



Taller de Desarrollo Web

Prof. Dr. Alfredo Teyseyre
Prof. Dr. Alejandro Zunino
ISISTAN – UNICEN

<http://www.exa.unicen.edu.ar/catedras/webtools/>

1

Distributed Systems

- Virtually all large computer-based systems are now distributed systems
- Information processing is distributed over several computers rather than confined to a single machine
- Distributed software engineering is now very important:
 - **extra dimension: component location**

6

Contents

- The World Wide Web
- HTTP/CGI, HTML Forms
- Data management technologies: JDBC, Hibernate, EJB, Key-Value (MongoDB), ...
- Application logic technologies: EJB, Spring, Express.js, ...
- Presentation technologies: JSP, JSF, AngularJS
- The MEAN Stack: AngularJS, Node.js, Express.js, MongoDB

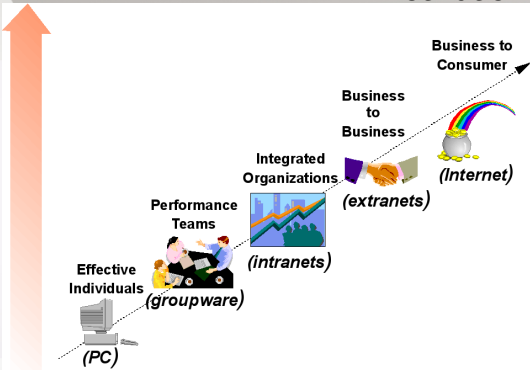
2

Distributed System Characteristics

- Advantages:
 - Resource sharing
 - Openness
 - Concurrency
 - Scalability
 - Fault tolerance
 - Transparency
- Disadvantages:
 - Complexity
 - Security
 - Manageability
 - Unpredictability

7

Motivation



3

Distributed System as an Enterprise System

- Accommodate changes gracefully:
 - scalability
 - dynamic reconfiguration
- Maintain high availability at all times
- Offer good performance in terms of response time and end-to-end "QOS"
- Fault tolerance
- Simplicity
- ...

8

Distributed Systems Architectures

- **Client-server architectures**

- Distributed services which are called on by clients. Servers that provide services are treated differently from clients that use services

- **Distributed object architectures**

- No distinction between clients and servers. Any object on the system may provide and use services from other objects

9

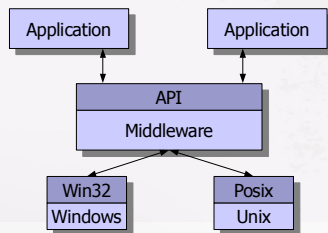
Middleware

- Middleware is usually off-the-shelf rather than specially written software
- Middleware typically includes a set of components such as resources and services:
 - Examples: Security, Directory and naming, transactions, support for mobile code.
- Examples:
 - OMG's CORBA defines a middleware standard.
 - J2EE Java enterprise edition is a middleware specification.
 - MEAN is a popular JavaScript stack

12

Middleware

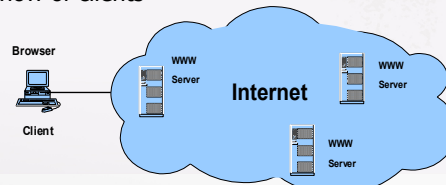
- Software that manages and supports the different components of a distributed system. In essence, it sits in the *middle* of the system



10

Client-Server Architectures

- The application is modeled as a set of **services** that are **provided by servers** and a set of **clients that use these services**
- Clients know of servers but servers need not know of clients



13

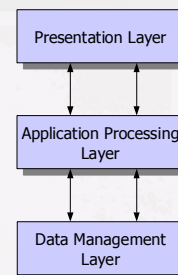
Middleware

- Layer between OS and distributed applications
 - Hides complexity and heterogeneity of distributed system
 - Bridges gap between low-level OS and programming language abstractions
 - Provides common programming abstraction and infrastructure for distributed application

11

Layered Application Architecture

- **Presentation layer**
 - Concerned with presenting the results of a computation to system users and with collecting user inputs
- **Application processing layer**
 - Concerned with providing application specific functionality ex., in a banking system, banking functions such as open account, close account, etc.
- **Data management layer**
 - Concerned with managing the system databases



14

Thin and Fat Clients

- **Thin-client model**

- In a thin-client model, all of the application processing and data management is carried out on the server. The client is simply responsible for running the presentation software.

- **Fat-client model**

- In this model, the server is only responsible for data management. The software on the client implements the application logic and the interactions with the system user.

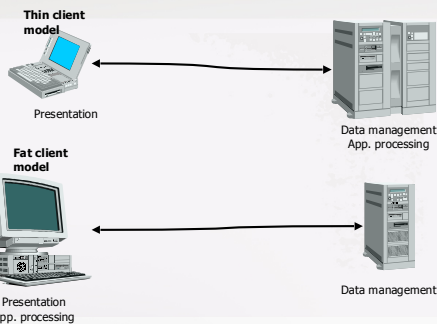
15

Fat Client Model

- More processing is delegated to the client as the application processing is locally executed
- Most suitable for new C/S systems where the capabilities of the client system are known in advance
- More complex than a thin client model especially for management. New versions of the application have to be installed on all clients

18

Thin and Fat Clients



16

Three-tier Architectures

- In a three-tier architecture, each of the application architecture layers may execute on a separate processor
- Allows for better performance than a thin-client approach and is simpler to manage than a fat-client approach
- A more scalable architecture - as demands increase, extra servers can be added

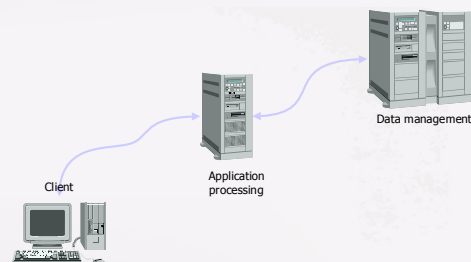
19

Thin Client Model

- Used when legacy systems are migrated to client server architectures.
 - The legacy system acts as a server in its own right with a graphical interface implemented on a client
- A major disadvantage is that it places a heavy processing load on both the server and the network

17

A Three-tier Architecture



20

Distributed Object Architectures (n-Tier)

- There is no distinction in a distributed object architectures between clients and servers
- Each distributable entity is an object that provides services to other objects and receives services from other objects
- Object communication is through a middleware system called an object request broker (software bus)
- However, more complex to design than C/S systems

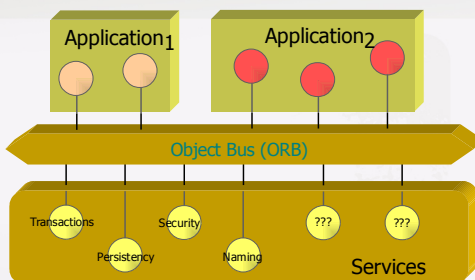
21

Uses of Distributed Object Architecture

- Allows you to structure and organize the system:
 - you think about how to provide application functionality solely in terms of services and combinations of services as a flexible approach to the implementation of client-server systems.
- The logical model of the system is a client-server model but both clients and servers are realized as distributed objects communicating through a software bus

24

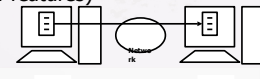
Distributed Object Architecture



22

Mobile Code

- Limitations and drawbacks with traditional approaches in large-scale distributed settings like the Internet
 - Scalability (growing size of network, network partitions)
 - Customizability (tailor functionality to specific needs)
 - Flexibility (dynamic nature of comm. infrastructure)
 - Extensibility (to add new features)



25

Advantages of Distributed Object Architecture

- It allows the system designer to delay decisions on where and how services should be provided
- It is a very open system architecture that allows new resources to be added to it as required
- The system is flexible and scalable
- It is possible to reconfigure the system dynamically with objects migrating across the network as required

23

Mobile Code

- Code mobility can be defined informally as the capability to dynamically change the bindings between code fragments and the location where they are executed
- Idea taken from process migration techniques
- Code mobility is exploited on Internet-scale
 - Programming is location aware
 - Mobility is under programmer's control
 - Mobility is not performed just for load balancing
 - service customization
 - extensibility
 - autonomy -> fault-tolerance
 - support for disconnected operations

28

The Future: The Semantic Web

- The focus is of providing autonomous programs (agents) to use Web accessible resources:
- The goal of the Semantic Web is to allow computers to "understand" not just the form but also the content of documents on the Web.
 - 1st generation, Internet enabled machines to exchange data
 - 2nd generation, enabled enormous amounts of information available, in human-readable form

27

Conclusions

- In a distributed object architecture, there is no distinction between clients and servers
- Distributed object systems require middleware to handle object communications
- Mobile code paradigms provide new possibilities by structuring systems in many small and autonomous mobile entities
- In the near future **programs will become users** of the Web

30

The Future: The Semantic Web

- The next generation of the net is an "agent-enabled" (Semantic Web) which makes information available in machine-readable form ... enabling "agent" communication at a Web-wide scale
- The Semantic Web is a vision: the idea of having data on the web defined and linked in a way that it can be used by machines

28

Assignment

- Download VirtualBox:
 - Ubuntu: install using the Software Center
 - Windows/Mac: <http://www.virtualbox.org>
- Download the Bitnami MEAN Stack
 - <https://bitnami.com/stack/mean/virtual-machine>
- Download a SSH/SFTP client & text editor (Windows):
 - <http://www.putty.org/>
 - <https://winscp.net>
 - <https://www.sublimetext.com/>

31

Conclusions

- Almost all new large systems are distributed systems
- Distributed systems support resource sharing, openness, concurrency, scalability, fault tolerance and transparency
- Client-server architectures involve services being delivered by servers to programs operating on clients
- User interface software always runs on the client and data management on the server

29