Posts and Telecommunications Institute of Technology
U
C
L
Ubiquitous Computing Laboratory

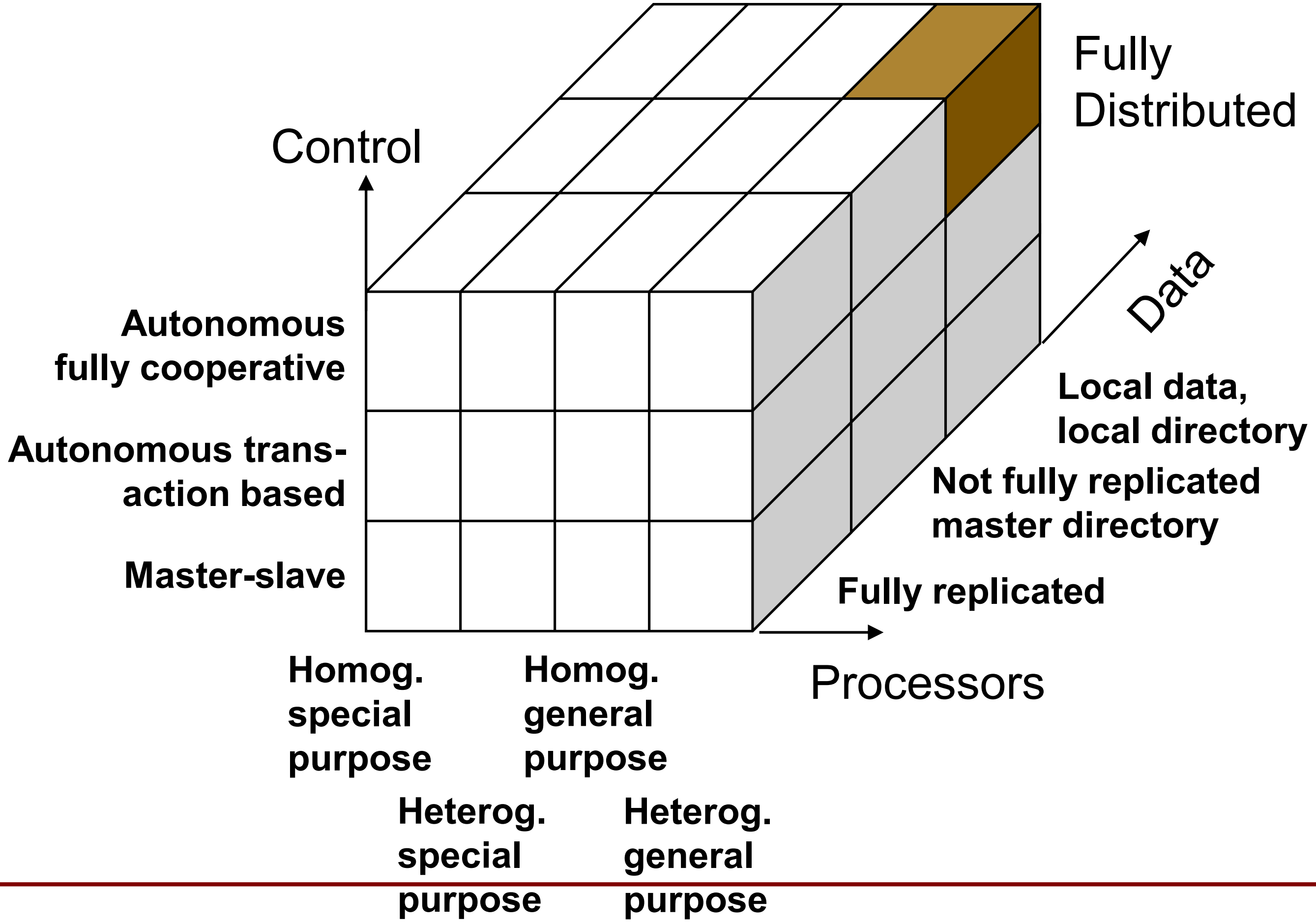
Chapter 1: Characterization of distributed systems

Sam Nguyen-Xuan, Ph.D.
Faculty of Information Technology No. 2
Posts and Telecoms of Institute and Technology
Ho Chi Minh City Campus, Vietnam

Outline

1. What is a Distributed System
2. Examples of Distributed Systems
3. Common Characteristics
4. Basic Design Issues
5. Summary

1. Distributed System Types



1. What is a Distributed System?

Definition: A *distributed system* is one in which **components** located at **networked computers** communicate and coordinate their actions only by passing messages. This definition leads to the following characteristics of distributed systems:

☞Concurrency of components

☞Lack of a global clock

☞Independent failures of components

1.1 Centralized System Characteristics

- One component with non-autonomous parts
- Component shared by users all the time
- All resources accessible
- Software runs in a single process
- Single point of control
- Single point of failure

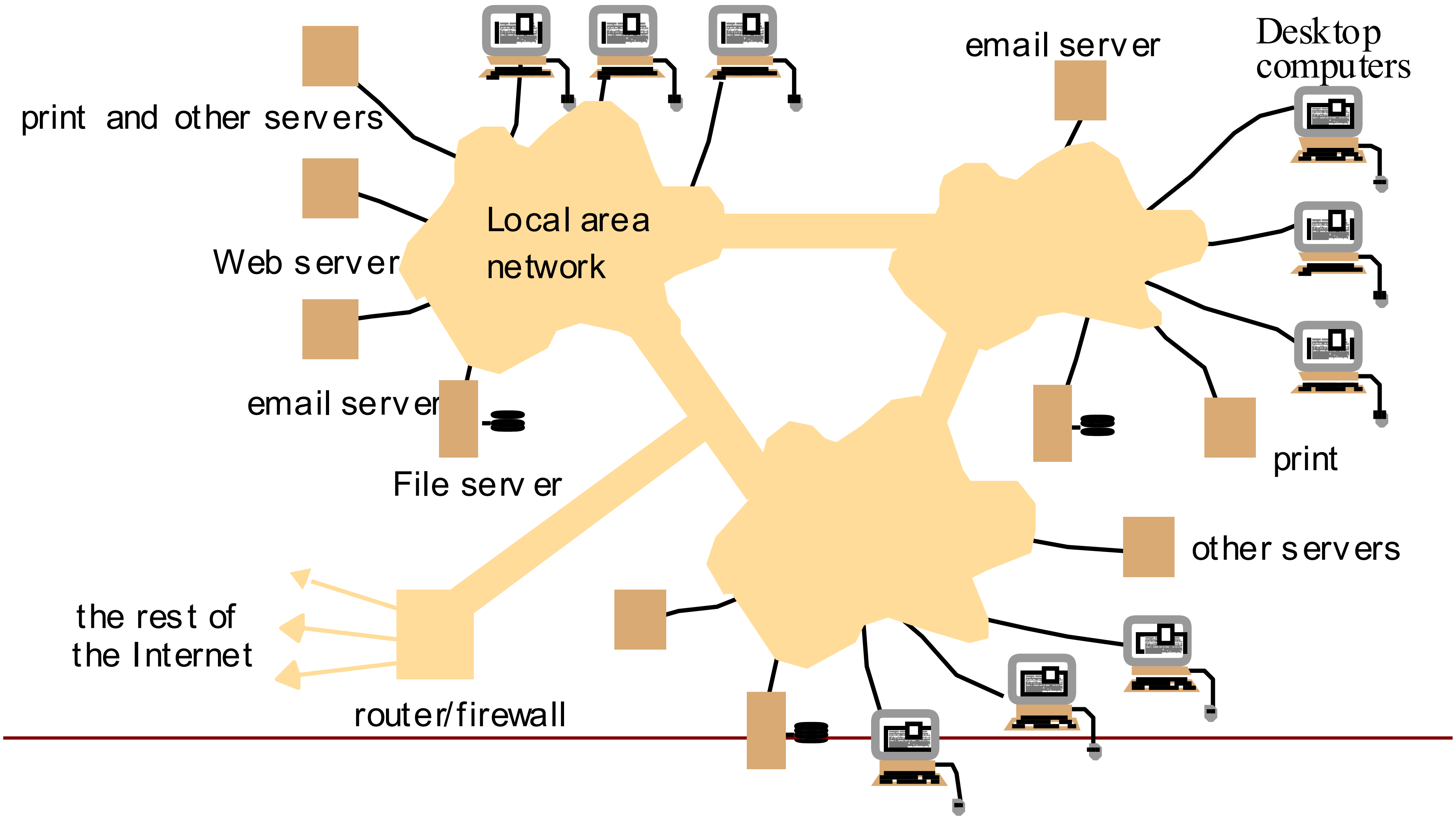
1.2 Distributed System Characteristics

- Multiple autonomous components
- Components are not shared by all users
- Resources may not be accessible
- Software runs in concurrent processes on different processors
- Multiple points of control
- Multiple points of failure

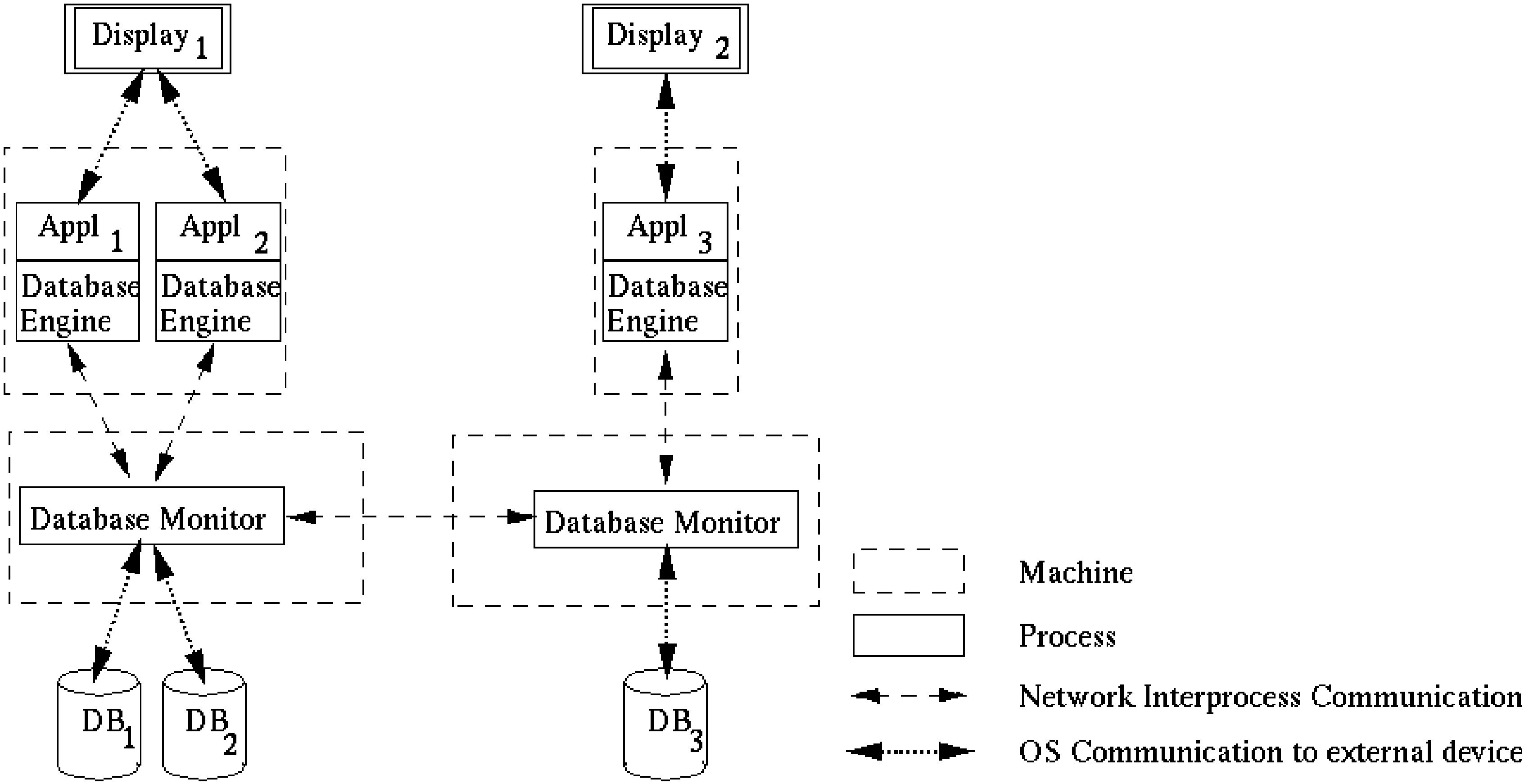
2. Examples of Distributed Systems

- Local Area Network and Intranet
- Database Management System
- Automatic Teller Machine Network
- Internet/World-Wide Web
- Mobile and Ubiquitous Computing

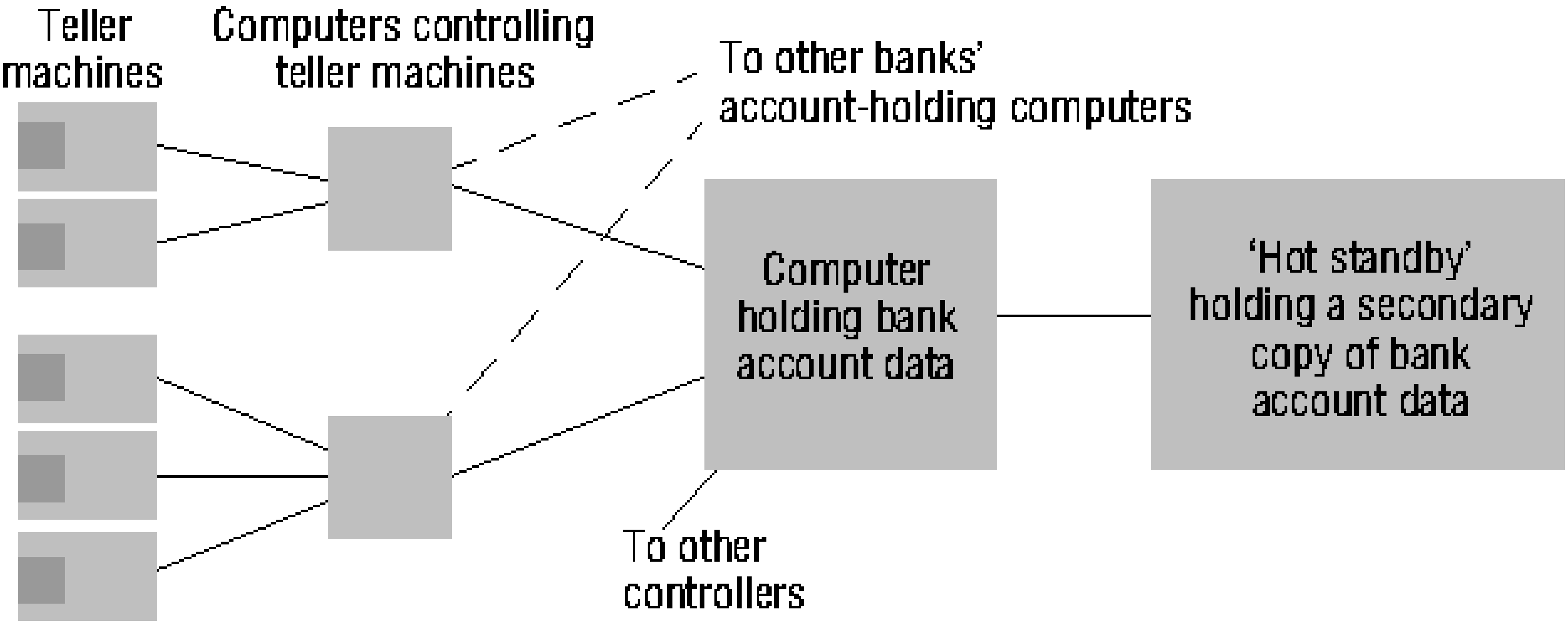
2.1 Local Area Network



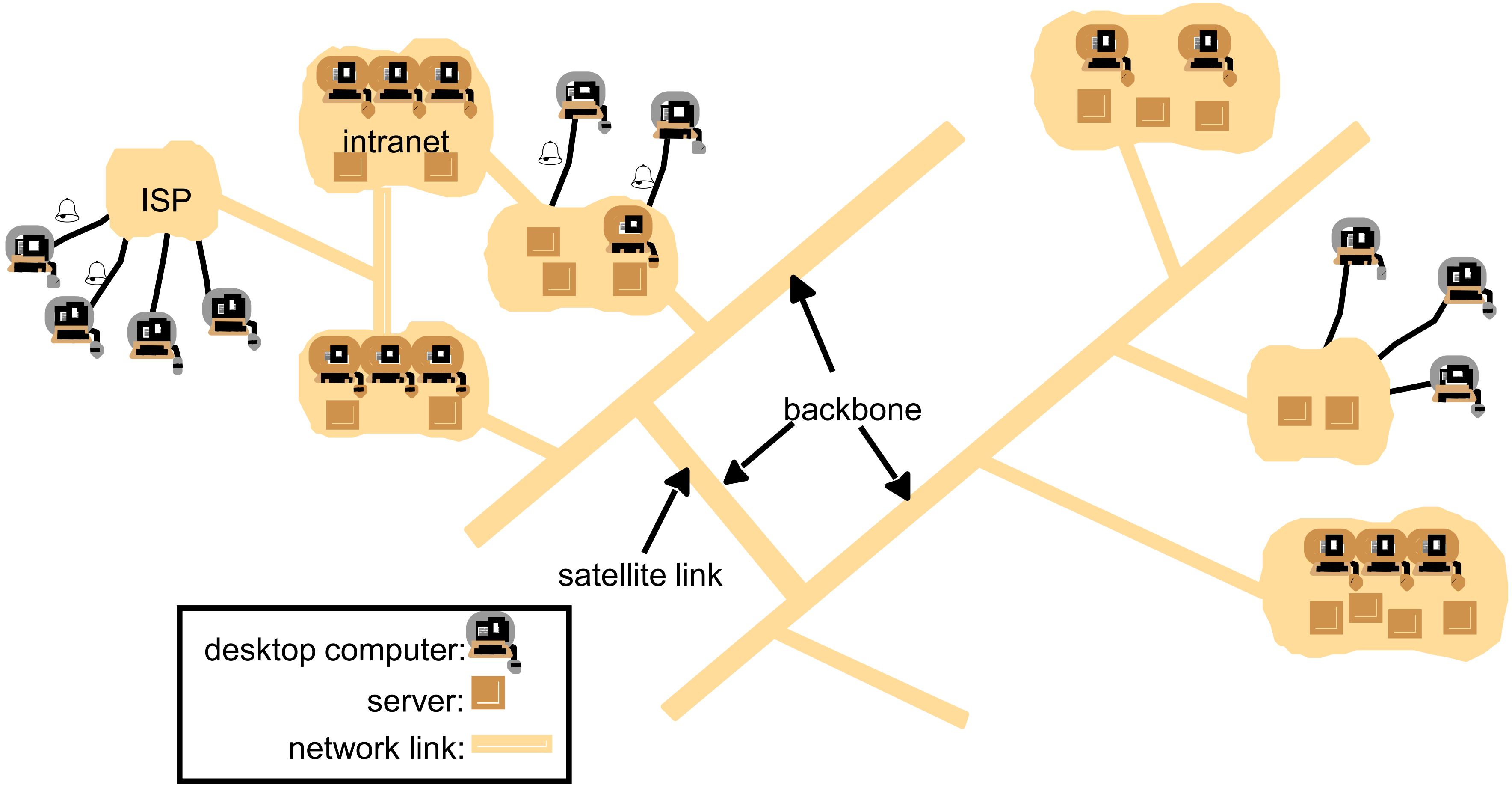
2.2 Database Management System (DBMS)



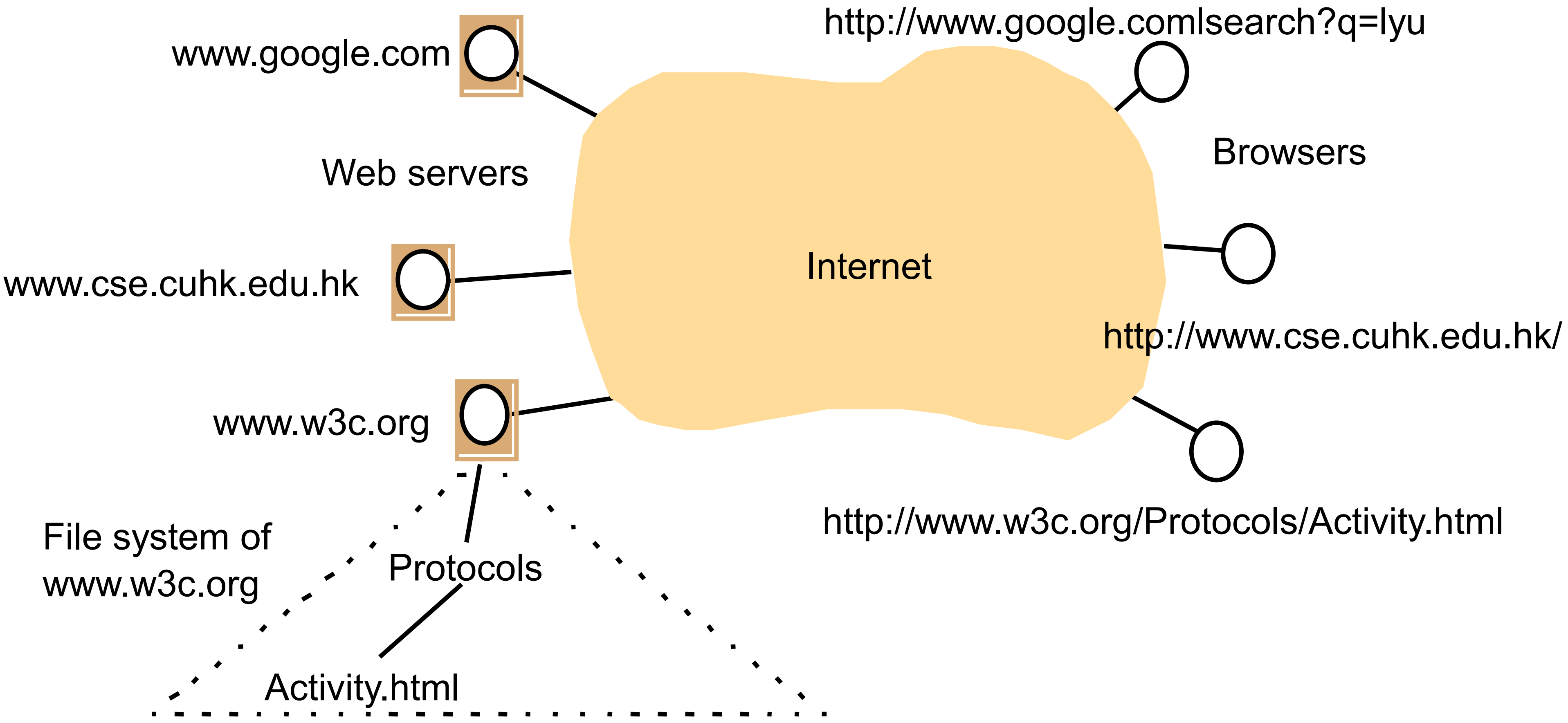
2.3 Automatic Teller Machine Network


Teller machines
Computers controlling teller machines
To other banks' account-holding computers
Computer holding bank account data
'Hot standby' holding a secondary copy of bank account data
To other controllers

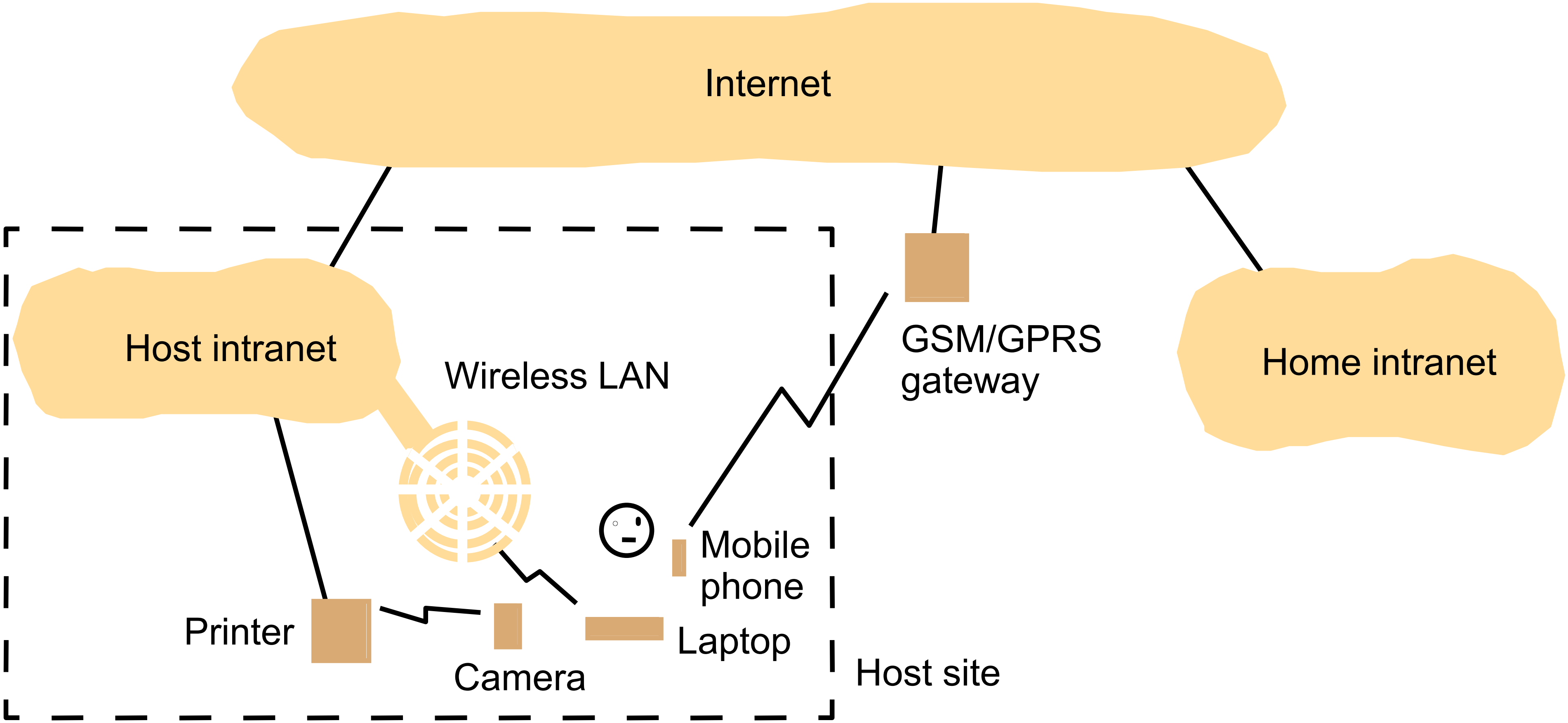
2.4 Internet


ISP
intranet
backbone
satellite link
desktop computer:
server:
network link:

2.4.2 Web Servers and Web Browsers



2.5 Mobile and Ubiquitous Computing



3. Common Characteristics

- What are we trying to achieve when we construct a distributed system?
- Certain common characteristics can be used to assess distributed systems
  - Heterogeneity
  - Openness
  - Security
  - Scalability
  - Failure Handling
  - Concurrency
  - Transparency

3.1 Heterogeneity

- Variety and differences in
  - Networks
  - Computer hardware
  - Operating systems
  - Programming languages
  - Implementations by different developers
- *Middleware* as software layers to provide a programming abstraction as well as masking the heterogeneity of the underlying networks, hardware, OS, and programming languages (e.g., Web service).
- *Mobile Code* to refer to code that can be sent from one computer to another and run at the destination (e.g., Java applets, Java *virtual machine, Apps*).

3.2 Openness

- Openness is concerned with extensions and improvements of distributed systems.
- Detailed interfaces of components need to be published.
- New components have to be integrated with existing components.
- Differences in data representation of interface types on different processors (of different vendors) have to be resolved.

3.3 Security

- In a distributed system, clients send requests to access data managed by servers, resources in the networks:
  - Doctors requesting records from hospitals
  - Users purchase products through electronic commerce
- Security is required for
  - Concealing the contents of messages: security and privacy
  - Identifying a remote user or other agent correctly: authentication
- New challenges:
  - Denial of service attack
  - Security of mobile code or apps

3.4 Scalability

- Adaptation of distributed systems to
  - accommodate more users
  - respond faster (this is the hard one)
- Usually done by adding more and/or faster processors.
- Components should not need to be changed when scale of a system increases.
- Design components to be scalable!

3.5 Failure Handling (Fault Tolerance)

- Hardware, software and networks fail!
- Distributed systems must maintain *availability* even at low levels of hardware/software/network *reliability*.
- Fault tolerance is achieved by
  - recovery
  - redundancy

3.6 Concurrency

- Components in distributed systems are executed in concurrent processes.
- Components access and update shared resources (e.g. variables, databases, device drivers).
- Integrity of the system may be violated if concurrent updates are not coordinated.
  - Lost updates
  - Inconsistent analysis

3.7 Transparency

- Distributed systems should be perceived by users and application programmers as a whole rather than as a collection of cooperating components.

- Transparency has different aspects.

- These represent various properties that distributed systems should have.

3.7.1 Access Transparency

- Enables local and remote information objects to be accessed using identical operations.
- Example: File system operations
- Example: Navigation in the Web
- Example: Database queries.

3.7.2 Location Transparency

- Enables information objects to be accessed without knowledge of their location.

- Example: File system operations

- Example: Pages in the Web

- Example: Tables in distributed databases

3.7.3 Concurrency Transparency

- Enables several processes to operate concurrently using shared information objects without interference between them.
- Example: File system operations
- Example: Automatic teller machine network
- Example: Database Management System (DBMS)

3.7.4 Replication Transparency

- Enables multiple instances of information objects to be used to increase reliability and performance without knowledge of the replicas by users or application programs

- Example: Distributed DBMS

- Example: Mirroring Web Pages

3.7.5 Failure Transparency

- Enables the concealment of faults

- Allows users and applications to complete their tasks despite the failure of other components.

- Example: Database Management System (DBMS)

3.7.6 Mobility Transparency

- Allows the movement of information objects within a system without affecting the operations of users or application programs

- Example: NFS

- Example: Web Pages

3.7.7 Performance Transparency

- Allows the system to be reconfigured to improve performance as loads vary and parallelism can be explored.

- Example: Hadoop which implements MapReduce.

3.7.8 Scaling Transparency

- Allows the system and applications to expand in scale without change to the system structure or the application algorithms.
- Example: World-Wide-Web
- Example: Distributed Database

4. Design Issues

- Specific issues for distributed systems:
  - Naming
  - Communication
  - Software structure
  - System architecture
  - Workload allocation
  - Consistency maintenance

4.1 Naming

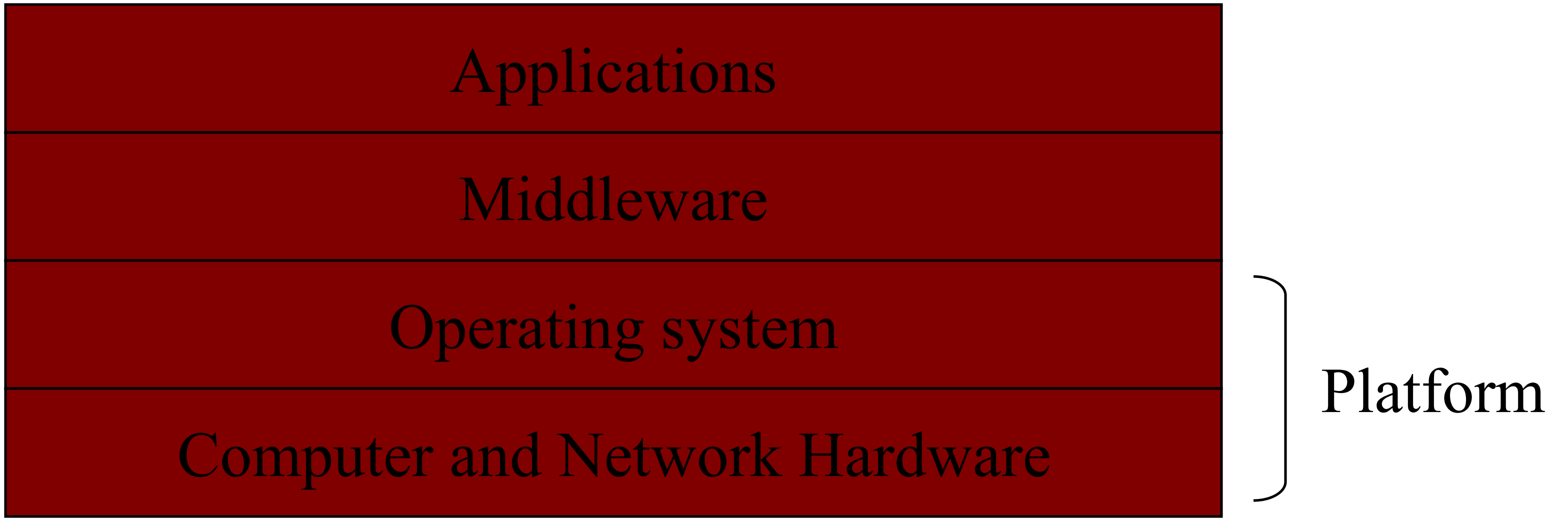
- A name is resolved when translated into an interpretable form for resource/object reference.
  - Communication identifier (IP address + port number)
  - Name resolution involves several translation steps
- Design considerations
  - Choice of name space for each resource type
  - Name service to resolve resource names to comm. id.
- Name services include naming context resolution, hierarchical structure, resource protection

4.2 Communication

- Separated components communicate with sending processes and receiving processes for *data transfer* and *synchronization.*
- Message passing: *send* and *receive* primitives
  - synchronous or blocking
  - asynchronous or non-blocking
  - Abstractions defined: channels, sockets, ports.
- Communication patterns: client-server communication (e.g., RPC, function shipping) and group multicast

4.3 Software Structure

- Layers in centralized computer systems:



Applications

Middleware

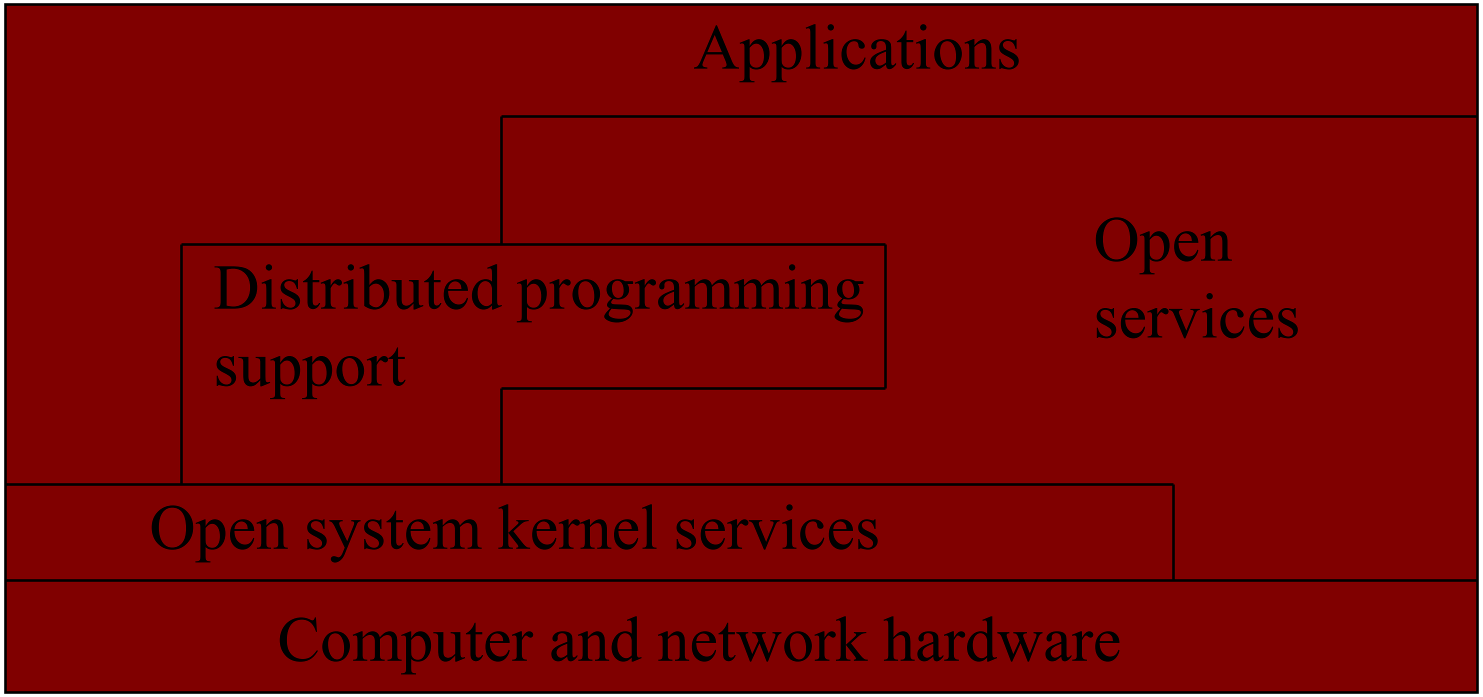
Operating system

Computer and Network Hardware

Platform

4.3 Software Structure

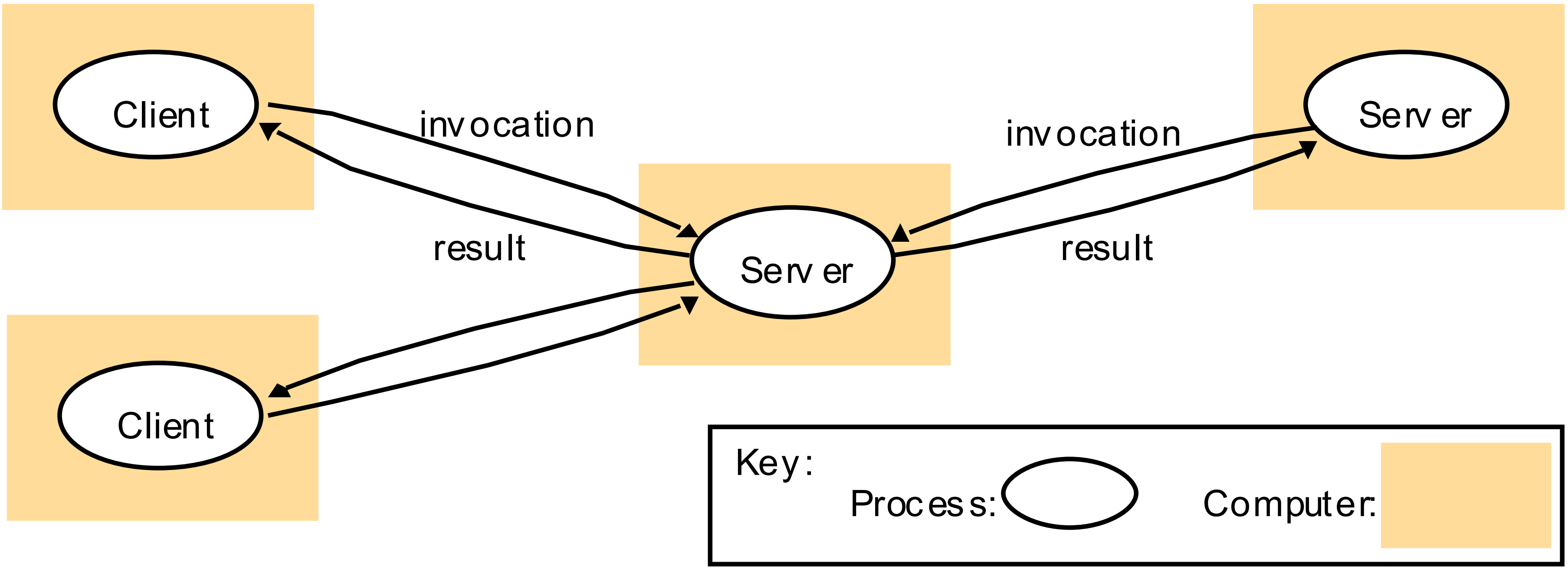
- Layers and dependencies in distributed systems:


Applications
Distributed programming support
Open services
Open system kernel services
Computer and network hardware

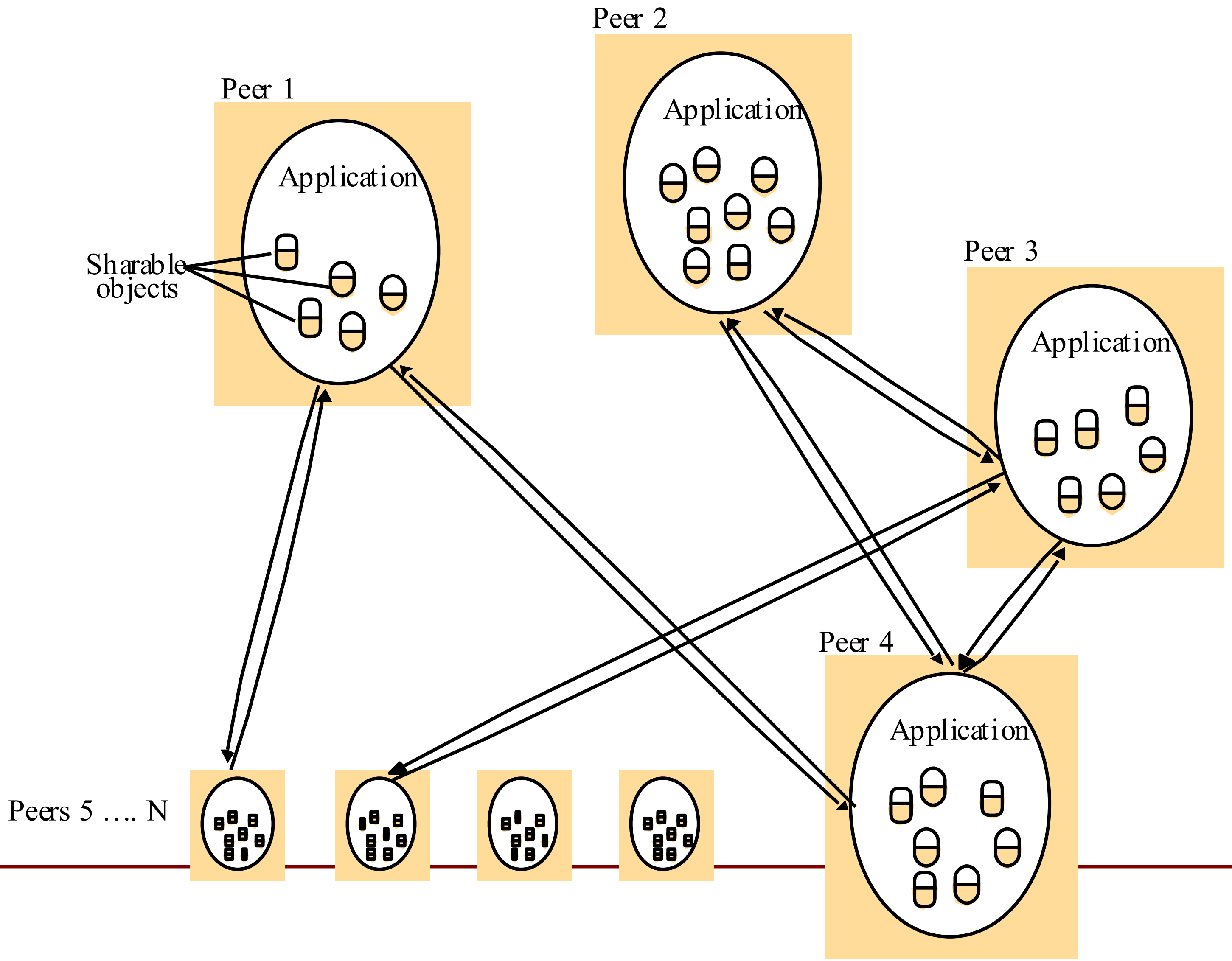
4.4 System Architectures

- Client-server
- Peer-to-peer
- Services provided by multiple servers
- Proxy servers and caches
- Mobile code and mobile agents
- Network computers
- Thin clients and mobile devices

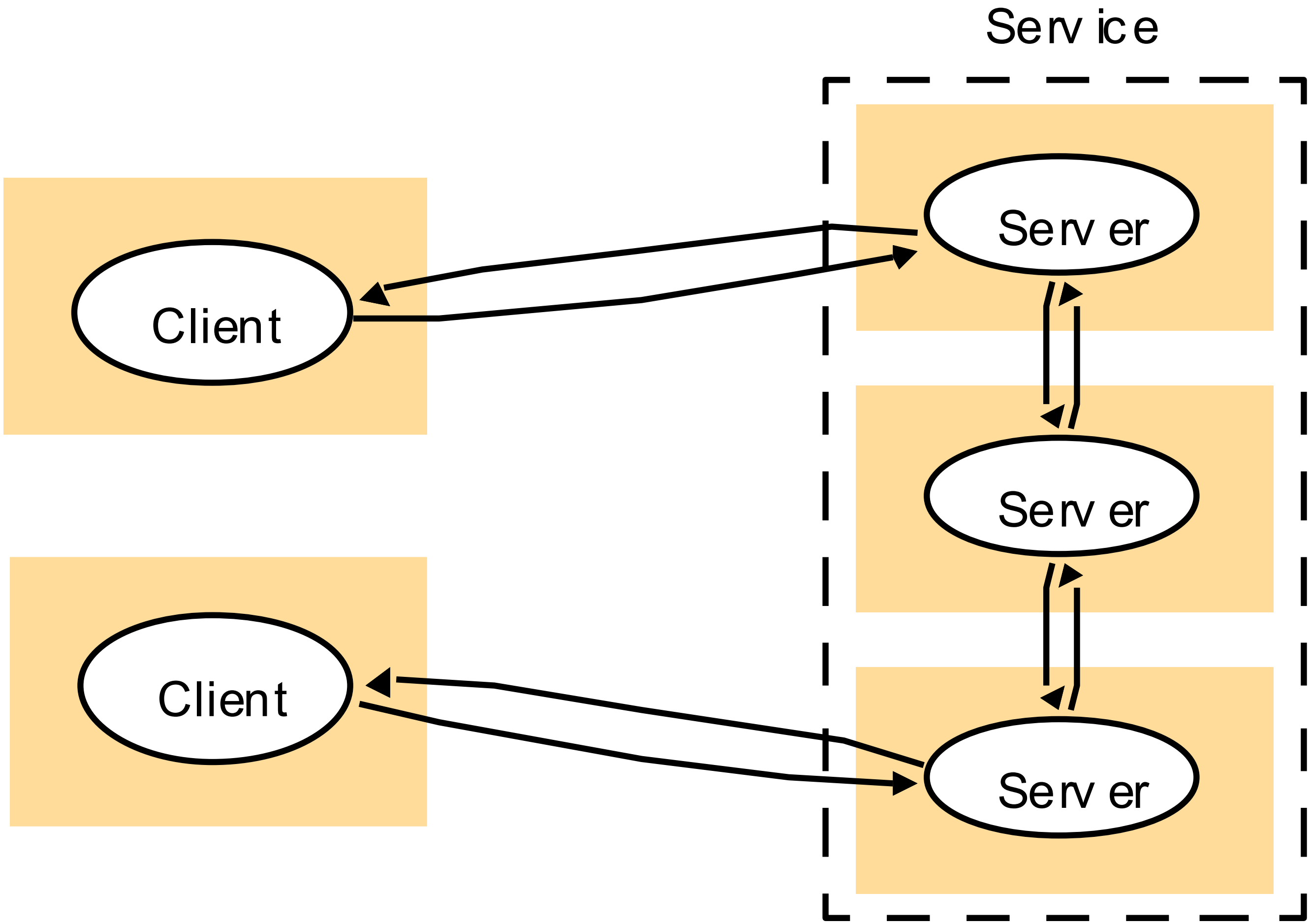
4.4.1 Clients Invoke Individual Servers



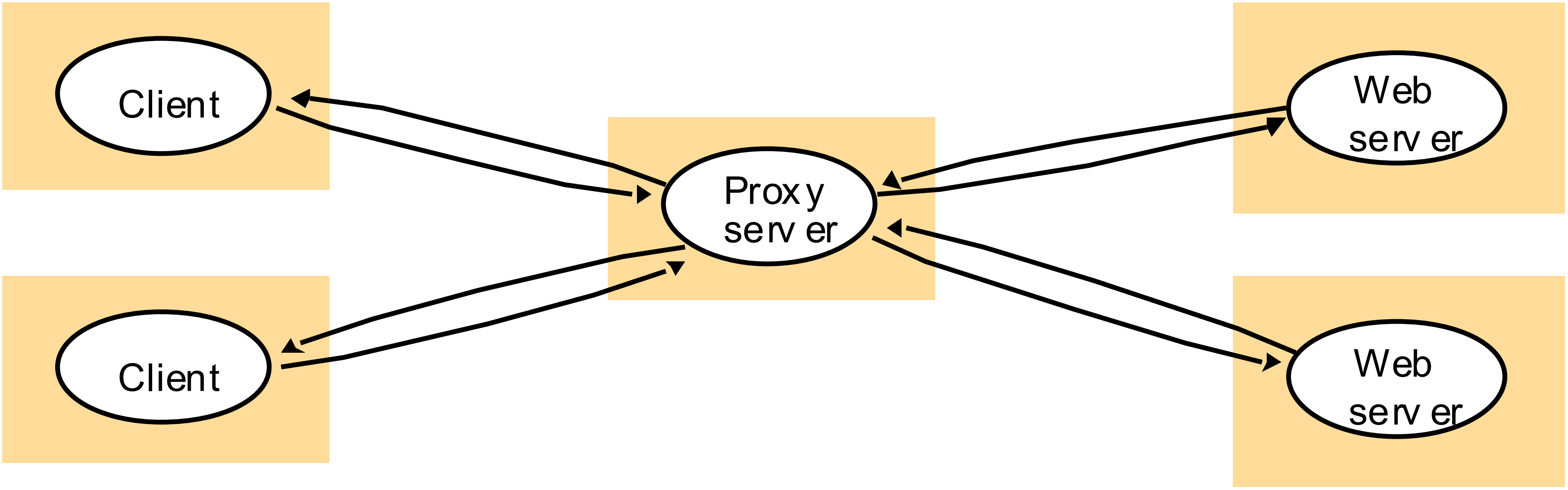
4.4.2 Peer-to-peer Systems



4.4.3 A Service by Multiple Servers

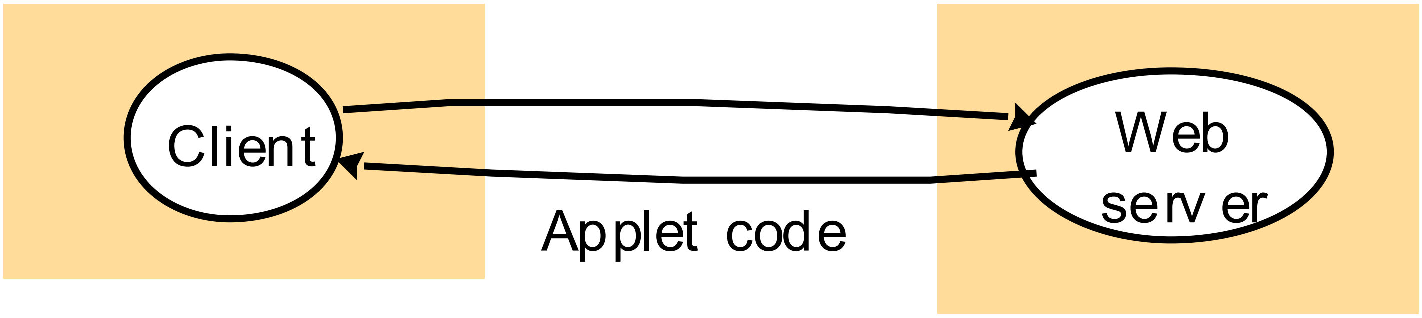


4.4.4 Web Proxy Server

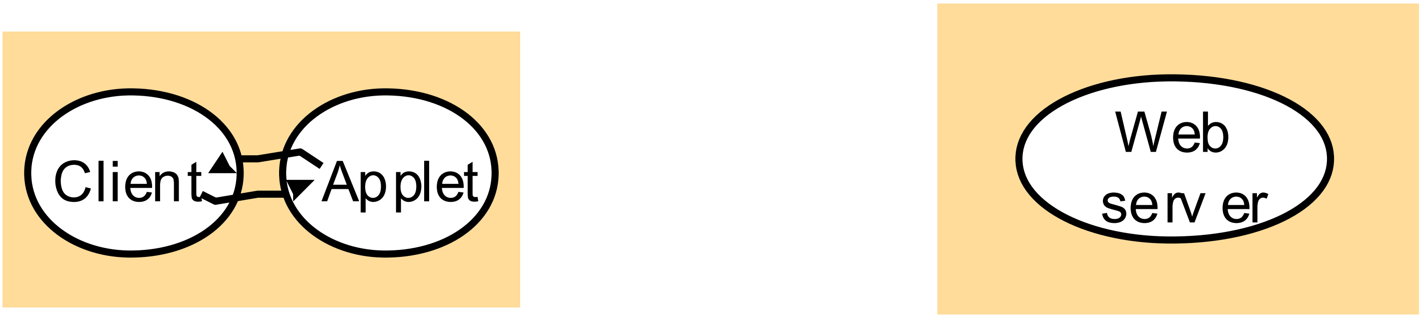


4.4.5 Web Applets

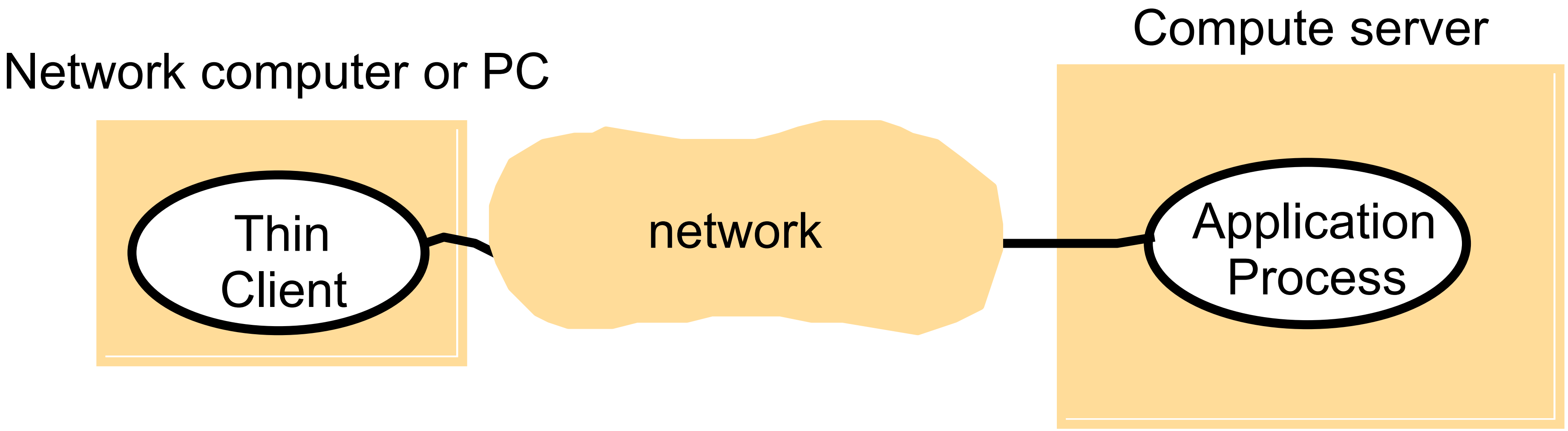
a) client request results in the downloading of applet code



b) client interacts with the applet



4.4.6 Thin Clients and Compute Servers



5. Summary

- Definitions of distributed systems and comparisons to centralized systems.
- The characteristics of distributed systems.
- The eight forms of transparency.
- The basic design issues.
- Read Chapter 1 and Chapter 2 of the textbook.