# Web Architectures

Pınar Yolum

Boğaziçi University, İstanbul, Turkey

Spring 2015

# Course Information

- Topics and schedule
- Course work
- Resources
- Grading
- Academic integrity

# Architecture Dimensions

- Distribution of nodes (Centralized vs. Distributed)
- Distribution of data
- Communication patterns
  - Roles of the nodes (Symmetric vs. Asymmetric)
  - Tasks of the nodes
  - Distribution of resources

# Monolithic Architecture

- Everything resides and runs in the same machine
- Application, data, ...
- No distribution so no communication

# Client Server Architecture (1)

- Two roles: Client and Server
- Asymmetric in resources, tasks, and interactions
- Server:
  - Has access to resources (database, printer, etc)
  - Waits for task requests from a Client
  - Performs the task and returns result to Client
  - Examples: Mail servers; Print servers

# Client Server Architecture (2)

- Client:
  - Represents a user with a task request
  - Talks to the server and sends requests
  - Waits for responses from the Server
  - Examples: Mail client

# Protocol

- Whenever there is communication, there is a *protocol*
- Set of rules that will be followed by the nodes
  - Which messages can be sent?
  - Who can send them?
  - When can they send them?
- Example protocol: Hypertext Transfer Protocol (HTTP)
- Example protocol: Read Me the First Sentence (RMTFS)

# Protocol Properties (1)

- Unambiguous: Should specify meanings of messages clearly.
  - What would happen if there were misunderstandings?
  - What would it take to be unambiguous?
- Complete: Should have messages for various occasions.
  - Illegal request?
  - Garbled data?

# Protocol Properties (2)

- Extendable: Should allow new messages to be added.
  - Versioning control
  - Need for standardization (e.g., W3C)
- Accessible: Should allow all participants to find out the details
  - Different applications from different vendors should be able to speak with each other
  - How can it be made accessible? Examples of inaccessible protocols?

# Protocol Types

- Synchronous
  - Client blocks after a request
  - Example: HTTP
- Asynchronous
  - Client continues after a request
  - Example: TELNET
- Example protocol: Read Me the First Sentence (RMTFS)

# Possible Server Functionalities

- Authentication
  - Establish who the client is
  - Generally through username and password
- Authorization
  - Check the credentials of the client
  - What is the client allowed to do?
  - Read mail; change files
- Recommendation
  - Suggest tasks to perform
  - Possibly based on previous tasks
- Concurrency
  - Enable other clients to access the system
  - Serve multiple clients at the same time
- What would these correspond to for Amazon.com?

# Two-Tier Client Server

- Typical C/S architecture is two-tier
- First tier: Presentation
  - User interface
  - Resides in the client
  - Allows invocation of business logic
- Second tier: Business logic
  - Application to carry out the tasks
  - Resides in the server
  - Data buried inside the application

# Three-Tier Client Server

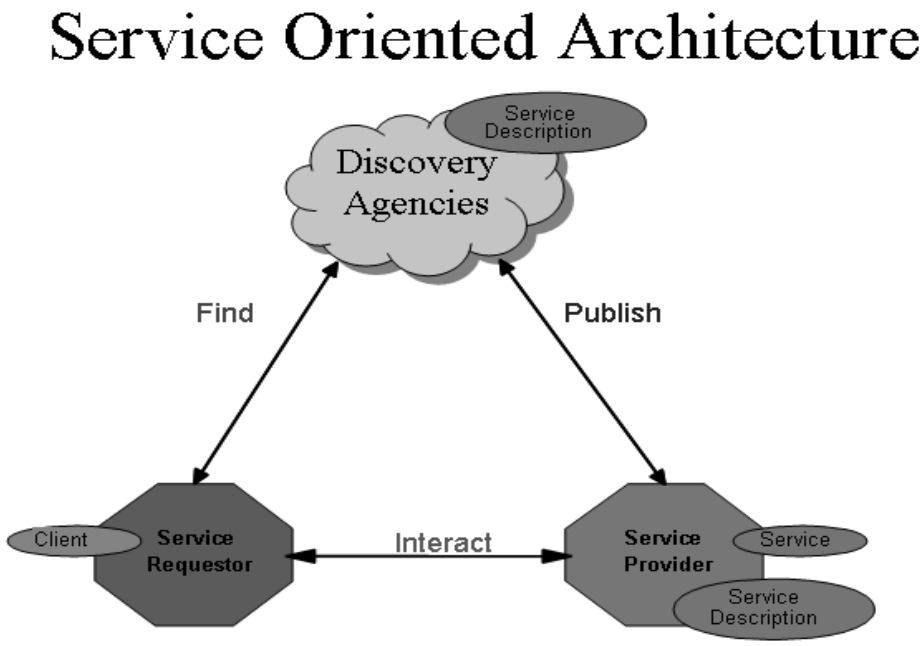
- Similar to two-tier C/S architecture
- Second tier divided into two
- Business logic tier
  - Application to carry out the tasks
  - Resides in the server
  - Does not contain data
- Data tier
  - Runs on a different data server (usually a DB server)
  - Provides access to data

# Peer-to-Peer Architecture

- Symmetric peers
- Each node can request and serve
- (Generally) multiple peers
- Peers enter and leave as necessary
- Communicate with a protocol
- Example: Music sharing
- Contrast with C/S
  - Bottleneck?
  - Pull vs. push

# Service-Oriented Architectures (SOA)

- Separate service implementation from the interface
  - No need to know the internal implementation
  - Follow a previously agreed protocol
- Find-Bind-Execute Paradigm



# SOA Entities

- Service Consumer
  - Locates the Producer in the Registry
  - Initiates the communication
  - Follows a Contract
- Service Producer
  - Delivers services
  - Advertises its services in a Registry
- Service Registry
  - Stores advertisements
  - Allows lookup service to Consumers

# Languages

- Depends on the tier
- Presentation: HyperText Markup Language (4, 5 ...) with possible scripting languages (e.g., Javascript, PHP)
- Business logic: Compatible with the environment it will run (e.g., Java)
- Data tier: To retrieve and update data (e.g., SQL) with appropriate layer (e.g., JDBC)

# Communication Paradigm

- Invocation
  - Allow one node to invoke methods of another node
  - Assumes enough knowledge about the working of the other node
  - Gives access to other's resources
  - Limits autonomy of the other node
  - Example: RMI
- Message passing
  - Allow nodes to send messages to each other
  - Assumes existence of a common protocol
  - Receiving node deals with the message
  - More autonomy for the other node
  - Example: Java messaging

# JAVA Platforms

- Simple Edition (SE): Standard packages
- Enterprise Edition (EE): Web services, multi-tier applications, XML support
- Micro Edition (ME): Mobile phones, wireless clients, small interfaces, etc.
- Orcale's Embedded (Java TV, Java Card, etc)

# JAVA Development Environments

- High-end, project-based, SVN-support
  - Eclipse
  - NetBeans
  - IntelliJ
- Platforms
  - Spring.io
  - Java EE