Chapter 2
Distributed Information Systems

Presented by: Dr. R. BENTRCIA

Department of Computer Science, M'sila University

Outline

- Objectives
- Definitions
- Distributed Information Systems
  - Examples of distributed systems
  - Organization of a distributed system
  - Challenges of distributed systems development
  - Goals of distributed systems
  - Design of an information system

Objectives

- To realize the concepts of distributed systems and information systems.
- To know the design structure of distributed information systems.

Definitions

- **A distributed system** consists of a collection of **autonomous** computers linked by a computer network and equipped with distributed system software. This software enables computers to coordinate their activities and to share the resources of the system hardware, software, and data.
- A collection of independent computers that appears to its users as a single **coherent** system.
- A collection of autonomous computational entities conceived as a single coherent system by its designer.
- **Information system**, an integrated set of components for collecting, storing, and processing data and for providing information, knowledge, and digital products.

Distributed Information Systems

- At the very beginning, computers were huge & expensive machines.
- Later, microprocessor technology made computational entities more powerful and cheap.
- High-speed computer networks made interconnection of computational entities possible at a wide range of scales and speeds.
- As a result, computer systems changed dramatically from centralized (single-processor) systems to decentralized (multi-processor), distributed systems.

Distributed Information Systems

- A computer service that runs at a single central location is more likely to become unavailable than a service distributed to many sites.
- There are two ways in which a service can be made to run at many sites: replication of the service, and *distribution* of the service.
- We are investigating ways in which to build distributed services i.e. services that have distinct components, at many different sites, that collaborate to ensure the quality of service.

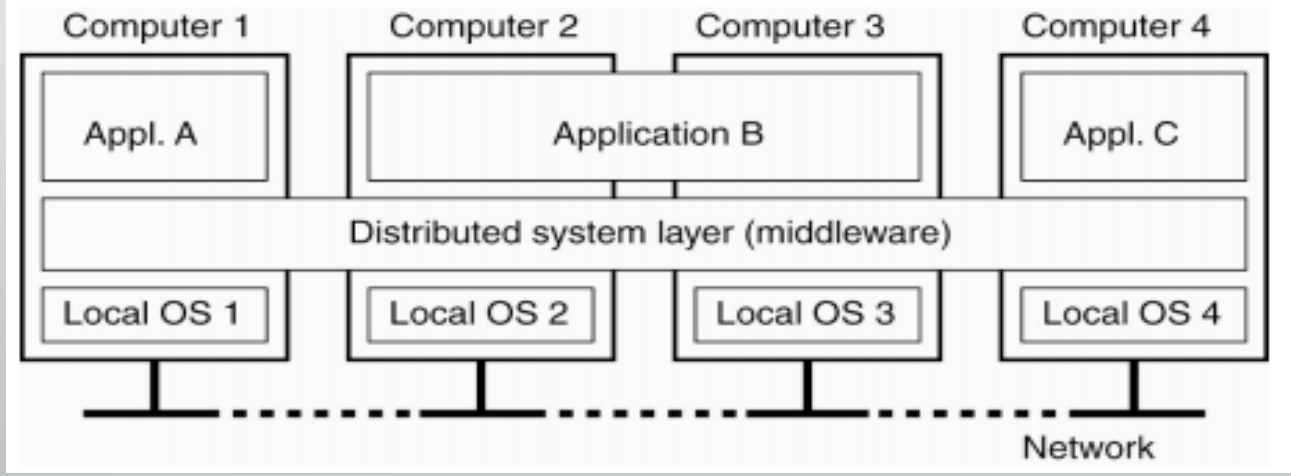
Examples of Distributed Systems

- Some distributed information systems are:

Finance and commerce	eCommerce e.g. Amazon and eBay, PayPal, online banking and trading
The information society	Web information and search engines, ebooks, Wikipedia; social networking: Facebook and MySpace
Creative industries and entertainment	online gaming, music and film in the home, user-generated content, e.g. YouTube, Flickr
Healthcare	health informatics, on online patient records, monitoring patients
Education	e-learning, virtual learning environments; distance learning
Transport and logistics	GPS in route finding systems, map services: Google Maps, Google Earth
Science	The Grid as an enabling technology for collaboration be-tween scientists
Environmental management	sensor technology to monitor earthquakes, floods or tsunamis

Organization of a Distributed System

- A distributed system is organized as a middleware. The *middleware* layer extends over multiple machines, and offers each application the same interface [Tanenbaum and van Steen, 2007].



Organization of a Distributed System

- False assumptions made by first time developer:
  - The network is reliable.
  - The network is secure.
  - The network is homogeneous.
  - The topology does not change.
  - Latency is zero.
  - Bandwidth is infinite.
  - Transport cost is zero.
  - There is one administrator.

These false assumptions typically produces all mistakes in the engineering of distributed systems.

Challenges of Distributed Systems Development

- What are the challenges of developing a distributed system?
  - Heterogeneity of their components.
  - Openness.
  - Security.
  - Scalability: the ability to work well when the load or the number of users increases.
  - Failure handling.
  - Concurrency of components.
  - Transparency.
  - Providing quality of service.

Goals of distributed systems

- Four goals for a distributed system:
  - Making (distributed, remote) resources available for use.
  - Allowing distribution of resources to be hidden whenever unnecessary.
  - Promoting openness.
  - Promoting scalability.

Making Resources Available

- Resources can be printers, scanners, storage devices, etc.
- The main goal of a distributed system is to make it easy for the users (and applications) to access remote resources, and to share them in a controlled and efficient way.

Heterogeneity

- Variety and differences in:
  - Networks.
  - Computer hardware.
  - Operating systems.
  - Programming languages implementations by different developers.
- Solution: Middleware as a software layer to provide a programming abstraction as well as masking the heterogeneity of the underlying networks, hardware, OS, and programming languages (e.g., CORBA, RMI, etc).

Distribution Transparency

- An important goal of a distributed system is to hide the fact that its processes and resources are physically distributed across multiple computers.
- A distributed system that is able to present itself to users and applications as if it were only a single computer system is said to be *transparent*.

Distribution Transparency

- The concept of *transparency* can be applied to several aspects of a distributed system:

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource is replicated
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource

Openness

- An *open* distributed system is a system that offers services according to standard rules that describe the syntax and semantics of those services.
- In distributed systems, services are generally specified through interfaces, which are often described in an *Interface Definition Language* (IDL).
- Interface definitions written in an IDL nearly always capture only the syntax of services. In other words, they specify precisely the names of the functions that are available together with types of the parameters, return values, possible exceptions that can be raised, and so on.

Security

- In a distributed system, clients send requests to access data managed by servers, such as:
  - 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).

Scalability

- *Scalability* is one of the most important design goals for developers of distributed systems.
- Scalability of a system can be measured along at least three different dimensions (Neuman, 1994).
  - First, a system can be scalable with respect to its *size*, meaning that we can easily add more users and resources to the system.
  - Second, a *geographically* scalable system is one in which the users and resources may lie far apart.
  - Third, a system can be *administratively* scalable, meaning that it can still be easy to manage even if it spans many independent administrative organizations.

Failure Handling (Fault Tolerance)

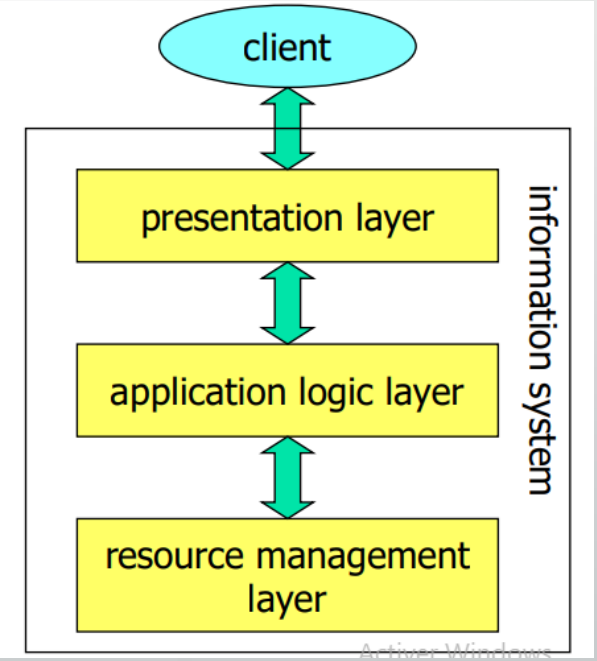
- Hardware, software and networks fail!
- Distributed systems must maintain availability even at low levels of hardware/ software/ network reliability.
- Fault tolerance is achieved by:
  - Recovery.
  - Redundancy.

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.

Design of an Information System

- Information systems consist of three layers: presentation, application logic, and resource management layer(RML).
- Responsibilities of the *presentation layer*:
  - Communication with the external entities (both humans and systems).
  - Information presentation.
  - Interaction with other systems by accepting operations and getting responses.
- Presentation layer can take various forms:
  - Graphical User Interface (GUI).
  - Module which formats a data set into a given syntactical representation.



Design of an Information System

- Responsibilities of the *application logic layer*:
  - Data processing behind the delivered results.
  - Implements actual operations submitted by clients through the presentation layer.
  - The operations may be referred to as the services.
- Depending on the complexity of the logic involved and of the selected implementation technique we can refer to this layer as:
  - Business processes,
  - Business rules,
  - Sometimes simply server.

Design of an Information System

- Responsibilities of the *Resource Management Layer (RML):*
  - Data management (databases, file systems, other repositories).
  - Implements different data sources of an information system independently of the nature of these data sources.
  - Also known as data layer.
- RML may include as a part any external system that provides information:
  - Other information systems with all three layers.
  - This allows for a recursive building of an information system.

Design of an Information System

- The design of an information system can be a top-down or a bottom-up design.
- Typical workflow of top-down information system design:
  - Starts from the functionality driven by clients and their interactions with the system.
  - Application logic is developed to fulfill the required functionality.
  - Supporting resources are defined to support application logic operations.
- Focuses first on the high-level goals of the problem and does everything to achieve the goals.
  - Specifies also how the system will be distributed across different computing nodes.
  - Usually created to run on homogeneous nodes.
- Advantages:
  - Emphasizes the final system goals.
  - Tailored to address functional (system operations) and non-functional (e.g., performance and availability) system properties.

Drawbacks:

- Can be applied only to the systems developed from scratch (rarely a case today).

Design of an Information System

- Typical workflow of bottom-up information system design:
  - Starting with the high-level goals as in the previous case.
  - Evaluating RML to see whether something can be fulfilled:
    - What are the costs, feasibility to obtain the information.
  - RML components are wrapped to enable proper interfaces for app. logic layer.
  - Designing application logic.
- Occurs from necessity rather than choice:
  - Integration of existing systems (e.g., legacy apps).
  - Everything is predefined and cannot be easily modified.
  - How to integrate such a system in a coherent whole?
- Yields loosely-coupled systems by design:
  - Legacy systems are used as components and maintained as stand-alone systems.
- No sense to talk about advantages/disadvantages:
  - Usually there is no other choice.
  - Web services are making bottom-up design more efficient, cost-effective, and simple.

References

- Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G. (2012). Distributed Systems. Concepts and Design. Pearson, 5th edition.
- Tanenbaum, A. S. and van Steen, M. (2007). Distributed Systems. Principles and Paradigms. Pearson Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition.