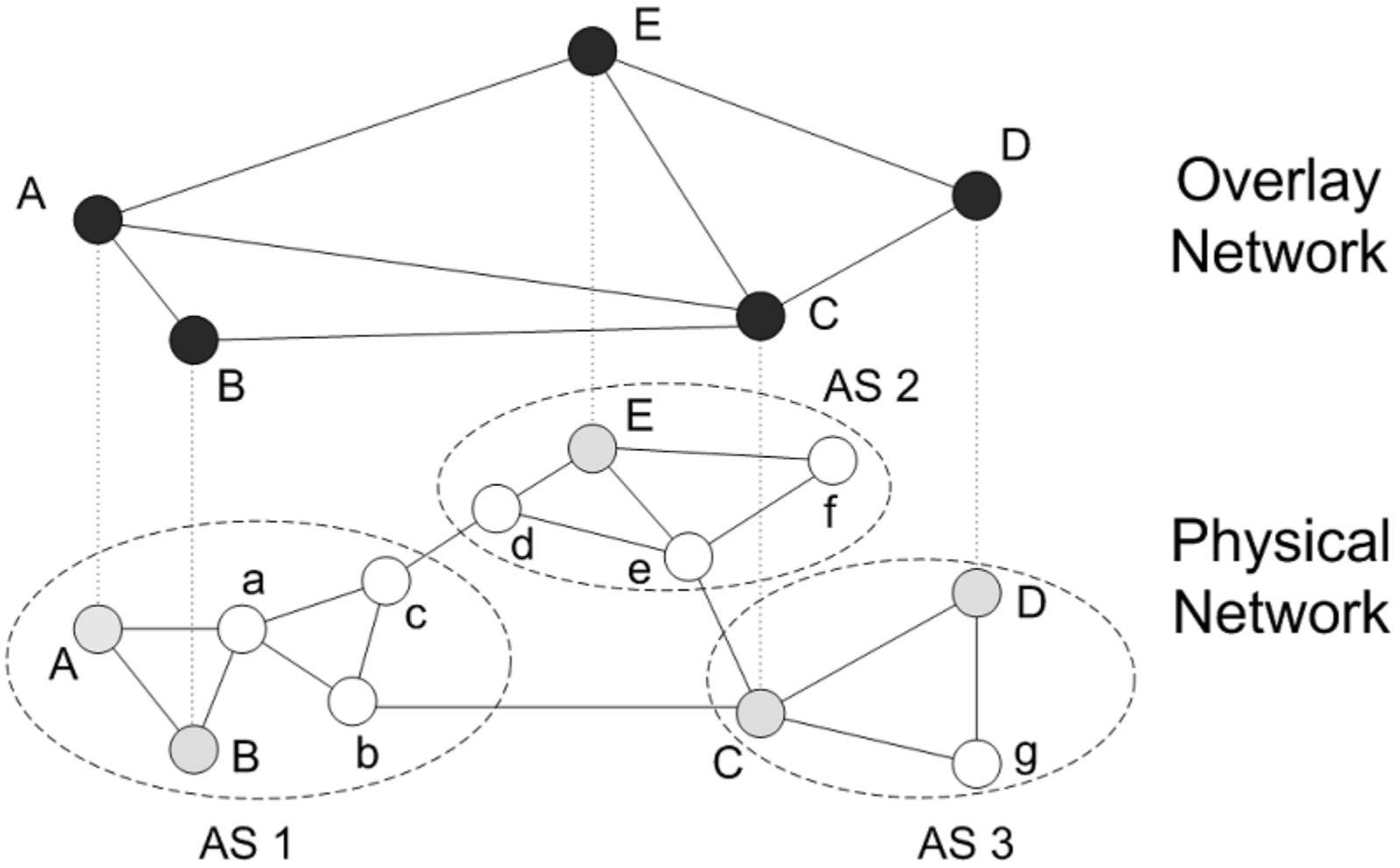
Cloud & Fog computing



2.2. Decentralized Architectures

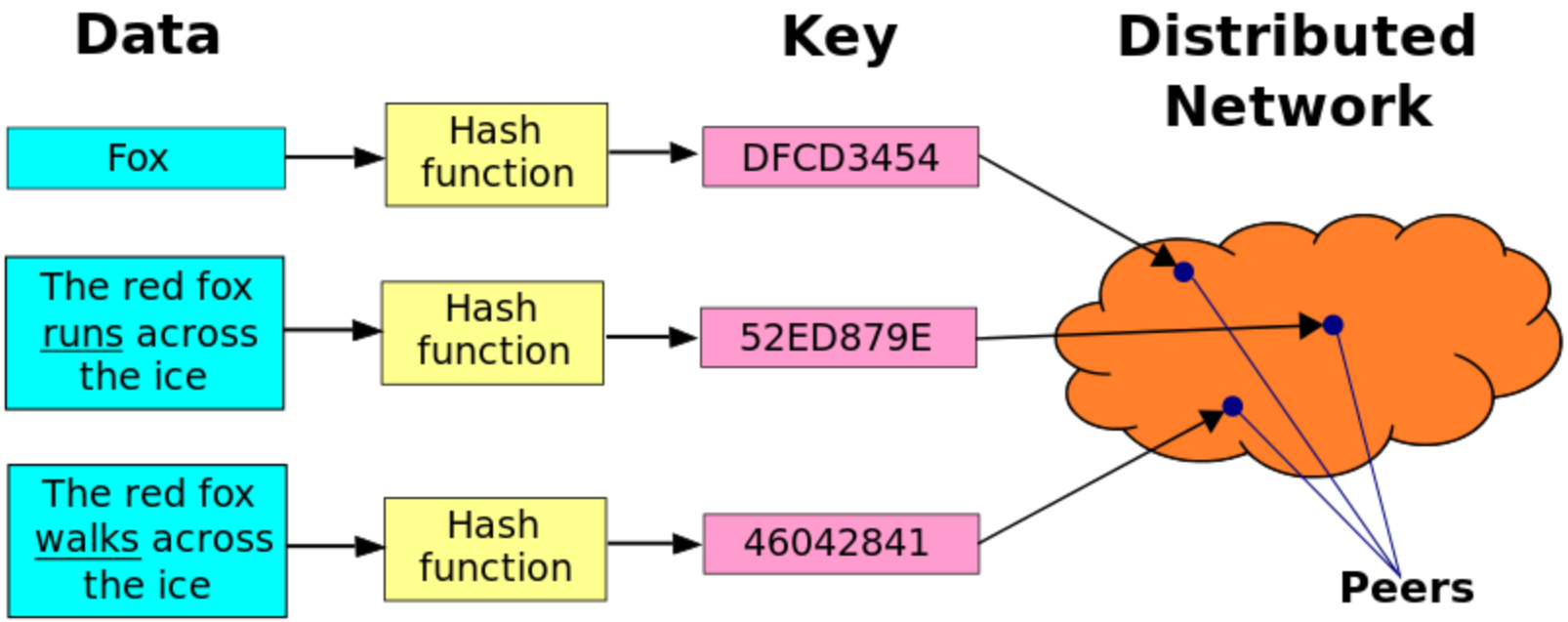
- No role of client and server
- Use Overlay network
- Structured/Unstructured P2P architectures

Overlay network

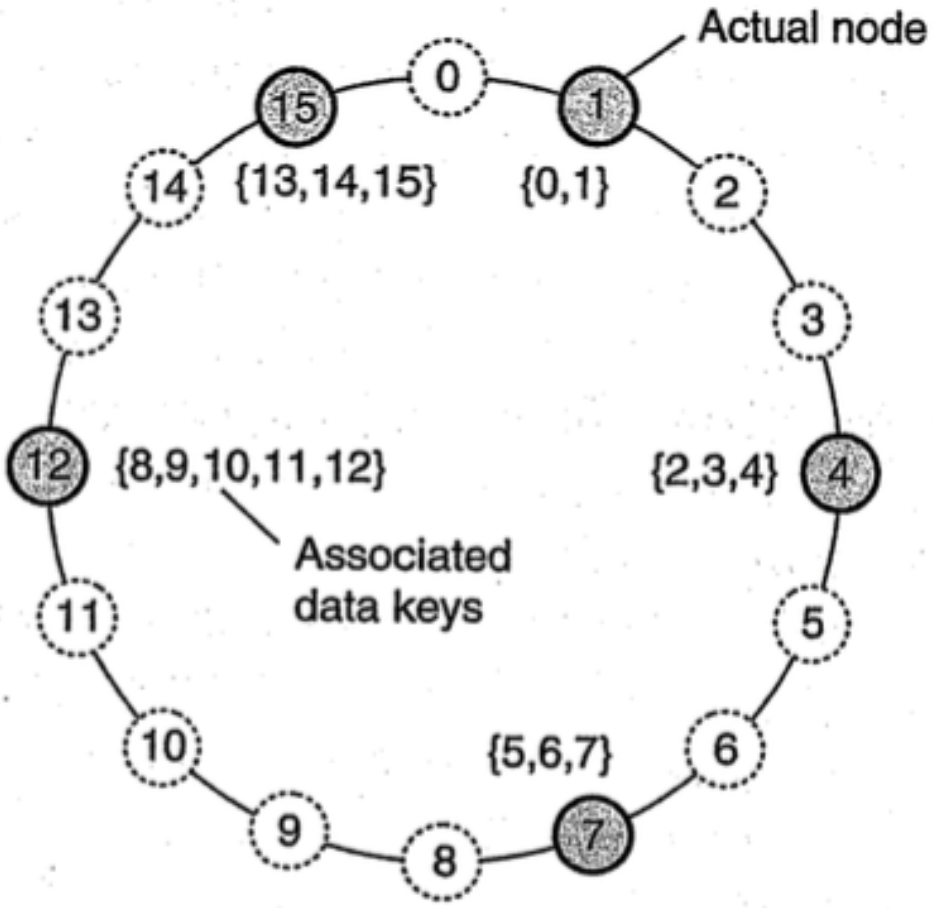


2.2.1. Structured P2P

- Overlay network is constructed using a deterministic procedure.
- DHT (Distributed Hash Table)

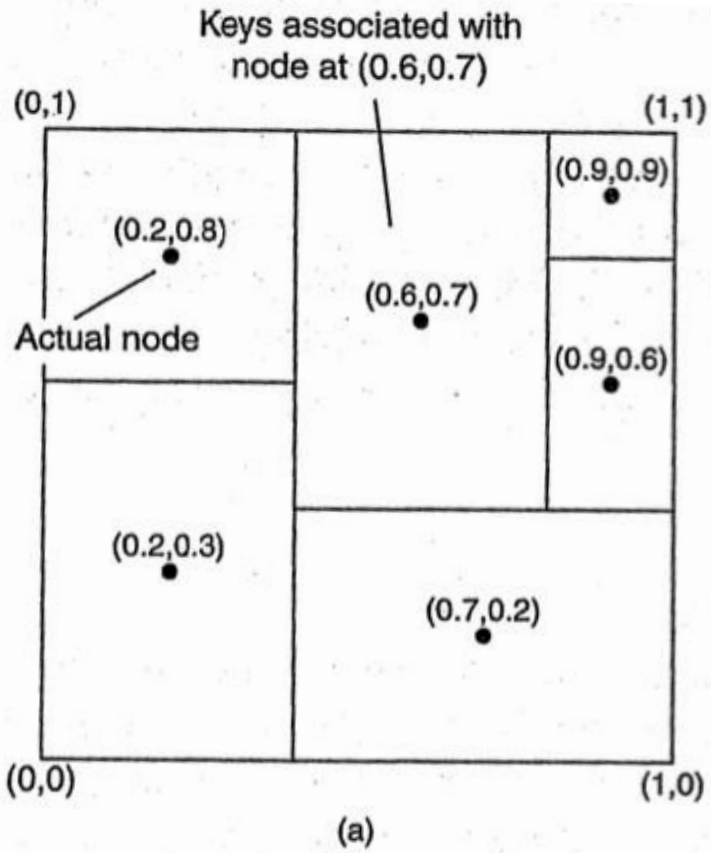


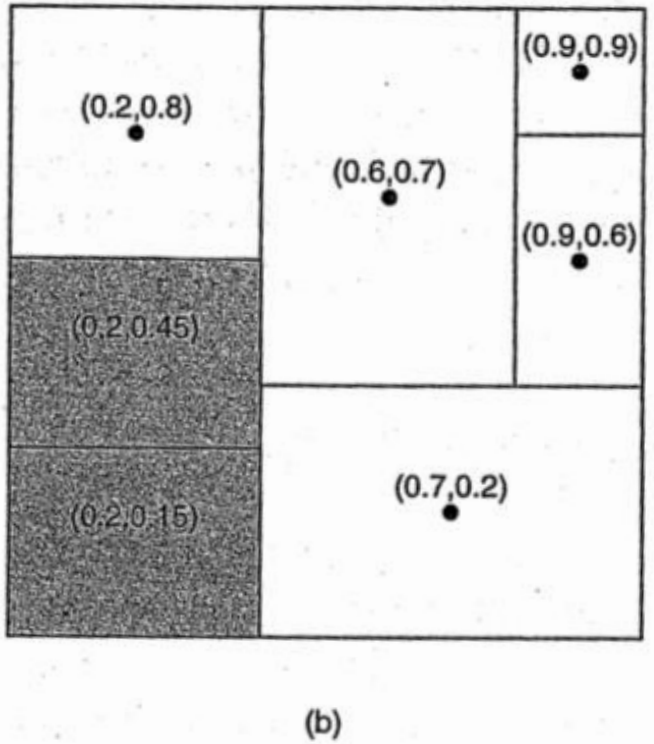
Chord system



- Logically organized in a ring
- Succ(k)
- Function LOOKUP(k)
- When a node wants to join the system
- When a node wants to leave the system

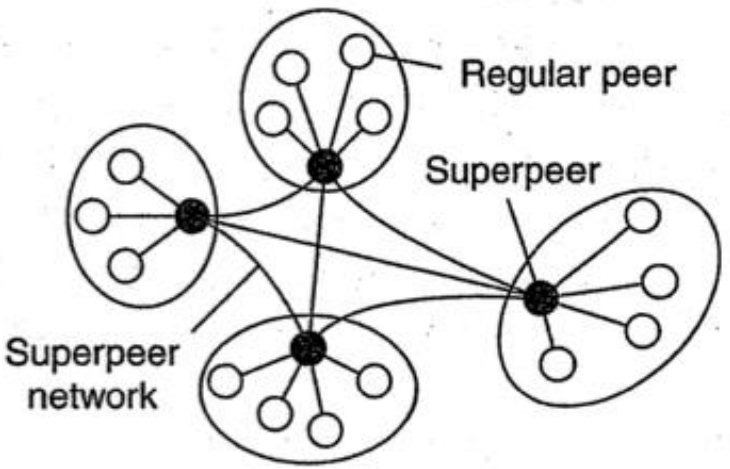
CAN system (Content Addressable Network)





2.2.2. Unstructured P2P architecture

- Randomized algorithms for constructing an overlay network.
- Each node maintains a list of neighbors
- Data items are assumed to be randomly placed on nodes → locating a specific data item needs flooding the network
- =>superpeers



2.3. Hybrid architectures

- Edge-Server Systems
- Collaborative Distributed Systems

Edge-Server Systems

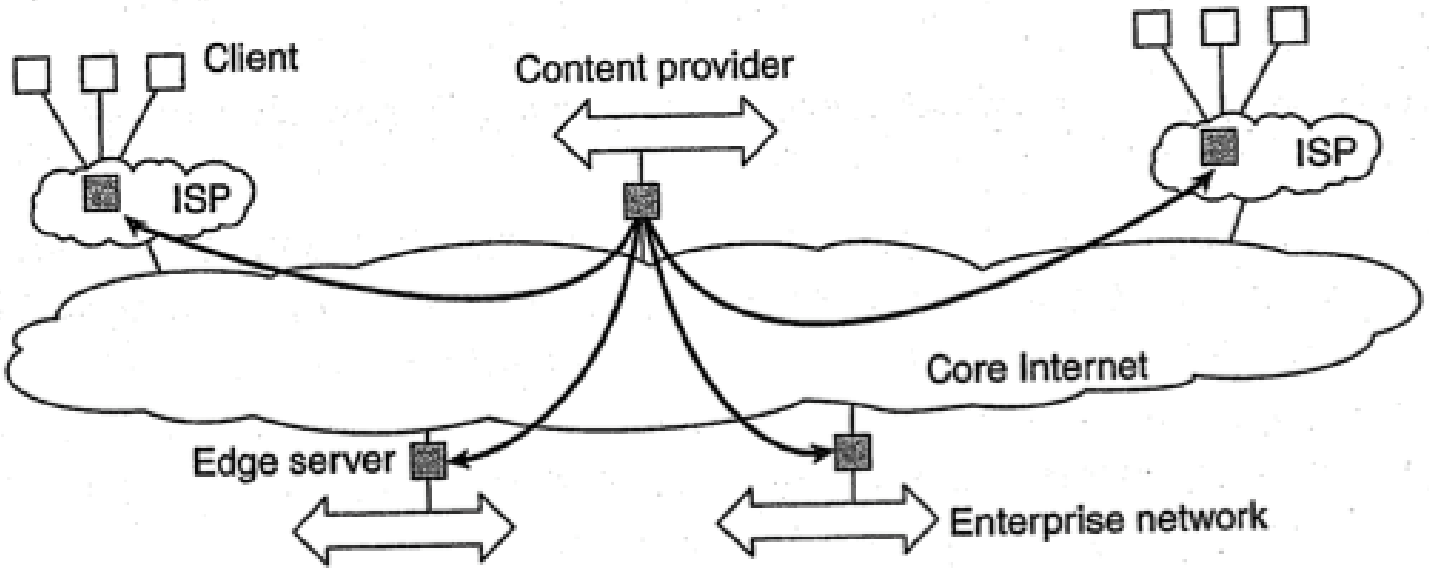
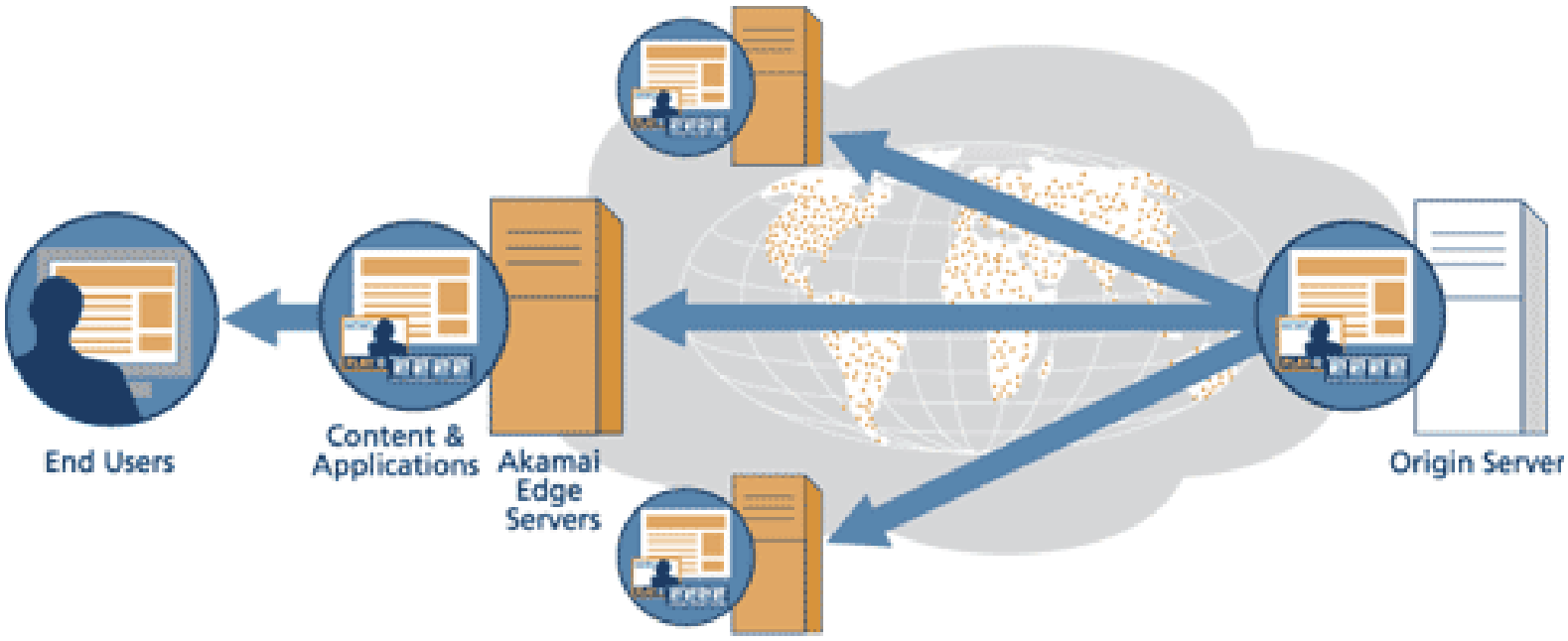
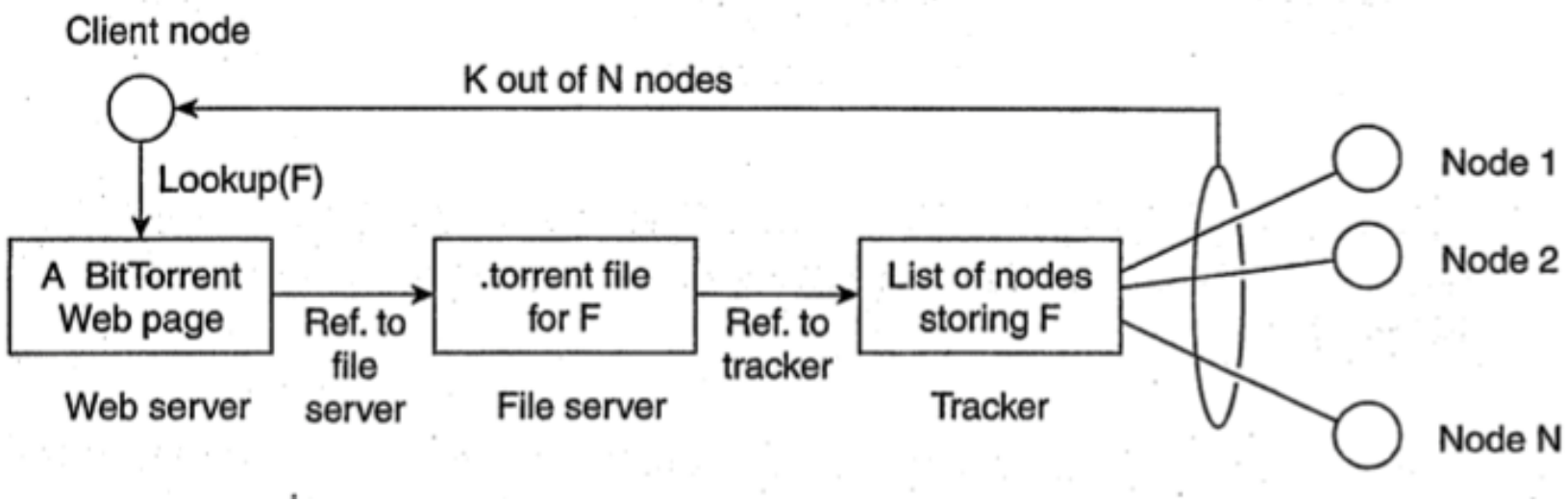


Figure 2-13. Viewing the Internet as consisting of a collection of edge servers.

Content Delivery Network



Collaborative Distributed Systems



BitTorrent system

3. Architectures versus Middleware

Where does Middleware fit in?

- Position of middleware
- E.g. CORBA (object-based architecture), TIB/Rendezvous (event-based architecture)
- Benefit: designing applications may become simpler
- Drawback: no longer be optimal for application developers
- Solutions:
  - make several versions of a middleware system
  - separate between policies and mechanisms → easy to configure, adapt and customize middleware

Interceptors

- software construct → break the usual flow of control and allow other code to be executed