Chapter 1

Characterization of Distributed Systems

Definition

A distributed system is one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages.

Tightly-coupled system

Homegeneity

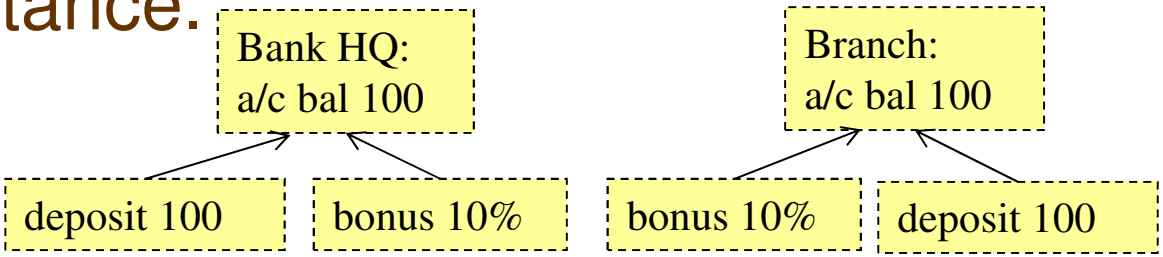
Loosely-coupled system

Heterogeneity

Autonomy

Characteristics

- Computers connected by a network may be spatially separated by any distance.



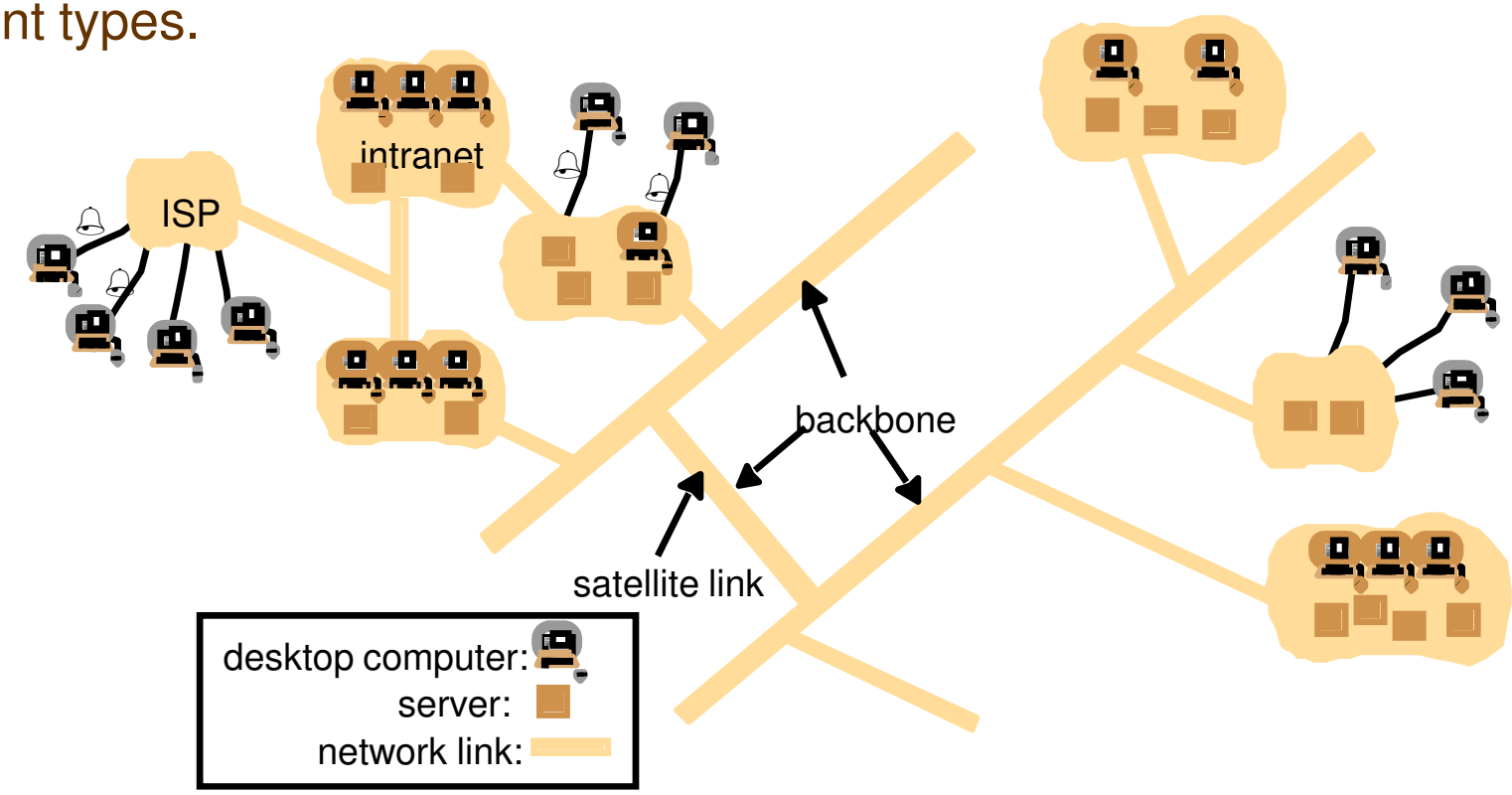
- This leads to
  - Concurrency – Coordination and handling of shared resources is required.
  - No global clock – Coordination of programs' actions depends on the shared idea of time, but computers in a network cannot accurately synchronize their clocks.
  - Independent failure – Each component of the system (e.g. network, computer, program) can fail independently, leaving the others still running.

Motivations

- The desire to shared resources
  - Hardware – Disks, printers etc.
  - Software – Files, databases, data objects of all kinds.
- Cheaper and more powerful hardware
- High-speed network technology at moderate cost

Example: Internet

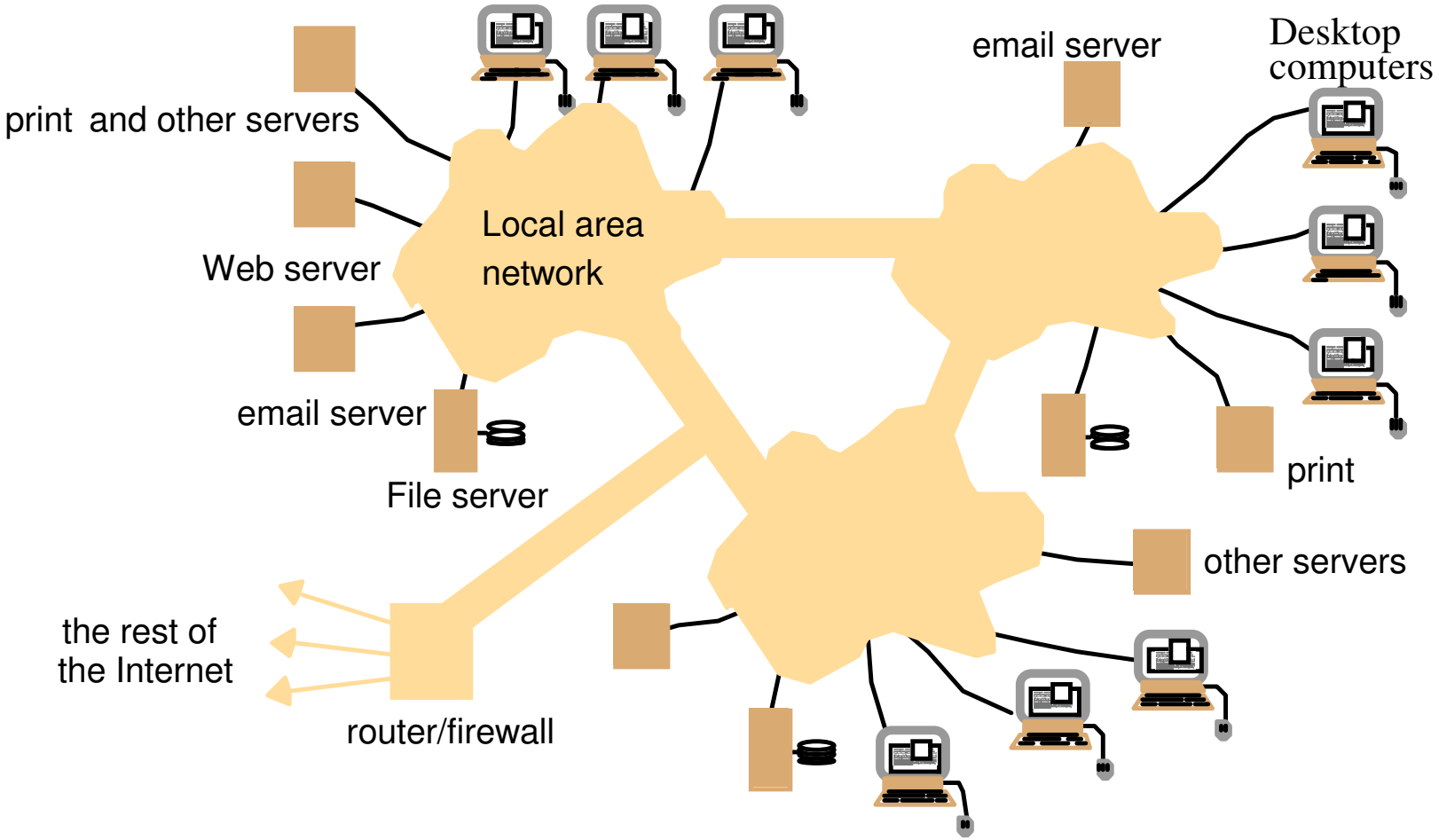
- Internet is a vast interconnected collection of computer networks of many different types.



- Passing of messages is achieved through a common means of communication (Internet protocols).
- Resources and services are shared (c.f. WWW, email, file transfer).

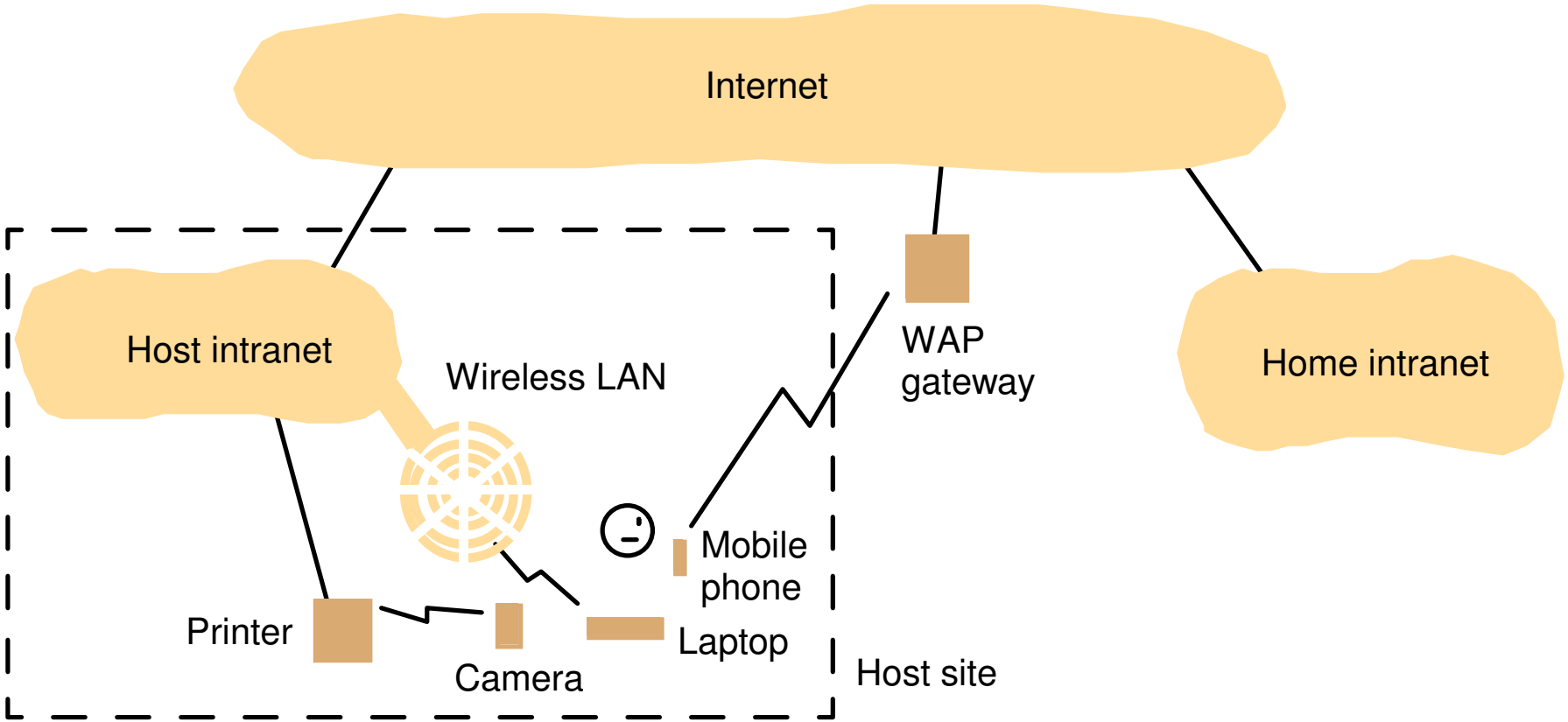
Example: Intranet

- Intranet is a network that is operated by companies and other organizations and local security policies can be enforced.



Example: Mobile Computing

- It is a result from device miniaturization and wireless computing.
- It is about portability of devices and ability to connect to networks in different places.



Example: Ubiquitous Computing

- It is also a result from device miniaturization and wireless networking.
- It suggests that small computing devices become pervasive in everyday objects that are scarcely noticed (e.g. washing machine, fridge).

Resource Sharing (1)

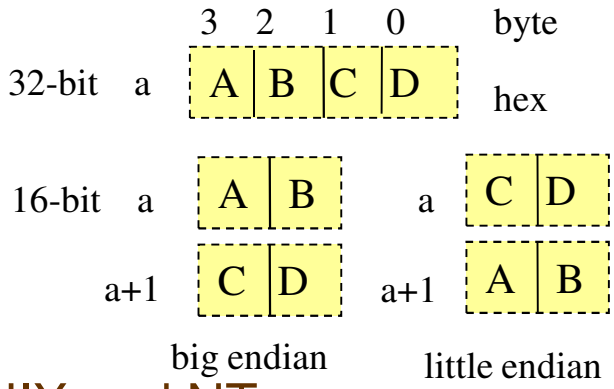
- Users concern sharing of higher level resources (e.g. DB, Web pages) more than sharing of hardware for cost reduction (e.g. disks or processors on which those are implemented).
- **Service** is a part of a computer system that manages a collection of related resources and presents their functionality to users and applications.
  - Resource access is restricted to a set of operations the service exports.
  - It is managed by a program that offers a communication interface for reliable access and consistent update.
  - Example: File service, printing service, electronic payment service

Resource Sharing (2)

- **Server** is a process on a computer that accepts requests, performs service, and responds.
- **Client** is a process on another computer that sends a message to request service.
- The terms refer to roles; the same process may be both client and server.
- This is a **remote invocation**.

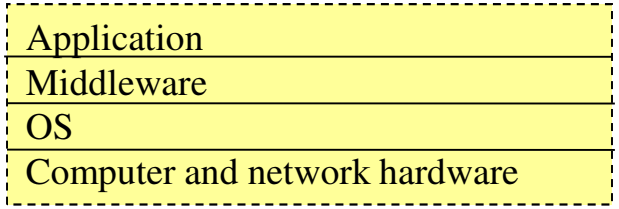
Challenge: Heterogeneity

- Networks
  - Different networks have an implementation of the common Internet protocols.
- Computer hardware
  - Two alternatives for byte ordering of integers
- Operating systems
  - Different APIs for exchange-message call in UNIX and NT
- Programming languages
  - Different presentations for characters and data structures such as arrays and records.
- Implementations by different developers
  - Need to use common standards (e.g. network communication, representation of primitive data items and data structures in messages).



row major vs. column major arrays

Middleware



- A software layer that provides
  - Programming abstraction
    - Example: CORBA and Java RMI provides remote object invocation.
  - Masking heterogeneity of the underlying networks, hardware, operating systems, and programming languages.
    - Implemented over Internet protocols
    - Deal with operating systems and hardware
- Mobile code from one computer to run on another (e.g. applet) overcomes heterogeneity by
  - Generation of code for a virtual machine instead of a particular hardware code
  - Implementation of the virtual machine for each type of hardware

Challenge: Openness

- It is determined by the degree to which new resource sharing services can be added and be made available for use by a variety of client programs.
- An **open distributed system** is based on uniform communication mechanism and published interfaces for access to shared resources.
- The system can be extended
  - Hardware level – Adding computers
  - Software level – Introducing new services or re-implementing old ones
  - Independence from individual vendors.

Challenge: Security

- Components of security of information resources
  - Confidentiality – Protection against disclosure to unauthorized individuals
  - Integrity – Protection against alteration or corruption
  - Availability – Protection against interference with resource access
- Concealing the content of the message sent over a network (e.g. credit card number) and ensuring the identity of the participants (e.g. shop).
  - Encryption techniques
- Denial of service attack – Sending of a large number of pointless requests to a service so that users cannot use it.
- Security of mobile code – Receiving executable to run (e.g. email attachment)

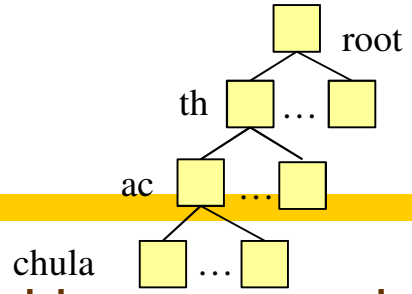
Challenge: Scalability

- A system is **scalable** if it will remain effective when there is a significant increase in the number of resources and users.
- Internet is scaling up.

Date	*Computers*	*Web servers*	*Percentage*
1993, July	1,776,000	130	0.008
1995, July	6,642,000	23,500	0.4
1997, July	19,540,000	1,203,096	6
1999, July	56,218,000	6,598,697	12

When system grows

www.chula.ac.th



- Cost of extending physical resources should be reasonable.
  - Adding a server when demand for a resource access grows.
- Performance loss should be predicted.
  - Hierarchic structure scales better than linear structure when lookup table grows (e.g. DNS).
  - Instead of a single master table, DNS partitions into many name tables dispersed throughout the Internet and administered locally.
  - Some resources are cached or replicated if accessed frequently (e.g. Web pages)
- Running out of software resources should be predicted.
  - Internet address has moved from 32-bit long (IPV4) to 128-bit long (IPV6) to accommodate more addressable hosts.
  - It is difficult to predict the demand; over-compensating for future growth may be worse than adopting change (e.g. large Internet address occupies more space).

domain name -> IP address

Challenge: Failure Handling

- Faults that occur in hardware or software may cause programs to produce incorrect results or stop abruptly.
- Failures are partial.
  - Some components fail while others continue to function.
  - This is particularly difficult to handle.
- But distributed system provides high **availability**.
  - Availability is a measure of the proportion of time that the system is available for use.
  - When one component fails, only the work that uses that component is affected.
  - The user may start work on another computer.

Techniques to handle failures

- Detecting
  - Checksum to detect corrupted data in messages or files
- Masking – to hide or make it less severe
  - Retransmission of messages
  - Dropping of corrupted data
  - Replication of data

Omission failure

- Tolerating
  - Browser informs user that it cannot contact Web server, leaving them free to try again later.
- Recovery
  - Data are brought back to consistent state after a server has crashed.
- Redundancy
  - At least two different routes between two communicating components
  - At least two replicas for data

Challenge: Concurrency

- Several clients may attempt to access a shared resource at the same time.
- The shared resource can be managed to take one client request at a time, but that limits throughput.
- The shared resource must be managed to ensure that concurrent accesses are synchronized and do not interfere with each other (e.g. seat reservation).

Challenge: Transparency (1)

- Transparency is concealment from the user and application programmer of the separation of components.
- The system is perceived as a whole rather than as a collection of independent components.
- Sometimes this is not required (e.g. a file is required to print at a particular printer rather than at any printers).
- **Access transparency**
  - Local and remote resources are accessed using identical operations.
- **Location transparency**
  - Resources are accessed without knowledge of their location.
- These two transparencies are the most basic and referred together as **network transparency.**

Transparency (2)

- **Concurrency transparency**
  - Processes operate concurrently using shared resources without interference between them.
- **Replication transparency**
  - Multiple instances of resources are used for reliability and performance without knowledge of the replicas by users or application programmers.
- **Failure transparency**
  - Concealment of faults to allow users and application programmers to complete their tasks despite failure of hardware or software.

Transparency (3)

- **Mobility transparency**
  - Resources and clients can move within a system without affecting the operation of the users or programs.
- **Performance transparency**
  - The system can be reconfigured to improve performance as loads vary.
- **Scaling transparency**
  - The system and applications can expand in scale without change to the system structure or the application algorithms.

Example of Transparency (1)

- GUI with file folders is the same whether the files are local or remote (access transparency).
- Access files using ftp program (no access transparency).
- Domain name in URL refers to computer name rather than Internet address (location transparency).
- Email (network transparency)
  - username@domainname to reach the recipient whether at local or remote.
  - Procedure to send a message does not depend on the location of the recipient.

Example of Transparency (2)

- Attempt to retransmit email message when servers or links fail, even if it takes several days (failure transparency).
- Caller's phone (client) and callee's phone (resource) are both moving from one environment (cell) to another when users are travelling. Users are not aware of the mobility of the phones (mobility transparency).