**WSs**
- Web Services form a support base for Distributed Information Systems (DISs)

**Study**
- Problems these try to solve
  - similar to DISs
- Design Constrains ← new

**Outline**
- [A] Limitations of Present Technologies
- [B] Web Services Techniques -
  - Concepts -
    - how do WSs tackle the Application Integration
- [C] Functionality - needed for distributed applications
- [D] WS Architecture -
  - Connection: Synchronous ← Asynchronous
  - Access: low level ← high level
  - (not RPC but SOAP)
- **Centralized Systems** → DBMS (one system)

- Network → Communication

- **Distributed System** → Dist. O.S. (commands)

- **Distributed Database System** → 1 database image
  - Directory
  - Recovery, Logging
  - Concurrency Control
  - Load Balancing
  - Transactions

- **Distributed Heterogenous Database System** →
  - Two or more DBMSs working under a common software system

- **Federated Database System** → (use JDBC)
[I] Web Services: Definitions

• Definition( 1 ) [72,133] -
  "An application accessible to other applications over the web".
  + any thing that has a URL (? )

• Definition( 2 ) [ UDDI Consortium ] [203] -
  " Self-Contained, modular business applications that have open Internet-Oriented, Standard-based interfaces".
  + Internet-Oriented ← standard;
  + open ← published interface, can be invoked across the web
  - (?) modular business applications (?)

• Definition( 3 ) [ WWWC (W3C) Consortium ] [212] -
  1. " A software application identified by a URI"
  + interfaces and bindings: Defined, Described and discovered using XML
  2. WS uses XML-based messages exchanged via internet based protocols.
Web Services

- **Large Scale B2B activity**: Car maker ↔ Parts supplier
- **Large Scale B2C activity**: Bank/Books/PCs ↔ Customer
- **Large Scale Application**: System Upgrades ↔ more Application integrations (Bottom-up design)

1. All levels (except Web) – directory
2. Homogeneous components – compatibility? (software/hardware can connect with ease)

- Web Services ↔ external world connectivity
Web Services: -

- WSs ("services") similar to middleware
  - up and running

- Describe and advertise

- possible to write client software
  - that can bind and interact with them.

- WSs are components
  - Can be integrated into complex distributed applications

- Definition (4): [Webopedia]: On-line technical dictionary -
  "A standardized way of integrating web-based applications - using:
  XML (tag the data),
  SOAP (transfer data),
  WSDL (description of services),
  UDDI (list of services) open standard, over the internet"
WSs : usage

- Service Oriented Paradigm in Application Development
- Integration of serveral systems
  - autonomous,
  - heterogeneous

Example: Toyota Car Company and its part suppliers

- Business Process Span - need for automation
  - [At Present]: Employees access internal systems
  - Fill up web forms or send Emails across to order goods
B2B interactions occur by accessing Web pages, filling Web forms, or via email.
Conventional Middleware

- Cross-organizational DISs -

(?) Where to put the middleware

- Earlier, middleware was opted (chosen) by one company
- Many Companies must agree on co-operation

Good idea: Few companies with close cooperation
General Case: + lack of trust
+ autonomy
+ hiding internal business

- Centralized middleware hosted by one company
  → Not a possible solution
Point-to-Point

- Customer - Supplier pair
  
  + use common middleware
  + (advantage) Only intended customer can see the business data

- Problems: -

- A company interacts with many different partners
  
  + each may have a different platform

- Each company must support and maintain many middleware systems
Point-to-Point Interaction

- Interactions are long
  - earlier assumption - Transactions are short

- Interactions do not call a function
  - may take a few hours or weeks
  - After the courier picks up the goods, may send despatch message

- Delays: Asynchronous exchanges

- Other Problems:
  - Trust domain is not same
  - authentication
  - encryption

- restriction - what other client can do
  - Companies severly restrict - what other clients can do
  - (?) What resources can be locked; (?) control over locking
<table>
<thead>
<tr>
<th>Web Activities</th>
<th>Web Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Every service available on the web ample</td>
<td>• Service in Software Sense ample</td>
</tr>
<tr>
<td>- Book store</td>
<td>- middleware</td>
</tr>
</tbody>
</table>

Outline:

- [A] Web Services: a software application
- [B] - published and stable program interface
- [C] - invocation made by a program
- [D] - usage same as middleware
- [E] - WSs are loosely coupled
- [F] Design: Services are autonomous
- [G] [New Problems]: redesign middleware for peer-peer function
[I] Middleware Protocols

- **Redesign**: Peer-to-peer and across companies

- **Example** P2P: Transaction Commit (2PC)
  Design assumptions - do not hold in cross-organization interactions

  + Central Coordinator: Can not work in Cross-organization environments

  + Central Coordinator can lock resources at other systems

  → [X] Coordinator - There is lack of trust and confidence

[A1] → 2PC to be redesigned: work in a fully distributed fashion

[A2] → must be extended: allow flexibility in terms of locking resources

→ + All interactions and coordination protocols ← Redesign for WSs
Middleware Protocols

- Many properties provided by conventional middleware
  + reliability / + guaranteed delivery \(\rightarrow\) Centralized platforms: redesign

- Example: Deadlock Handling:
  \(\rightarrow\) (centralized coordinator)

  + make a Transaction Wait For Graph (TWFG); or

  + send a probe to find the deadlocked list of transactions
languages and protocols standardized, eliminating need for many different middleware infrastructures (need only the Web services middleware)
Web Services middleware

- Web Services need to be based on standards
- Interaction between companies (B2B integration problems) -
  - P2P
  - Through standard protocols
  - There is no central middleware platform

- WSs: Each party exposes its internal operations as WSs,
  → entry point to local information system
  → WS interface: functionality performed by internal system

- Web Services (WSs) use middleware for execution of P2P protocols

- Expose (interface is discoverable) -
  1. WSs are sophisticated wrappers;
  2. Encapsulate one or more applications;
  3. provide a unique web access interface.
Company A (provider)  Company B (client)

middleware

internal service

Web service

Web service

Web service

wide area network (Internet)

client

middleware

internal service

internal service

Copyright Springer Verlag Berlin Heidelberg 2004
WSs Parts

- WSs wrappers: Hide heterogeneity

- clients -[see]- wrappers (standards) homogeneous
  + reduce difficulties of integration (standard components)

- Future Software applications
  + out of box with WS interfaces

- Even LAN based systems: WSs based?
Company A (or a LAN within Company A)

integrating application (contains the composition logic)

Web service-enabled broker

SmartQuotation  DBMS applications  SmartForecasting  sendmail application  XYZ

assumes all back-end systems are accessible as Web services
<table>
<thead>
<tr>
<th>What</th>
<th>● What is Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>How</td>
<td>● How service - can be described ?</td>
</tr>
<tr>
<td></td>
<td>● [A] Service Descriptions in Conventional Middleware</td>
</tr>
<tr>
<td></td>
<td>● [→] are known to programmers developing the clients</td>
</tr>
<tr>
<td></td>
<td>● [i] Sematic of different operations</td>
</tr>
<tr>
<td></td>
<td>● [ii] Order in which these will be invoked</td>
</tr>
<tr>
<td></td>
<td>● [iii] Other properties</td>
</tr>
<tr>
<td></td>
<td>● [→] Based on a) interfaces b) IDLs</td>
</tr>
<tr>
<td></td>
<td>● [+] IDLs automatically generate stubs and contribute basis for dynamic binding and linking</td>
</tr>
</tbody>
</table>
vertical standards

properties and semantics

business protocols

interfaces

common base language
Components

1. Common Base Language
   - common meta language for all aspects
   + XML
   + has a flexible syntax

2. Interfaces (IDLs in case of middleware)
   + WSDL: need to specify URI of service
   + WSDL: How to invoke (Which transport protocol to use (HTTP?))

3. Business Protocols: WSs offer many operations
   → clients invoke operations
   Example: Procurement - Customer + request a quote; + order; + pay.
   [ Operations for a purpose ] → conversation
   [ Business Protocols ]: → rules for conversations
   + WSCL - WSs Conversation Language (being studied)
   + BPEL: Business Process Execution Language for WSs
Components

- [4] Properties and Semantics:
  + Functional Interface
  + Non-functional interface (new)
  - Cost of service; - text (goods return policy);
  → UDDI used: How to organize and information
- [5] Verticals
  + Vertical Standards define specific interfaces, protocols, ...
  → that WSs in each domain should support

Model: WSs → Vertical Standards ← Client
Service Discovery

• Clients: [interactions] [P2P fashion] : Directory Service
• APIs and protocols → UDDI (publish and discover)
  → Concerned with static and dynamic binding
  + middleware properties (horizontal protocols)
  + protocols infrastructure (meta-protocols)
  + messaging (basic and secure)
  + Transport
<table>
<thead>
<tr>
<th><strong>middleware properties (horizontal protocols)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>protocol infrastructure (meta-protocols)</strong></td>
</tr>
<tr>
<td><strong>basic and secure messaging</strong></td>
</tr>
<tr>
<td><strong>transport</strong></td>
</tr>
</tbody>
</table>
[I] UDDI based Service Interactions

• 1. Transport:

WSs: Communication Network is hidden behind a transport protocol
+ use many (HTTP - most common)

• 2. Messaging:

WSs: standard way to format and package
+ use SOAP (Simple Object Access Protocol)
  → specifies a message template to add on the top
  → specifies transactional property and Security property (encryption)

• **WS-Security**: How to implement secure exchanges with SOAP
  + Additional Specifications and standards
3. Protocol Infrastructure (meta-protocols)

Before an interaction: client and services

+ need to agree: protocols? co-ordination? messages?

**WS-Coordination**: Specification

→ Standardize the meta-protocols

→ How to use the WSDL and SOAP for a task

4. Middleware (horizontal) protocols

P2P protocols

**Example**: reliability + transaction properties

→ execute 2PC, Deadlock handling, …

**WS-Transaction**: Specification

→ Based on WS-Coordination;

→ Defines - how to build a transaction Perspective
[I] WSs Architecture

1. WSs Technologies (internal architecture)
   → internal middleware (similar to conventional middleware)
   Request → Basic system → response

2. WSs Technologies (external architecture)
   + standardization efforts
   + Purpose: Integrate different WS components
     [A] Centralized Brokers
     → Route Messages + Provide properties for interactions:
        logging; transactional guarantee; Name and directory; reliability
     [B] Protocol Infrastructure: Implement P2P protocols
     [C] Service Composition Infrastructure Tools:
        + definition and execution of composite services
Internal Architecture

- WSs: (a) Simple; (b) Composite
- WSs invoke internal services
  → implement the needed application logic
  → Collect results
- Cause overheads
- package and unpack messages
Company A (service provider)

Web service interface
access to internal systems

Web services middleware (internal)

service interface
integration logic

conventional middleware (includes middleware services)

other tiers

clients from other companies

Conventional middleware provides lots of services (load balancing, transaction support, etc). Current Web services middleware is quite poor in this respect.
[I] External Architecture of WSs

- 1. Centralized Brokers

- 2. Protocol Infrastructure

- 3. Service Composition Infrastructure

- Example:
  - Implementation of Name and Directory Service

(?) Where should this middleware reside

- [A] Peer-to-peer: participants cooperate to provide service

- [B] Intermediatories and brokers act as the necessary middleware
### [I] External Architecture of WSs

<table>
<thead>
<tr>
<th>Main Issue</th>
<th>Main Assumption</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>• (?) provide the degree of reliability and trust required for industrial strength systems</td>
<td>• Can find a trusted and reliable site → Name and Directory service</td>
<td>• [A] Such servers ← Part of WSs middleware Infrastructure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• [B] → → Participants and a part of middleware : may reside at different sites.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• [C] Standardized WSs broker: only one (so far) → Name and Directory service</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• [D] Most of reference papers on Architecture → Name and Directory service</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• [E] WSs discovery → part of External middleware</td>
</tr>
</tbody>
</table>
WSs External Architecture

- Service Providers:
  - create WSs
  - define interfaces for invoking them
  - generate service descriptions
  - publish service in service registry

- Service Registry: catalogues and searches when service request arrives
  - Answer to - Where to locate and how to invoke

- Service Requester: Can bind to service provider by invoking the service (directory/Service registry)

- Address and Interface details are known apriori by the requester
[I] Distributed Information Systems

Company A (service requester)

Web service client

Web services middleware (internal)

other tiers

Company B (service provider)

Web service

Web services middleware (internal)

other tiers

Company C (directory service provider)

Service descriptions

1. publish the service description
2. find
3. interact

the abstraction and infrastructure provided by the registry are part of the external middleware

Copyright Springer Verlag Berlin Heidelberg 2004
External Architecture

- Figure 5.13 [shows] service discovery
  - as the only component of the WSs middleware

- (?) TM features of the TP monitor

- (?) Services offered by CORBA (Object Broker)

Research Problems

- 1. How to build centralized brokers at well known locations
- 2. Provide high degree of efficiency
- 3. A broker to mediate business interactions
  - may not be acceptable to companies

→ → Two approaches - [A.] Centralized  [B.] P 2 P
<table>
<thead>
<tr>
<th>Concept</th>
<th>Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Imagine centralized broker for WSs</td>
<td>• Implementation - similar to brokers in conventional middleware</td>
</tr>
</tbody>
</table>

**Outline**

- [A] Technically feasible
- [B] Difficult in practice
- [→] requires a standard way of running transactions (accepted by everyone) - so that transaction semantics are not violated
- [C] Transactional Semantics → depend on middleware platform
- [D] Standardize transactional semantics across middleware tools
<table>
<thead>
<tr>
<th>Idea</th>
<th>• under preparation - <strong>WS-Transaction</strong> Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>• Middleware platforms will follow a common transactional interface</td>
</tr>
<tr>
<td>outline</td>
<td>• [A] Assumption - All participants trust the broker</td>
</tr>
<tr>
<td></td>
<td>• [B] Assumption ? (O.K.) in restricted settings</td>
</tr>
<tr>
<td></td>
<td>• [C] Few Examples: Trusted brokers in on-lone shopping</td>
</tr>
<tr>
<td></td>
<td>• [Examples] - Yahoo ! Lycos !</td>
</tr>
<tr>
<td></td>
<td>• [E] Situation not same as directory service</td>
</tr>
<tr>
<td></td>
<td>• [F] More research going on</td>
</tr>
</tbody>
</table>
[B.] Consider a P2P model TM

- Each service requester its own TM
  → on transaction execution - its own TM executes
- Requires → standardized transactional interactions
  + same as centralized solution
- Functionality provided by a P2P solution (?) may be a subset of earlier systems

- Present Trends: P2P model
Service Composition Tools

- Can be centralized

- Implementations are often proprietary and confidential

- Infrastructure → will be provided by the service provider
  + [ P2P model ] not by a Third party
Distributed Information Systems

Company A
(service requester)
- Web service client
- internal middleware
- other tiers

external middleware
- transaction mgmt
- other protocol infrastructure
- composition engine

Company B
(service provider)
- Web service
- internal middleware
- other tiers

Company C
(directory service provider)
- service descriptions
I] Summary

• 1. WSs are being used as - Sophisticated wrappers over conventional middleware platforms

• 2. WSs comprise additional Tier

• 3. Allow middleware services to be invoked as WSs

• WSs are defined by -
  • [A.] Internal Architecture
    + Connection and local information system
      - supported by internal middleware
  • [B.] External Architecture
    + How to discover and interact
      - supported by external middleware
      - relies on standards
      - Configure WSs landscape
      - perform cross-organization interactions across internet