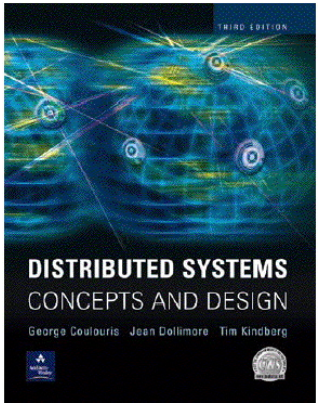
CSC 458: Distributed Systems

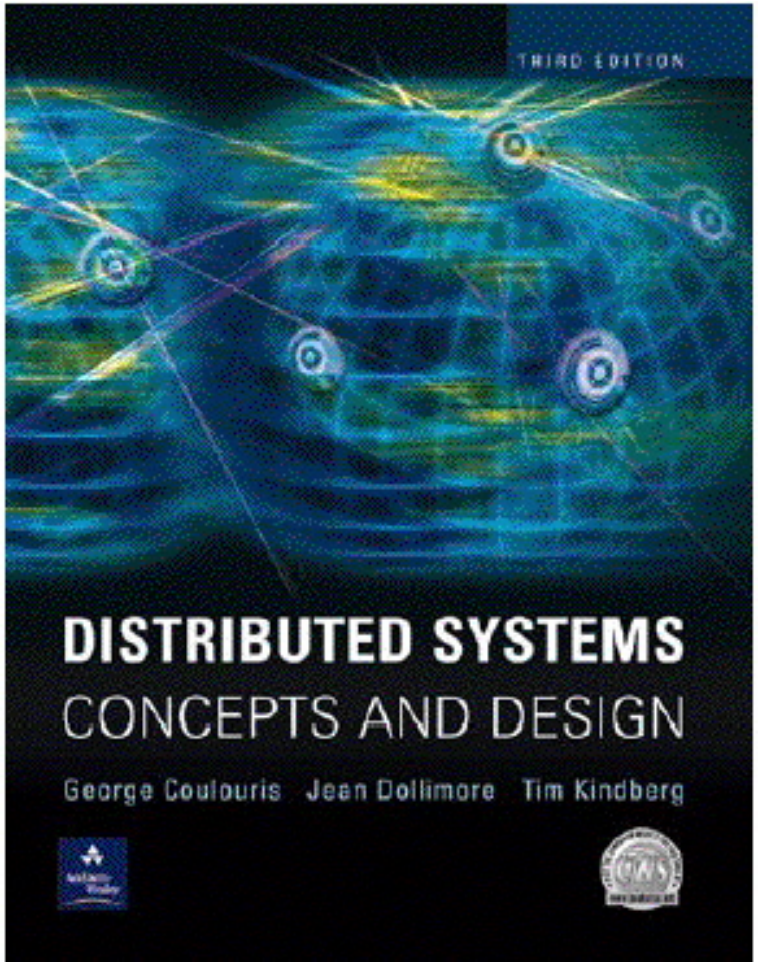
Dr. Gannouni Sofien

Introduction to Distributed Systems and Characterisation

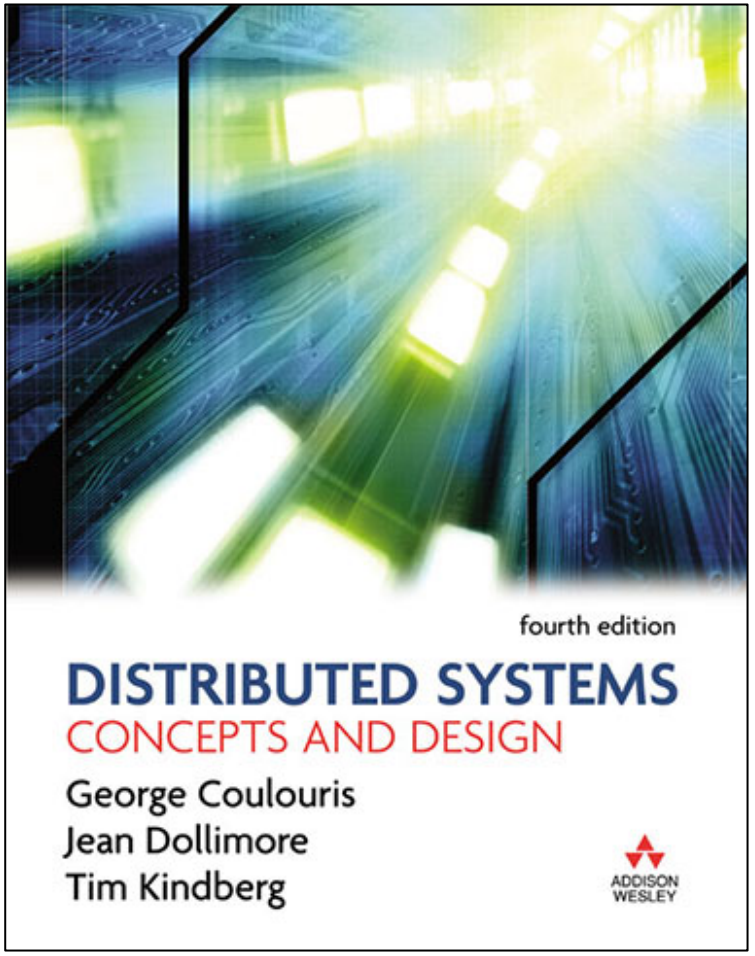


Most concepts are drawn from Chapter 1

Text Book

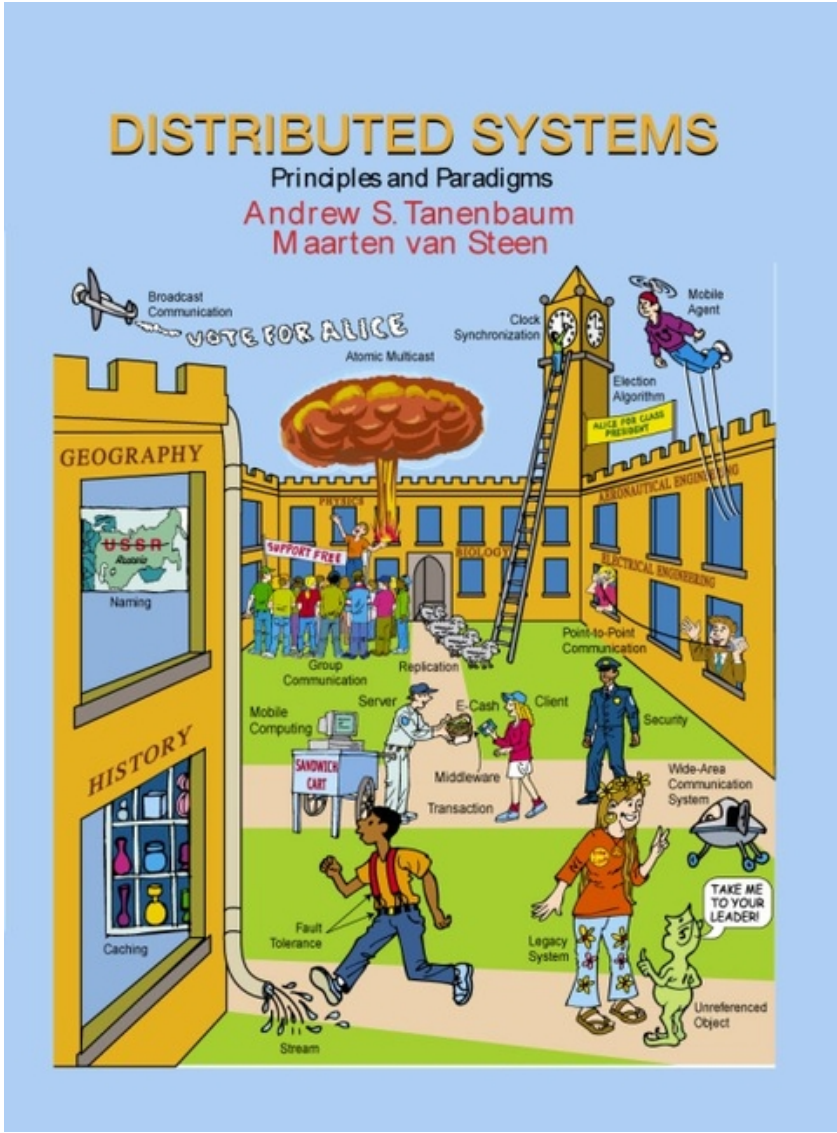


OR

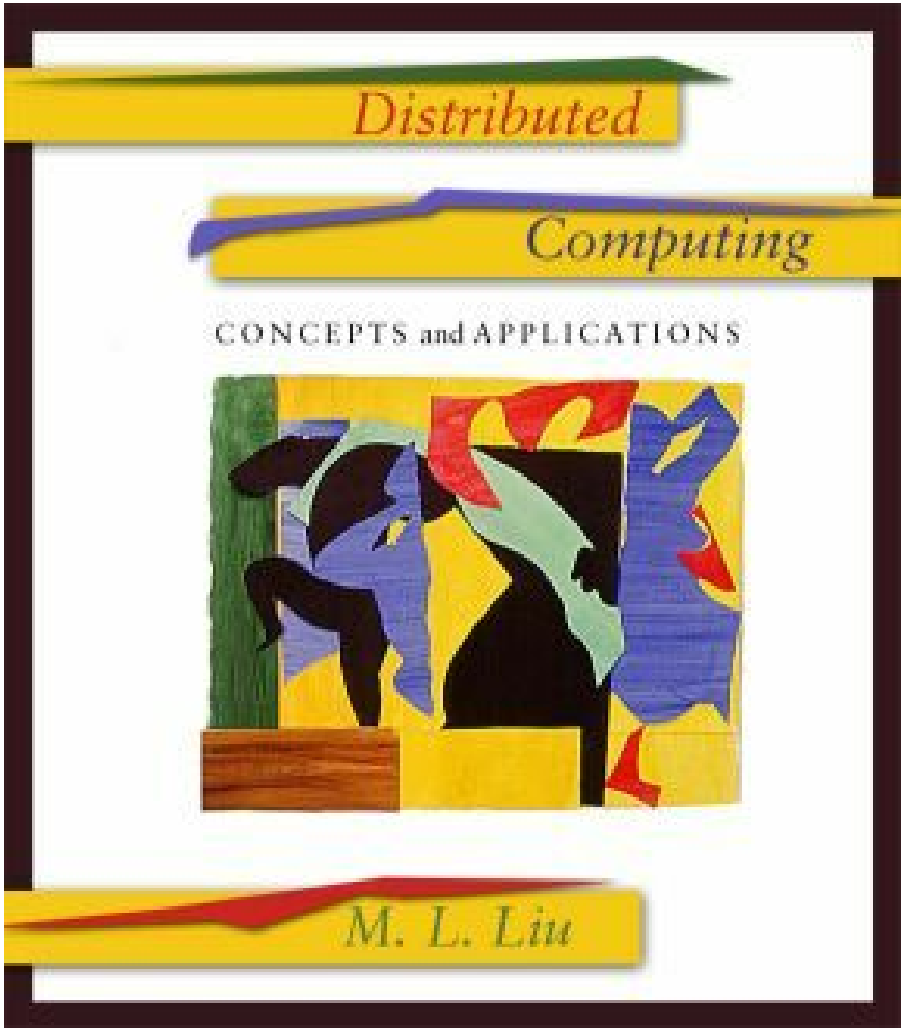


4th edition published in June 2005.
It has extra chapters: P2P, Grids, Web Services.

Reference Book – Alternate Text Book



Programming Reference



Presentation Outline

- Introduction
- Defining Distributed Systems
- Characteristics of Distributed Systems
- Example Distributed Systems
- Challenges of Distributed Systems
- Summary

Introduction

- Networks of computers are everywhere!
  - Mobile phone networks
  - Corporate networks
  - Factory networks
  - Campus networks
  - Home networks
  - In-car networks
  - On board networks in aero planes and trains

- This subject aims:
  - to present the main concepts and techniques that have been developed to help in the tasks of designing and implementing systems and applications that are based on networks.

Distributed Systems

What is a Distributed System?

- Tanenbaum and van Renesse: A distributed system is one that looks to its users like an ordinary, centralized, system but runs on multiple independent CPUs

- Symptoms? Shroeder:
  - Multiple, independent processing units
  - Processors communicate via a hardware interconnect
  - Processing unit failures are independent
  - Manage resource sharing
  - State is shared among processors

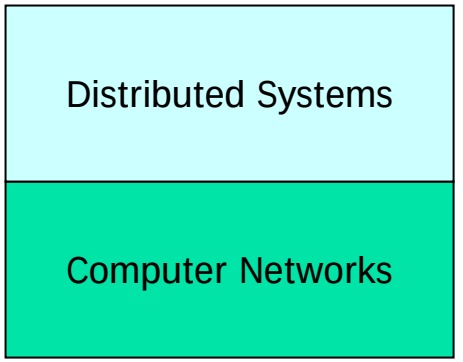
Defining Distributed Systems

- “A system in which hardware or software components located at networked computers communicate and coordinate their actions only by message passing.” [Coulouris]

- “A distributed system is a collection of independent computers that appear to the users of the system as a single computer.” [Tanenbaum]

Networks vs. Distributed Systems

- Networks:
  - A media for interconnecting local and wide area computers and exchange messages based on protocols.
  - Network entities are visible and they are explicitly addressed (IP address).
  - the autonomous computers are explicitly visible (have to be explicitly addressed)
- Distributed System:
  - existence of multiple autonomous computers is transparent
- However,
  - many problems in common, but at different levels.
    - Networks focuses on packets, routing, etc., whereas distributed systems focus on applications.
  - in some sense networks (or parts of them, e.g., name services) are also distributed systems, and
  - Every distributed system relies on services provided by a computer network.



Reasons for Distributed Systems

- Replication of processing power:
  - independent processors working on the same task
    - distributed systems consisting of collections of microcomputers may have processing powers that no supercomputer will ever achieve
      - 10000 CPUs, each running at 50 MIPS, yields 500000 MIPS,
- Physical separation:
  - systems that rely on the fact that computers are physically separated (e.g., to satisfy reliability requirements).
- Functional Separation:
  - Existence of computers with different capability and purpose:
  - Clients and Servers
  - Data collection and data processing

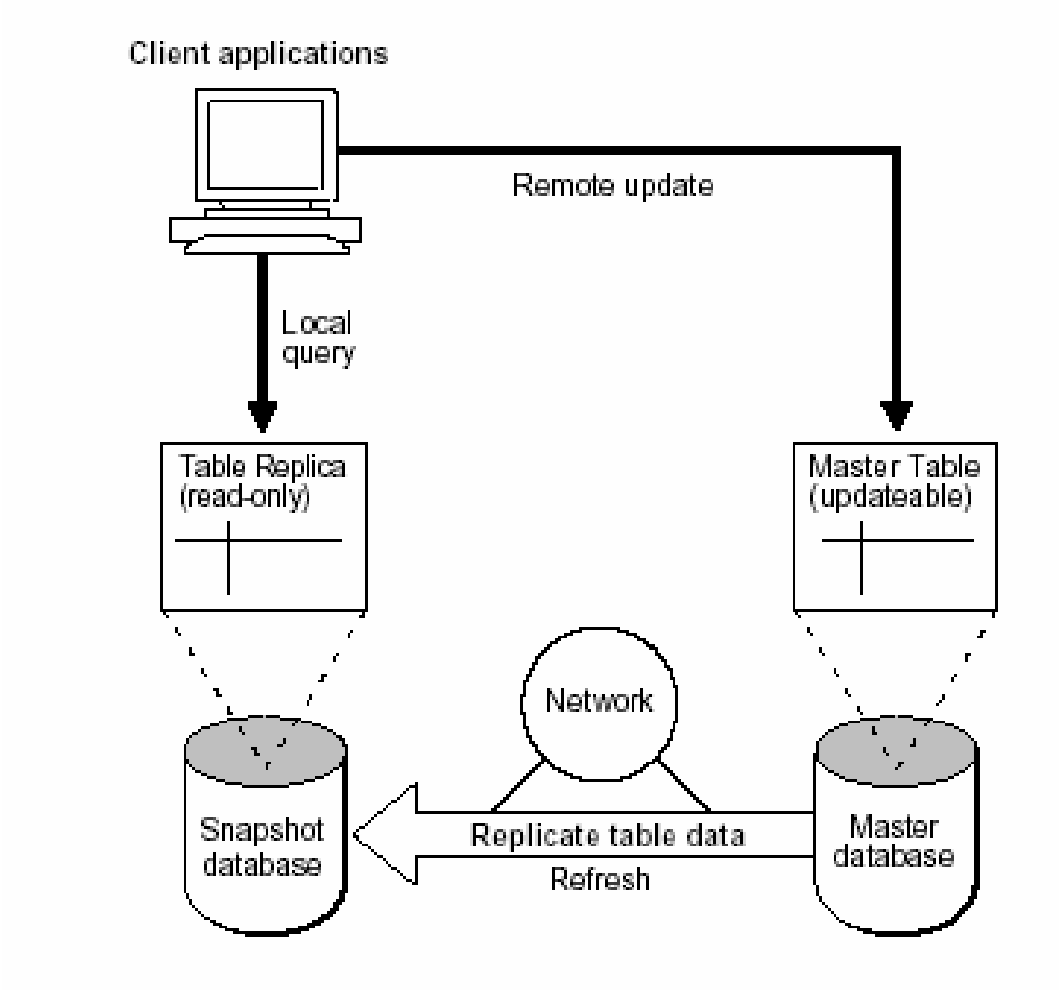
Reasons for Distributed Systems

- Inherent distribution:
  - Information:
    - Different information is created and maintained by different persons (e.g., Web pages)
  - People
    - Computer supported collaborative work (virtual teams, engineering, virtual surgery)
  - Retail store and inventory systems for supermarket chain (e.g., Al-Jazeera, Carrefour)
- Power imbalance and load variation:
  - Distribute computational load among different computers.
- Reliability:
  - Long term preservation and data backup (replication) at different location.
- Economies:
  - Sharing a printer by many users and reduce the cost of ownership.
  - Building a supercomputer out of a network of computers.
  - collections of microprocessors offer a better price/performance ration than large mainframes
    - mainframes: 10 times faster, 1000 times as expensive

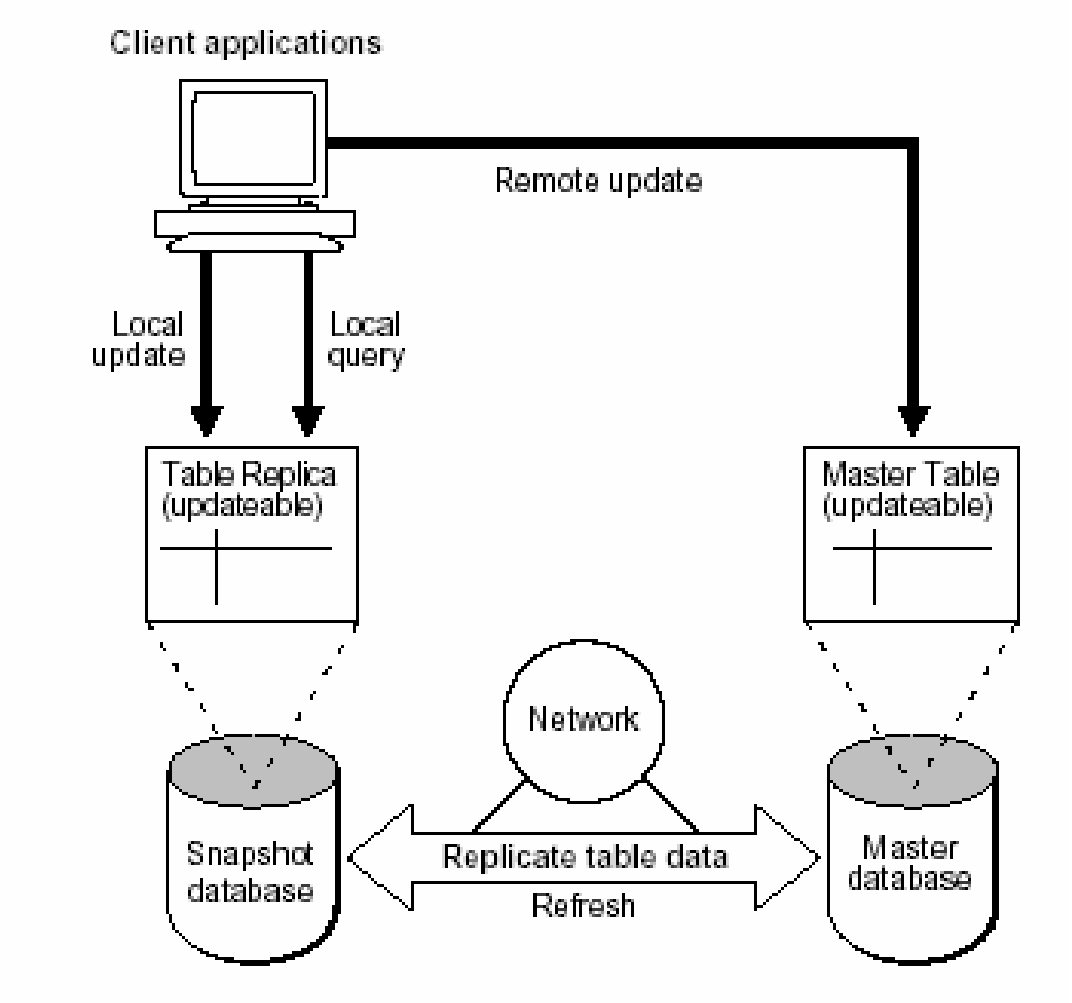
What Distributing

- Control (processing)
  - The control is distributed if there is no static hierarchical relationships between processes.
    - No master process and Slave processes
- Data
  - Replication
    - Symmetric replication
    - Asymmetric replication
  - Fragmentation
    - Horizontal Fragmentation
    - Vertical Fragmentation

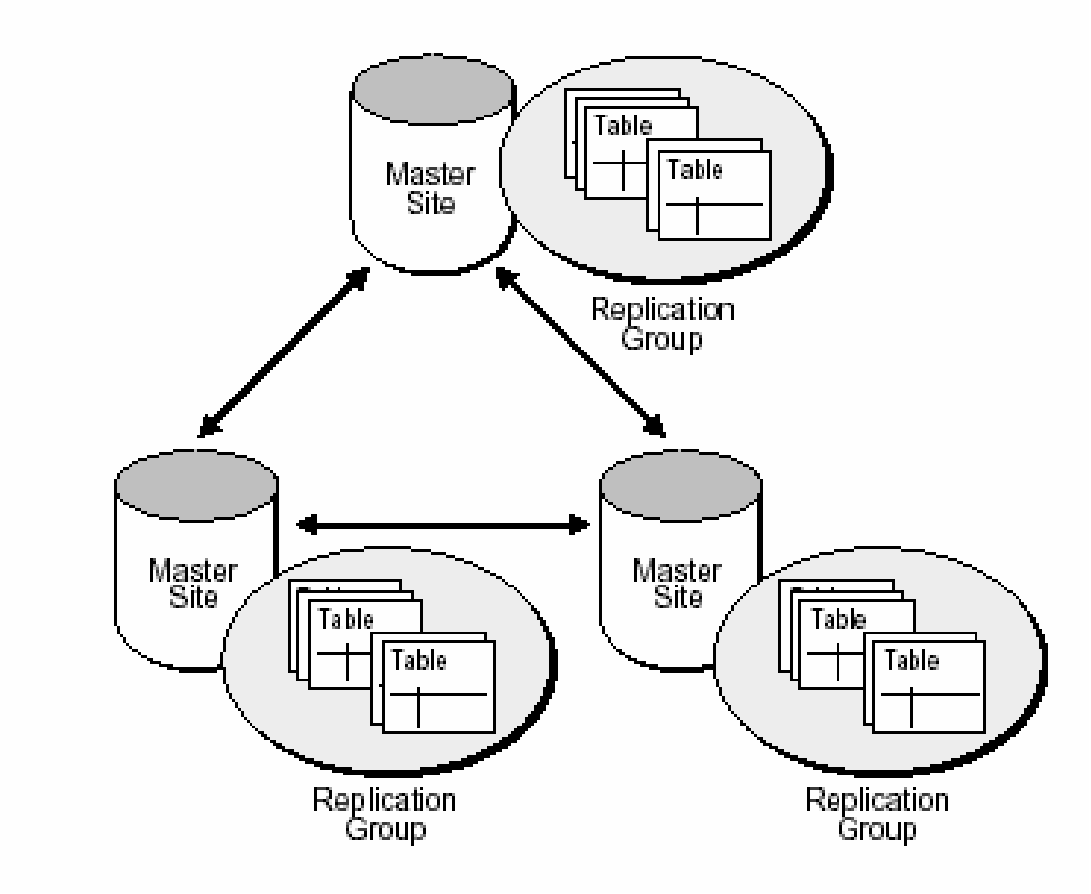
Asymmetric Replication



Symmetric Replication



Symmetric Replication



Fragmentation Alternatives

- Horizontal fragmentation
  - **Store Whole Tuples on Different machines.**
  - Uses the restriction relation algebra's operator
- Vertical fragmentation
  - **Store Different Fields of the same tuples on Different machines**.
  - Uses the projection relation algebra's operator
  - requires redundant storage of at least one primary key per tuple

Correctness of Fragmentation

- Completeness
  - Decomposition of a relation R into fragments R1, …, Rn is complete if each data item in R can also be found in some Ri.
- Reconstruction
  - If a relation R is decomposed into fragments R1, …, Rn then there should exist some relational operator Op such that:
    - R = Op Ri(ì:1..n)
- Disjointness
  - If a relation R is decomposed into fragments R1, …, Rn and a data item dj in Rj, then dj should not be in any other fragment Rk (k!=j) .

Consequences of Distributed Systems

- Computers in distributed systems may be on separate continents, in the same building, or the same room. DS have the following consequences:
  - Concurrency – each system is autonomous.
    - Carry out tasks independently
    - Tasks coordinate their actions by exchanges messages.
  - Heterogeneity
  - No global clock
  - Independent Failures

Characteristics of distributed systems

- Parallel activities
  - Autonomous components executing collaborative tasks
- Concurrent activities
  - Autonomous components executing concurrent tasks
- Communication via message passing
  - No shared memory
- Resource sharing
  - Printer, database, other services
- No global state
  - No single process can have knowledge of the current global state of the system
- No global clock
  - Only limited precision for processes to synchronize their clocks

Goals of Distributed Systems

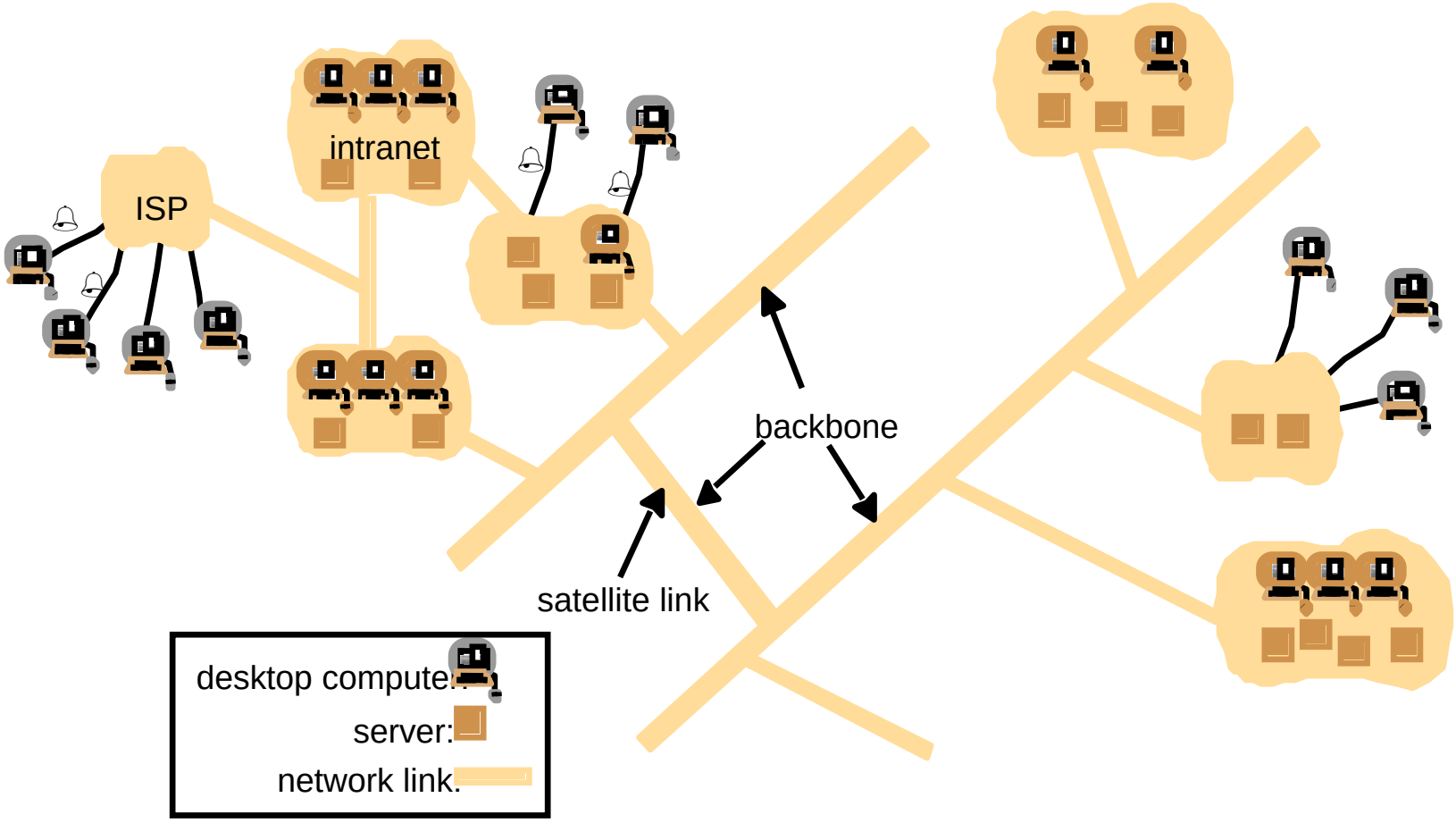
- Connecting Users and Resources
- Transparency
- Openness
- Scalability
- Enhanced Availability

Examples of Distributed Systems

- They are based on familiar and widely used computer networks:
  - Internet
  - Intranets, and
  - wireless networks

A typical portion of the Internet and its services:

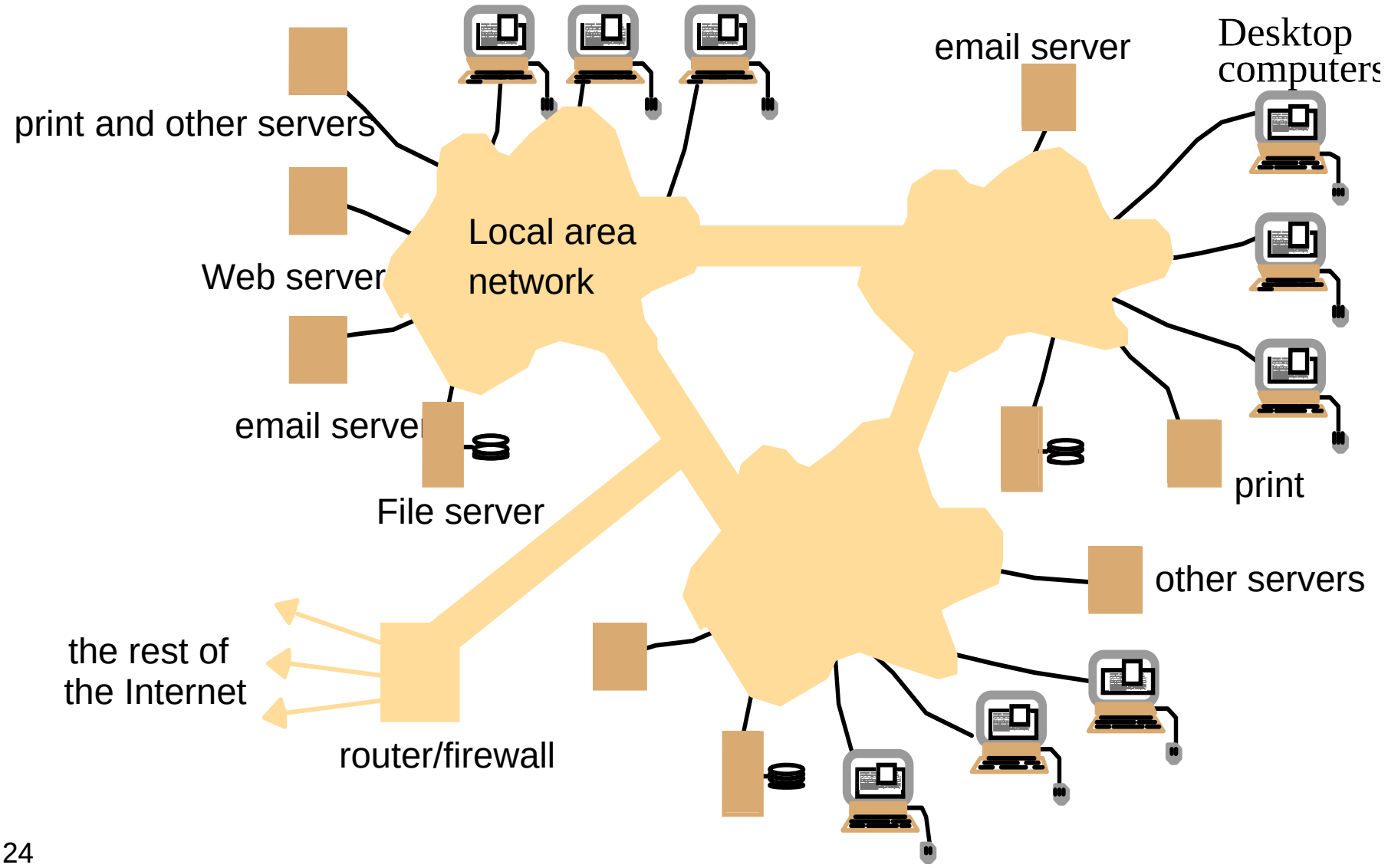
Multimedia services providing access to music, radio, TV channels, video conferencing and supporting several users.



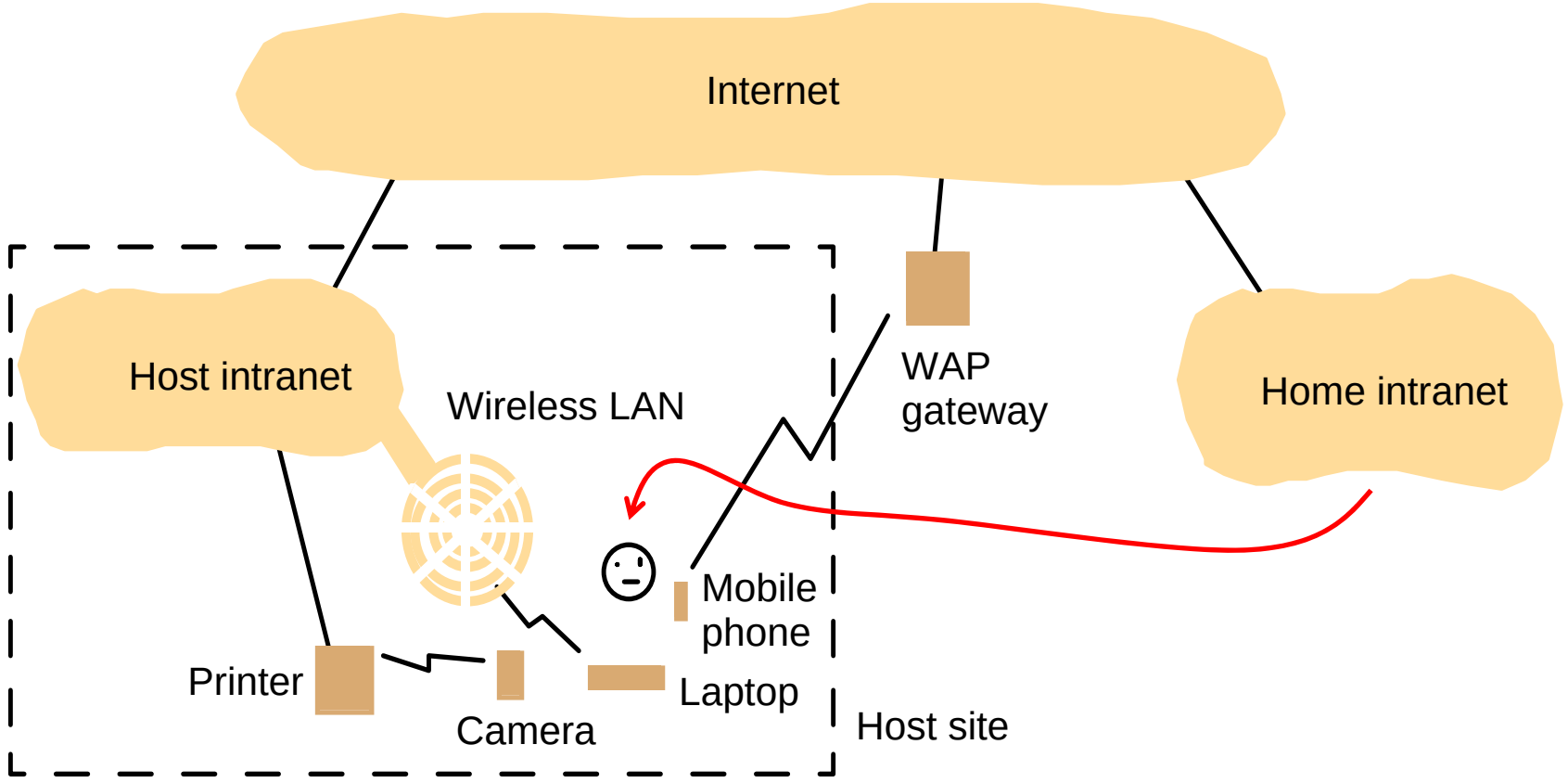
- The Internet is a vast collection of computer networks of many different types and hosts various types of services.

A typical intranet:

A portion of Internet that is separately administered & supports internal sharing of resources (file/storage systems and printers)



Mobile and ubiquitous computing: portable and handheld devices in a distributed system



- Support continued access to Home intranet resources via wireless and provision to utilise resources (e.g., printers) that are conveniently located (location-aware computing).

Business Example and Challenges

- Online bookstore (e.g. in the World Wide Web)
  - Customers can connect their computer to your computer (web server):
    - Browse your inventory
    - Place orders
    - …

This example Adopted from Torbin Weis, Berlin University of Technology

Business example – challenges I

- What if
  - Your customer uses a completely different hardware? (PC, MAC,…)
  - … a different operating system? (Windows, Unix,…)
  - … a different way of representing data? (ASCII, EBCDIC,…)
  - **Heterogeneity**
- Or
  - You want to move your business and computers to the Caribbean (because of the weather)?
  - Your client moves to the Caribbean (more likely)?
  - **Distribution transparency**

Business example – challenges II

- What if
  - Two customers want to order the same item at the same time?
  - **Concurrency**
- Or
  - The database with your inventory information crashes?
  - Your customers computer crashes in the middle of an order?
  - **Fault tolerance**

Business example – challenges III

- What if
  - Someone tries to break into your system to steal data?
  - … sniffs for information?
  - … your customer orders something and doesn't accept the delivery saying he didn't?
  - **Security**
- Or
  - You are so successful that millions of people are visiting your online store at the same time?
  - **Scalability**

Business example – challenges IV

- When building the system…
  - Do you want to write the whole software on your own (network, database,…)?
  - What about updates, new technologies?
  - **Reuse** and **Openness** (Standards)

Overview challenges I

- Heterogeneity
  - Heterogeneous components must be able to interoperate
- Distribution transparency
  - Distribution should be hidden from the user as much as possible
- Fault tolerance
  - Failure of a component (partial failure) should not result in failure of the whole system
- Scalability
  - System should work efficiently with an increasing number of users
  - System performance should increase with inclusion of additional resources.

Overview challenges II

- Concurrency
  - Shared access to resources must be possible
- Openness
  - Interfaces should be publicly available to ease adding new components
- Security
  - The system should only be used in the way intended

Heterogeneity

- Heterogeneous components must be able to interoperate
  - Operating systems
  - Hardware architectures
  - Communication architectures
  - Programming languages
  - Software interfaces
  - Security measures
  - Information representation

Distribution Transparency I

- To hide from the user and the application programmer of the separation/distribution of components, so that the system is perceived as a whole rather than a collection of independent components.
- ISO Reference Model for Open Distributed Processing (ODP) identifies the following forms of transparencies:
- Access transparency
  - Access to local or remote resources is identical
  - E.g. Network File System
- Location transparency
  - Access without knowledge of location
  - E.g. separation of domain name from machine address.
- Failure transparency
  - Tasks can be completed despite failures
  - E.g. message retransmission, failure of a Web server node should not bring down the website.

Distribution Transparency II

- Replication transparency
  - Access to replicated resources as if there was just one. And provide enhanced reliability and performance without knowledge of the replicas by users or application programmers.
- Migration (mobility/relocation) transparency
  - Allow the movement of resources and clients within a system without affection the operation of users or applications.
  - E.g. switching from one name server to another at runtime; migration of an agent/process form one node to another.

Distribution Transparency III

- Concurrency transparency
  - A process should not notice that there are other sharing the same resources
- Performance transparency:
  - Allows the system to be reconfigured to improve performance as loads vary.
  - E.g., dynamic addition/deletion of components. switching from linear structures to hierarchical structures when the number of users increase.
- Scaling transparency:
  - Allows the system and applications to expand in scale without change to the system structure or the application algorithms.
- Application level transparencies:
  - Persistence transparency
    - Masks the deactivation and reactivation of an object
  - Transaction transparency
    - Hides the coordination required to satisfy the transactional properties of operations

Fault tolerance

- Failure: an offered service no longer complies with its specification
- Fault: cause of a failure (e.g. failure of a component)
- Fault tolerance: no failure despite faults

Fault tolerance mechanisms

- Fault detection
  - Checksums, heartbeat, …
- Fault masking
  - Retransmission of corrupt messages, redundancy, …
- Fault toleration
  - Exception handling, timeouts,…
- Fault recovery
  - Rollback mechanisms,…

Scalability

- System should work efficiently at many different scales, ranging from a small Intranet to the Internet.
- Remain effective when there is a significant increase in the number of resources and the number of users.
- Challenges of designing scalable distributed systems:
  - Cost of physical resources
    - Cost should linearly increase with system size
  - Performance Loss
    - For example, in hierarchically structure data, search performance loss due to data growth should not be beyond O(*log n*), where n is the size of data.
  - Preventing software resources running out:
    - Numbers used to represent Internet address (32 bit->64bit).
  - Avoiding performance bottlenecks:
    - Use decentralized algorithms (centralized DNS to decentralized).

Concurrency

- Provide and manage concurrent access shared resources:
  - Fair scheduling
  - Preserve dependencies (e.g. distributed transactions)
  - Avoid deadlocks

Openness and Interoperability

- Open system:
  "... a system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered applications software to be ported across a wide range of systems with minimal changes, to interoperate with other applications on local and remote systems, and to interact with users in a style which facilitates user portability" (Guide to the POSIX Open Systems Environment, IEEE POSIX 1003.0).
- Open spec/standard developers - communities:
  - ANSI, IETF, W3C, ISO, IEEE, OMG, Trade associations,...

Security I

- The resources are accessible to authorized users and used in the way they are intended.
- Confidentiality
  - Protection against disclosure to authorized individual.
  - E.g. ACLs (access control lists) to provide authorized access to information.
- Integrity
  - Protection against alternation or corruption.
  - E.g. changing the account number or amount value in a money order

Security II

- Availability
  - Protection against interference with the means to access the resources.
  - E.g. denial of service attacks
- Non-repudiation
  - Proof of sending / receiving an information
  - E.g. digital signature

Security mechanisms

- Encryption
  - E.g. Blowfish, RSA
- Authentication
  - E.g. password, public key authentication
- Authorization
  - E.g. access control lists

Summary

- Distributed Systems are everywhere.
- The Internet enables users throughout the world to access its services wherever they are located.
- Resource sharing is the main motivating factors for constructing distributed systems.
- Construction of DS produces many challenges:
  - Heterogeneity, Openness, Security, Scalability, Failure handling, Concurrency, and Transparency.
- Distributed systems enable globalization:
  - Community (Virtual teams, organizations, social networks)
  - Science (e-Science)
  - Business (e-Bussiness)