

CmpE 494: Service-Oriented Architectures and Web Services

Pınar Yolum

`pinar.yolum@boun.edu.tr`

Department of Computer Engineering
Boğaziçi University

Part I: Architectures

Internet Architectures

- A set of nodes collaborate to carry out a job
 - A node wants to print, but doesn't have access to a printer
 - A node needs data that is available at a different node
- A common language for communication
 - Usually not a full-fledged language
 - Protocol that specifies what to do in specific situations

Protocols

- Set of rules that will be followed by the participants
 - Events that take place
 - The initiators
 - The timing of events
 - The data formats
- Does not specify how the protocol should be implemented
- Example*: Hypertext Transfer Protocol (HTTP)

Protocol Properties (1)

- Unambiguous
 - The protocol state should state clearly what should be done in a situation
 - No misunderstandings
- Complete:
 - The protocol should cover all possible requests
 - Garbled data?
 - Illegal request?

Protocol Properties (2)

- Extendable:
 - The protocol should allow new requests and responses to be added
 - Versioning of protocols
 - World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) work to standardise versions of protocols
- Accessible
 - Different clients and different servers may be designed and implemented by different programmers
 - Should still be able to speak the same language

Protocol Types

- ● Client sends a request and blocks
 - Server responds
 - Example: HTTP, SMTP
- Asynchronous
 - Clients and server send information at the same time
 - Example*: TELNET
- Deferred Synchronous
 - Continue operation until a certain point and then wait
 - Example: CORBA

Monolithic Architecture

- All parts run on the same machine
- Direct access to data
- Difficult to change or maintain

Client/Server Architecture (1)

- Client and server asymmetric in capabilities
- Client*
 - Represents a user
 - Program that request tasks from servers
- Often users interact with a client through a GUI
- Client translate the user requests to protocol tokens
- Clients initiate the interaction with a server

Client/Server Architecture (2)

- Server: Program that waits for incoming communication requests from a client
- Usually has some resources that the client does not have
 - Bandwidth, access to printer
- Takes the request
 - Process data
 - Perform a task
 - Return results client
- Examples: Mail servers, Print servers, Web servers

Client/Server Architecture (3)

- Common protocols
 - Mail servers: SMTP
 - Print servers: LPP, IPP
 - Web servers: HTTP
- On request, the server performs a service and sometimes returns a success/fail response
- The client *pulls* the information

Server Tasks

- Application-dependent
- Authentication: Verify the identity of the client
 - Check if the Username/Password supplied by the client matches the one already stored
- Authorization: Check that the client has rights to perform what it wants to perform
 - Check mail of another user?
 - Delete a file?
- Concurrency: Allow multiple clients to use the server at the same time

Two-Tier Architecture

- Example*: registration.boun.edu.tr
 - What does the client have? Database, logic?
 - Who uses the client?
 - What does the server have? Database, logic?
 - Two tiers?

Three-Tier Architecture

- Three separate layers
- Presentation tier
 - Runs on client
 - Provides user interface
 - Invokes business logic
- Business logic tier
 - Runs on server
 - Has application logic, business rules, etc.
- Data tier
 - Runs on a database server
 - Stores and provides access to data
- Advantages?
- N-Tier?

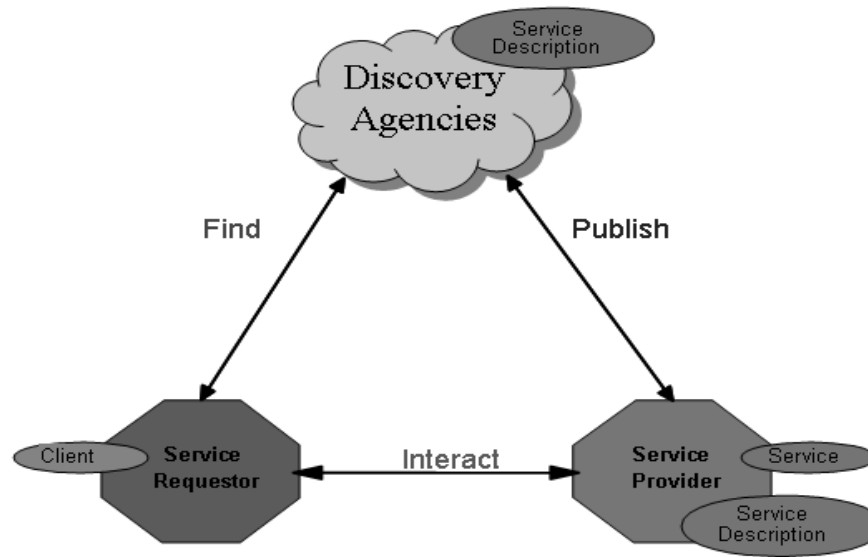
Invocation vs. Message-Oriented

- Invocation
 - Assumes knowledge of the other party
 - Gives access to others's resources
- Message-Oriented
 - Recipient deals with the message
 - Recipient can change its functioning
 - Increased abstraction

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

Service Oriented Architecture



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

SOA Concepts

- Service Contract
 - Specifies how the Consumer and the Producer will interact
 - Specifies the necessary conditions for the Producer to execute a task
 - Defines the QoS requirements

SOA Properties (1)

- Entities are autonomous
 - Resources owned and managed by individuals
 - Choose how they will carry out their tasks
 - Choose whom they will carry out business with
- Entities are dynamic
 - Entities can change their behavior
 - Entities may not always be available
- Entities are interoperable
 - Entities can communicate even if they are written in different languages or run on different platforms
 - Standard protocols or data formats should be available

SOA Properties (2)

- Services are loosely coupled
 - Established by contracts and dynamic binding
 - No dependency on the service implementation
- Services are composable
 - Put together to offer a composite service
 - Dependencies between the services should be handled

Comparison of Architectures

	Monolithic	C/S	N-tier	Service-Oriented
Data formats	Proprietary	Proprietary	Open	Standard
Protocols	None	Proprietary	Open	Standard
Scalability	Low	Low	Medium	High
Number of nodes	Small	Small	Medium	High
Coupling	None	Tight	Tight	Loose

Part II Web Services Overview

Component-Based Development

- Groups of objects
- Provides application functionality
 - Rather than access to individual data items
- Components communicate to yield enhanced functionality
- Components are composed and compiled at design time

Service-Based Development

- Allows late binding
 - Consumer looks up a service in a registry
 - Possibly chooses among several possibilities
 - Binds to the one it selects
 - Enacts the service
- Web-based standards
- Standardized data formats

Web Services Stack (1)

- Service Transport
 - Transfer data between different nodes
 - HTTP used most often
 - Not affected by firewalls
 - Connectionless and stateless: Independent requests and responses
- Service Messaging
 - Extensible Markup Language (XML)
 - Self-describing messages
 - Data structured as a tree
 - Simple Object Access Protocol (SOAP)
 - Defines how data is packaged in an XML message
 - Contains an envelope, a header, and a body

Web Services Stack (2)

- Service Description
 - Functionalities that the service provides
 - Set of acceptable messages
 - Protocol with which consumers can bind and communicate with the service
- Service Registry
 - Universal Description, Discovery and Integration (UDDI) Registry
 - Itself a Web service
 - Allow service providers to publish information
 - Allow service consumers to find Web services for given service characteristics
 - Communication through SOAP messages

Web Services Stack (3)

- Service Composition
 - Each Web service is thought of carrying out small task
 - Combine tasks from different Web services into one large transaction
 - Example:
 - Web service A can be used for booking a flight.
 - Web service B can be used for reserving a hotel room
 - Compose them into one service to arrange the entire trip.
- Business Process Execution Language for Web Services (BPEL4WS)

Quality of Service

- Availability

- When can it be used? Now? At certain intervals?
- Metrics for measuring availability.
- Example: Time-to-repair (TTR)

- Accessibility

- Extent of finishing the requested service
- Metrics for measuring accessibility
- Example: Success rate

- Can a Web service be available but not accessible?

Quality of Service (2)

● Performance

- Throughput: How many service requests can be handled by the Web service in a unit time?
- Latency: How long does it take to get a response to a request?
- Maximize throughput, minimize latency

● Reliability

- Can it guarantee the same performance over a period of time?
- How many failures take place in a period of time?

Quality of Service (3)

- Integrity

- How correctly is the source executed?
- All tasks need to be performed in the correct order
- Otherwise, roll back

- Security

- Provide authorization and authentication for accessing resources
- Preserve confidentiality of private consumer information

Example Applications

- Travel planning
 - Simple, individual services
 - Can be composed in various ways
 - Some service providers may be preferred over other