Web Services

Serge Abiteboul

INRIA-Futurs

Abstract

Abstract: web services

Web Services are the next step in the evolution of the World Wide Web and allow active objects to be placed on Web sites providing distributed services to potential clients.

Noise comes from e-commerce.However, one of their main current uses is for the management of distributed information. Distributed database systems always suffered from platform and software incompatibilities. Web services are not inventing anything new, but they are bringing an important breakthrough to distributed data management simply because they propose web solutions that can be easily deployed independently of the nature of the machine, the operating system and the application languages.

Ubiquitous XML distributed computing infrastructure

We first discuss **SOAP**, the Simple Object Access Protocol. SOAP is an XML based lightweight protocol for exchange of information in a distributed environment. In particular, it allows to specify the (XML) types of arguments and service results. SOAP can be used, in particular, in combination with HTTP.

We then turn to **WSDL**, the Web Service Definition Language, a language for describing web service interfaces, something like Corba's IDL for the web. WSDL is an XML format for describing network services based on operations and messages. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define a functionality of a service.

Abstract (3)

Next we consider **UDDI**, Universal Discovery Description and integration. UDDI is a specification for distributed Web-based information registries of Web Services. UDDI is also a publicly accessible set of implementations.

To illustrate how this may be put to work, we consider **Active XML**, a research project at INRIA. The underlying model is based on XML documents possibly embedding calls to web services. Web services operations can also be defined by means of **XQuery** on Active XML documents. Being Active XML data themselves, the arguments and results of service calls may also contain service calls allowing for distributed query processing over the web.

Organization

- *Abstract*
- Introduction
- Prerequisite: XML
- Web services: SOAP protocol
- Publishing web services: WSDL
- Discovering web services: UDDI
- Research glance: Active XML
- Conclusion

Something like Corba (in simpler)

Something like IDL

Something like Yellow Pages

Everything and more in much better ☺

Jargon

Yes it is boring!

WSFL

XHTML

.NET

XML

DTD

RDF

RosettaNet

XSL-FO

Xschema

namespace

XSL

XSLT

ebXML

HTTPS

SOAP

HTTP

ICE

OASIS

OAGIS

MIME

WSDL

RSS

UDDI

WSDL

Introduction

The web today

- Protocol: HTTP
- Documents: HTML
- Millions of independent web sites and billions of documents
- Browsing and full-text indexing
- Publication of databases using forms

Step 1: Go XML!

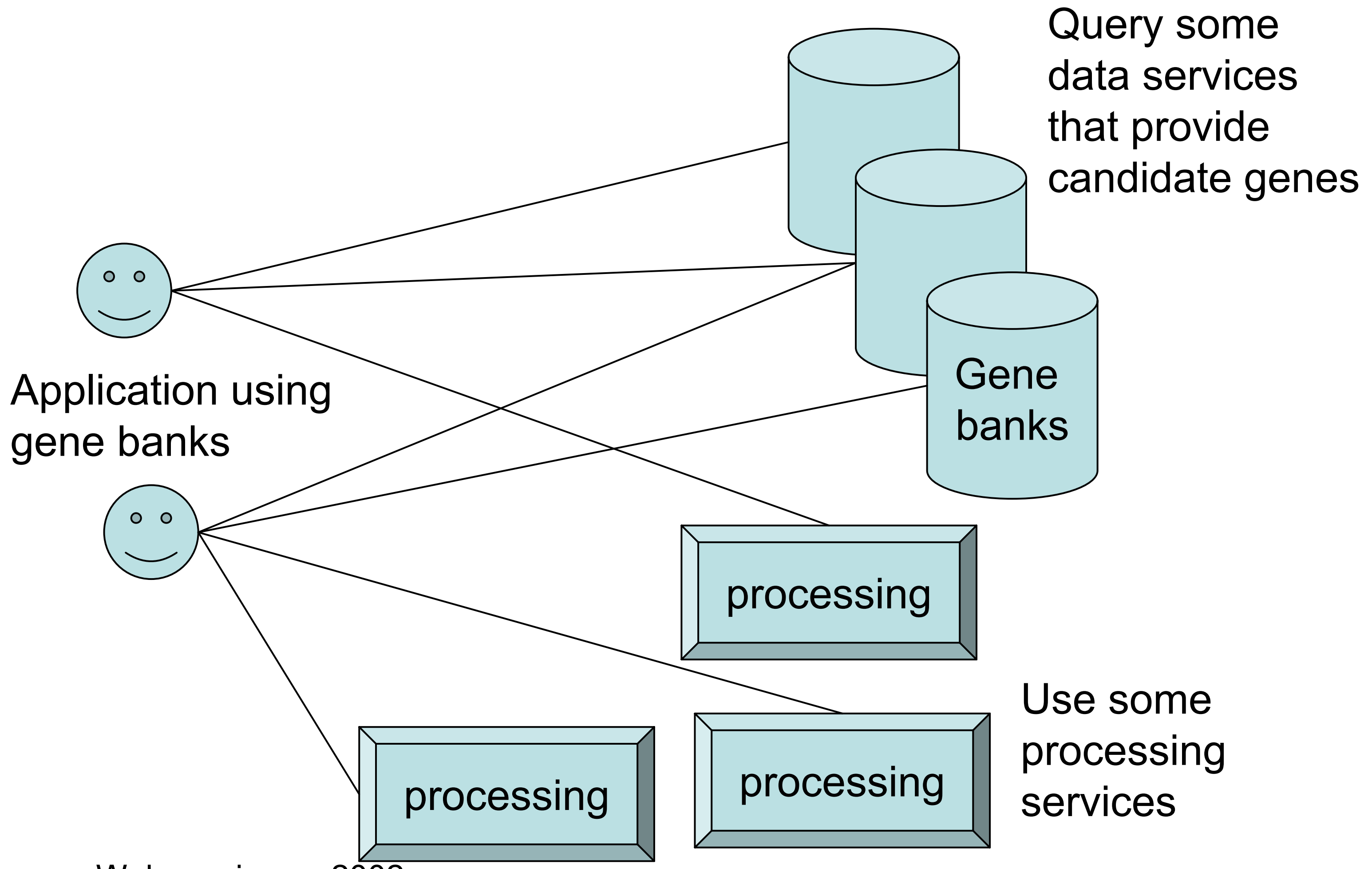
- Prerequisite
- In short: ***labeled ordered trees***
- In short: move from a document world to a data world

- Is it the ultimate data model? No
- Purely syntax – more semantics needed
- Is it OK for now? Definitely yes (because it is a standard)

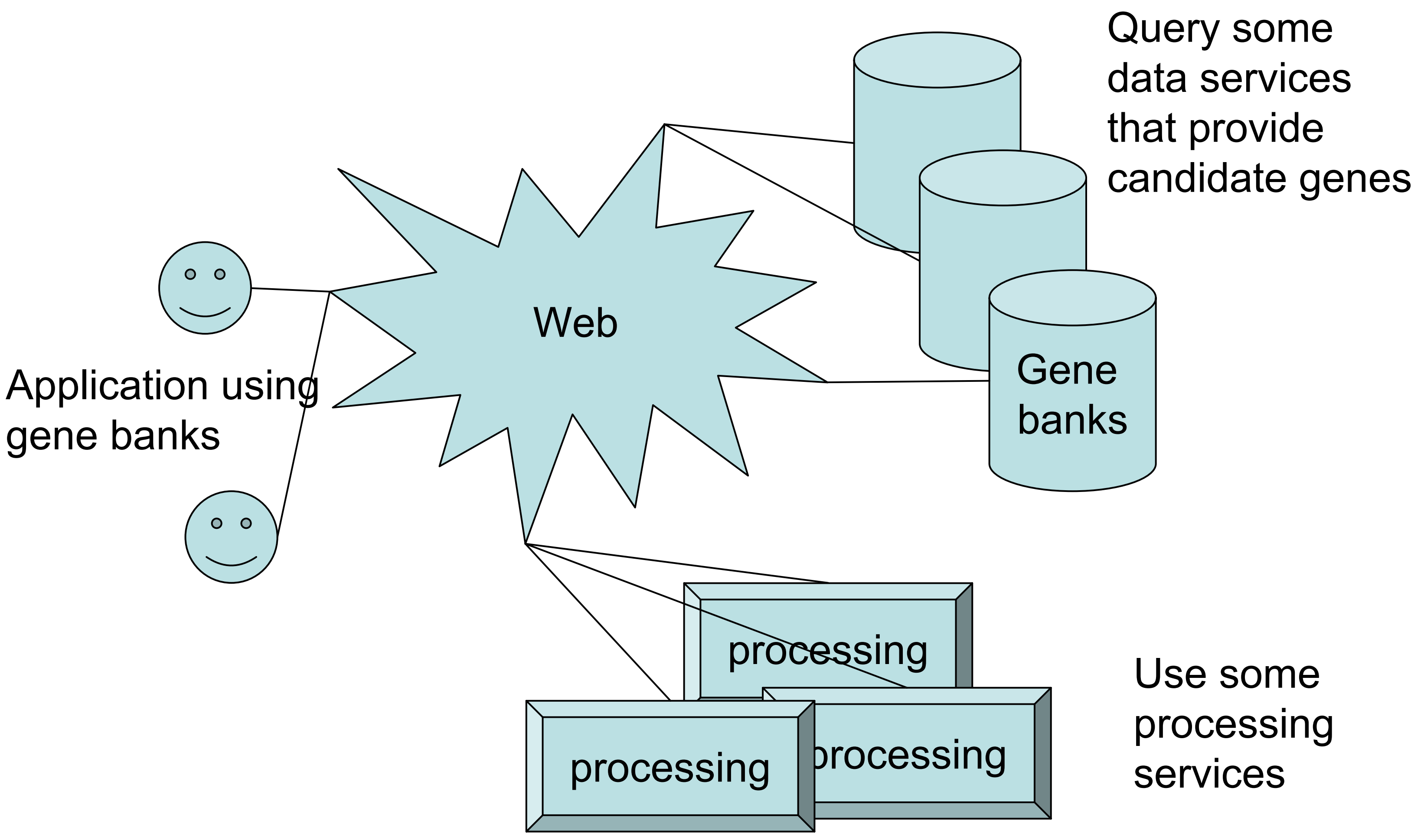
Step 2: web services

- Possibility to activate a method on some remote web server
- ***Ubiquitous XML distributed computing infrastructure***
- 2 main applications
  - E-commerce
  - ***Access to remote databases***

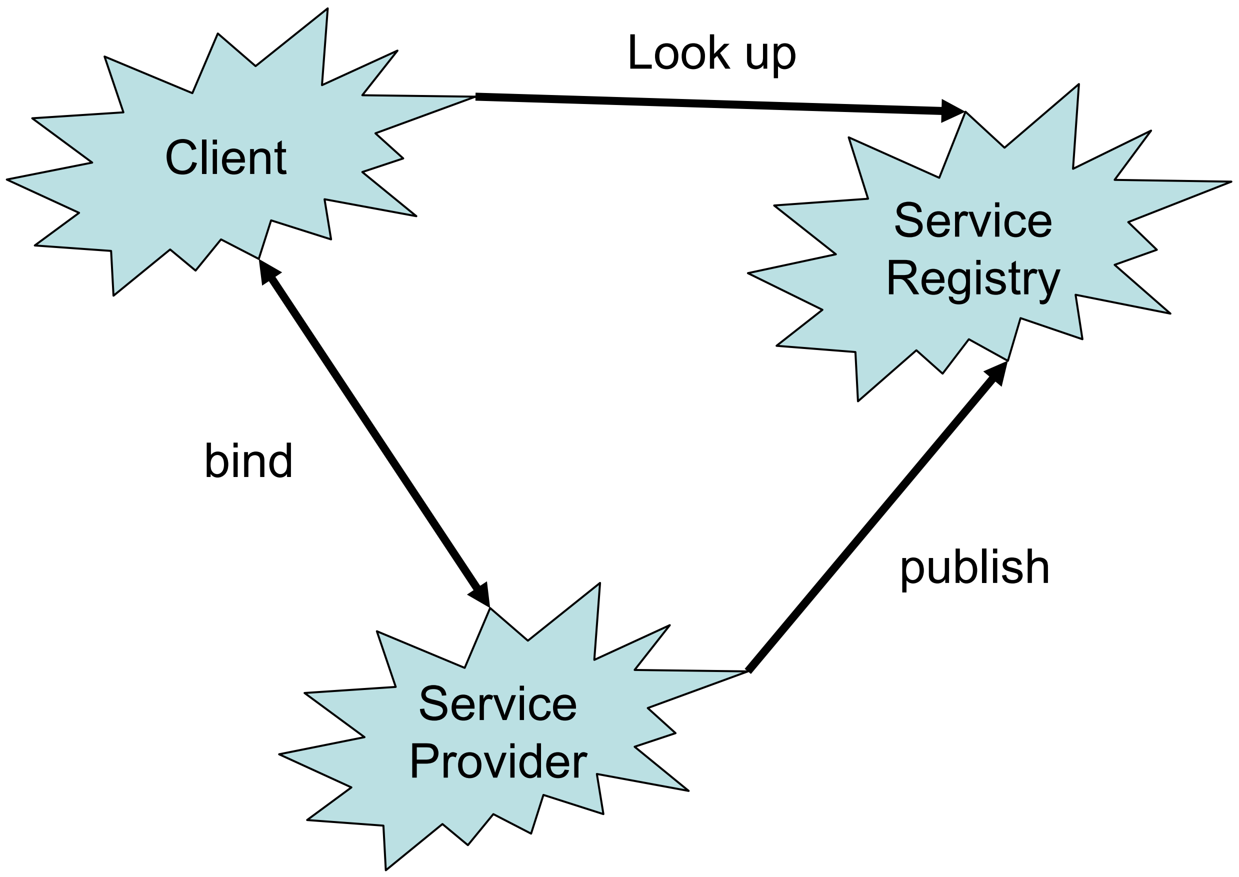
Accessing remote information


Query some data services that provide candidate genes
Gene banks
Application using gene banks
processing
processing
processing
Use some processing services

Same with web services


Query some data services that provide candidate genes
Web
Gene banks
Application using gene banks
processing
processing
processing
Use some processing services

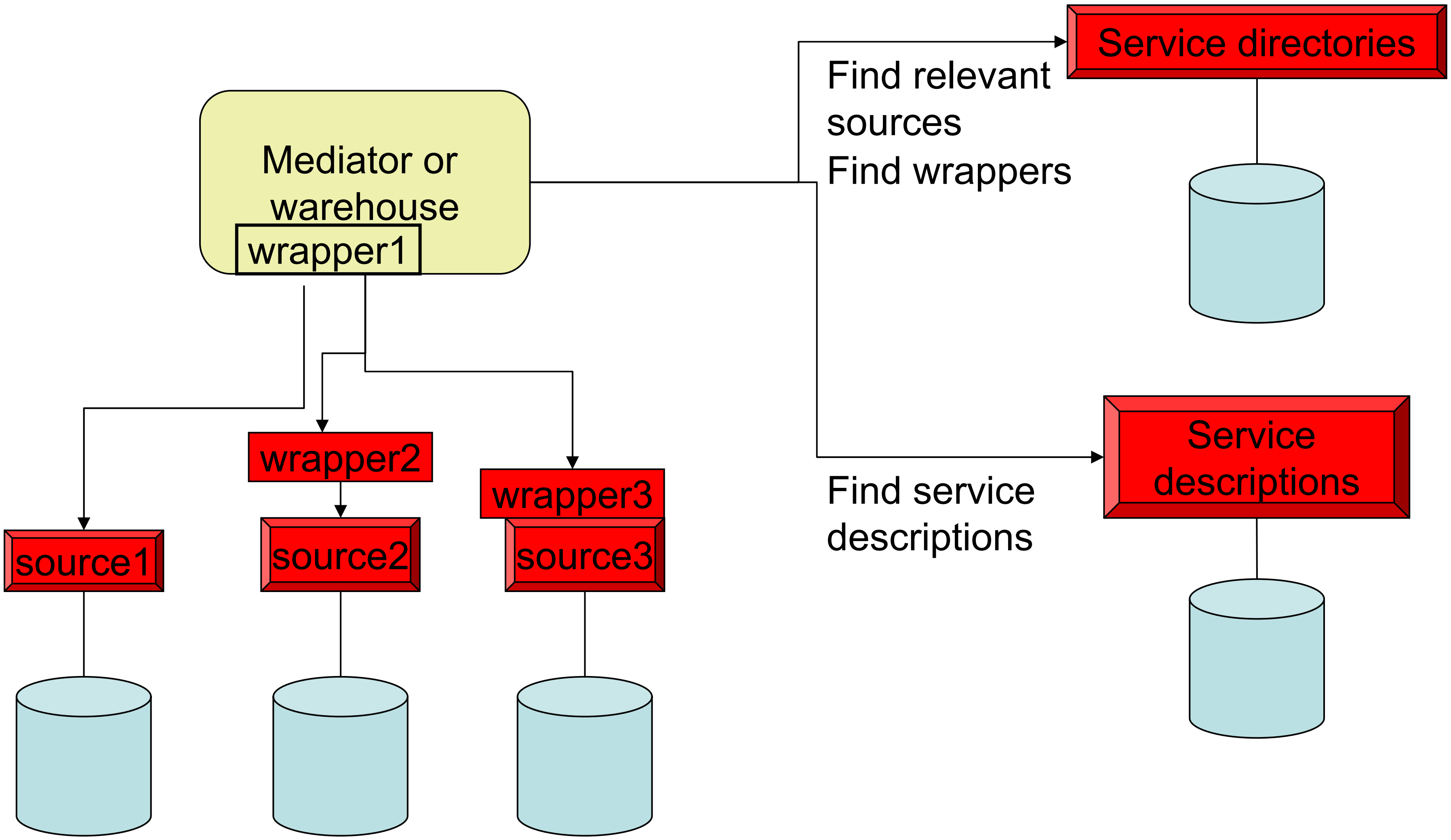
The main roles



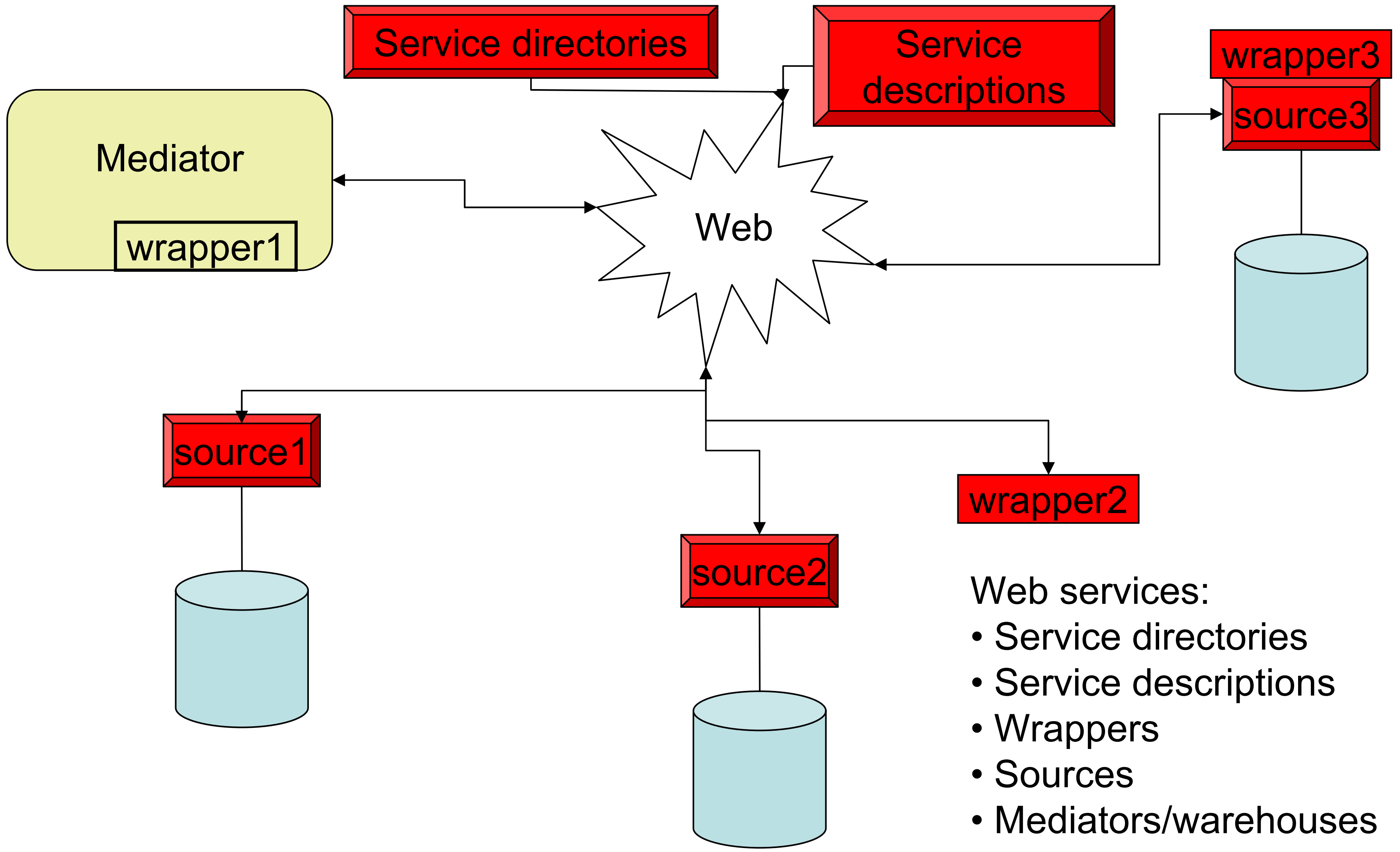
The vision: ***Looking for a Truc***

1. Query a directory (yellowpages): who is a Truc provider?
2. Negotiate with the candidate providers
   - Nature of service
   - Quality/cost/etc.
3. Use the information
   - Get the information
   - Introduce the service in your processing
4. Eventually compose services
5. Eventually publish services

Data integration – Logical view

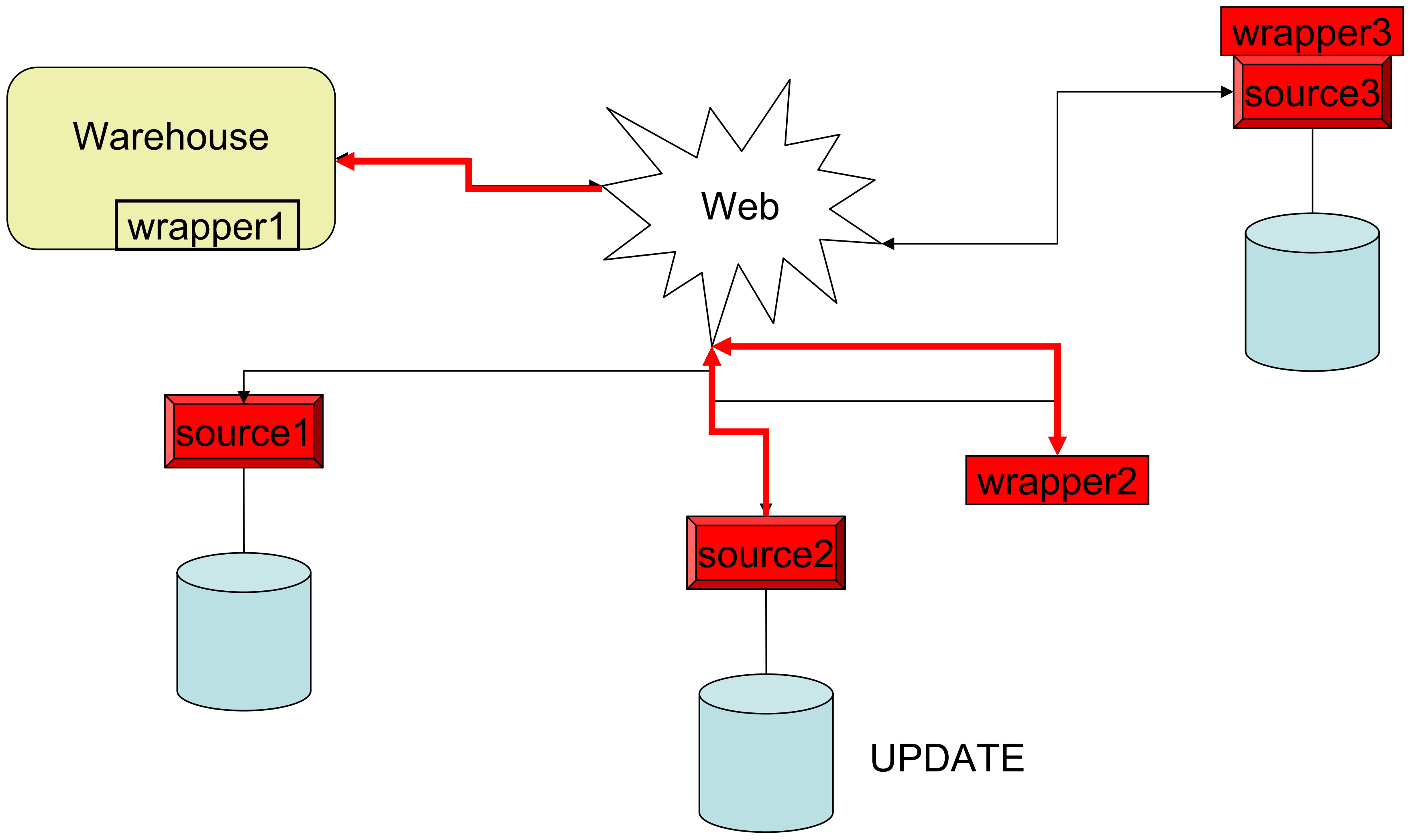


Mediation with web services

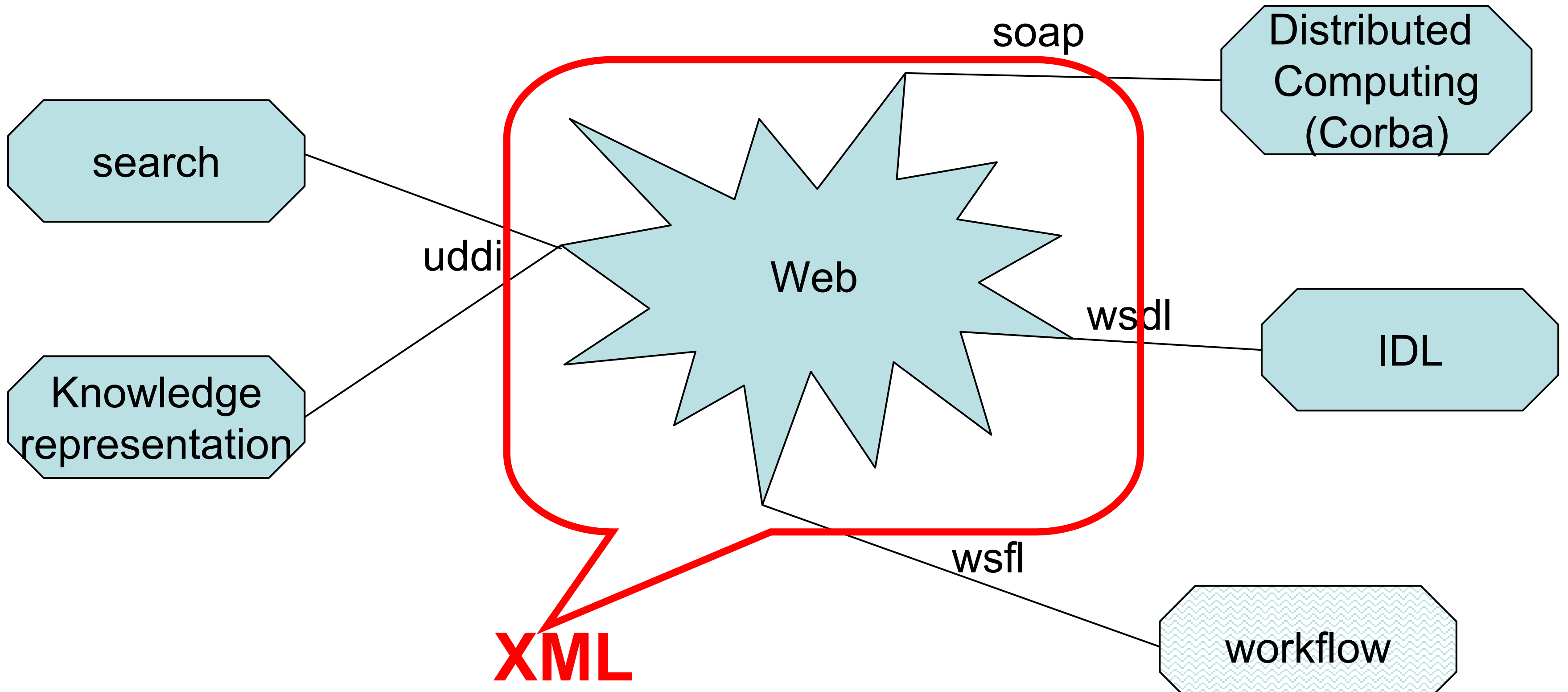


Web services:
- Service directories
- Service descriptions
- Wrappers
- Sources
- Mediators/warehouses

Warehousing with web services



The solution: integration of technologies within web standards



The solution: emerging standards

- XML
- Web services: SOAP
- Service definition for import/export: WSDL; web service description language
- Service composition: WSFL; web service flow language
- UDDI; Universal Description, Discovery and Integration of services

XML

Prerequisite

Recall: labeled ordered trees

+ lots of gadgets: namespace, Xpath, Xlink, Xquery, XML schema…

SOAP
Simple Object Access Protocol

Distributed systems – history

- RPC and DCE (in the seventies)
  - Interface specification via IDL
  - Client-side proxy and server-side stub
  - Link code against proxy/server
- Corba and DCOM (eighties)
  - Interface specification via IDL
  - Object-based
  - Based on names and not physical location

Web services: infrastructure for distributed systems

- Calls based on HTTP+SOAP
- Arguments and results in XML
- Simple protocol (compared to Corba)
- What’s new?
  - – HTTP/XML is universal
  - – Everybody has a browser
  - – Content is rich: XML + all gadgets around (XSL/T, DOM, Xschema, Xquery, …)

SOAP

- XML protocols
  - 1st generation (XML 1.0): WDDX, XML-RPC
  - 2nd generation (namespace and XML schema): SOAP
- XML-RPC
  - Simple exchange of XML data; built on HTTP
  - Lack for extensibility and too limited typing
- SOAP
  - Initiative of Microsoft
  - SOAP 1.1: note to W3C in 1999 by M. and IBM
  - IBM SOAP server donated to Apache + Sun gets in
  - W3C XML Working Group Protocol in 2000

The SOAP beef?

Ubiquitous XML distributed computing infrastructure

- Distributed computing: goal is interoperability between distributed applications
- Ubiquitous: usable everywhere (e.g. on the Internet), so need to be language and platform independent
- XML: data format should be XML with all goodies such as XML schema and namespaces

The SOAP beef?

- Technology: nothing new & relatively simple
- Standards for:
  - SOAP message: unit of communication (body and header)
  - XML as the exchange format (convention for serializing programming language data types in XML)
  - A convention for Remote Procedure Calls
  - A mechanism for extensions (more complex protocols)
  - A binding to HTTP (and more)
  - SOAP fault: error handling

SOAP message embedded in an HTTP request

The envelope is independent of the communication protocol

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8“
Content-Length: nnnn
SOAPAction: "Some-URI“

<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle=
        "http://schemas.xmlsoap.org/soap/encoding/">
        <SOAP-ENV:Body>
                <m:GetLastTradePrice xmlns:m="Some-URI">
                <symbol>DIS</symbol>
                </m:GetLastTradePrice>
        </SOAP-ENV:Body>
   </SOAP-ENV:Envelope>
```

SOAP response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope xmlns:SOAP-ENV=
      "http://schemas.xmlsoap.org/soap/envelope/"
      SOAP-ENV:encodingStyle=
      "http://schemas.xmlsoap.org/soap/encoding/"/>
      <SOAP-ENV:Body>
           <m:GetLastTradePriceResponse
           xmlns:m="Some-URI">  <Price>34.5</Price>
           </m:GetLastTradePriceResponse>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A SOAP Engine: Axis

- Developed by Apache (engineers from HP, IBM, Macromedia)
- Open-source
- Java-based
- Supports recent versions of SOAP
- Runs on top of a simple servlet engine or J2EE application server

http://xml.apache.org/axis

Service Provider View

- Implement the service backend
- Deploy it as a web service
  - Trivial
  - Java web service file “.jws”
  - When the service is invoked, the file is compiled and provides the Web service

Service User View

Accessible from any language and platform

1. Access from its WSDL description $\rightarrow$ find the type and use it [like IDL in Corba or COM]
2. Direct access using a Java library
   - Set the URL of the service
   - There is one method for each operation exposed by the service
   - The signature of the method is exactly that of the service
   - SOAP and XML may be ignored $\rightarrow$ if desired focus on Java types [no need to understand how Java types are marshaled]

Example of a Client

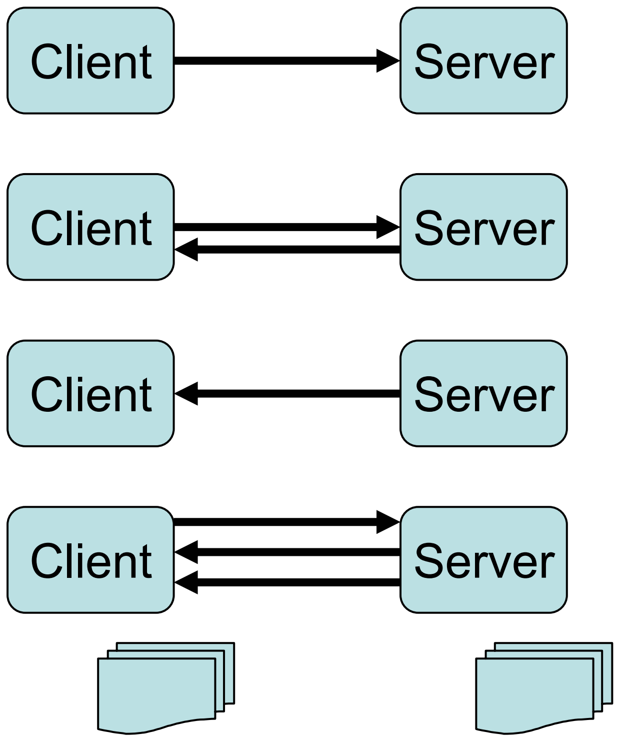
```
package ch3.ex2
import org.apache.axis.client.ServiceClient
public class InventoryCheckClient
{   private String url;
    public InventoryCheckClient(String targetUrl)
    {     url = targetUrl; }
    public boolean doCheck(String sku, int quantity)
    {     ServiceClient call = new ServiceClient(url);
          Boolean result = call.invoke(
                 “”, “doCheck”,
                 new Object[] { sku, new Integer(quantity) } );
          result.booleanValue(); } }
```

Create new service

Invoke service doCheck

Beyond SOAP: Messaging

- 1-to-1; 1-2-many; sync. vs. async.
- Interactions



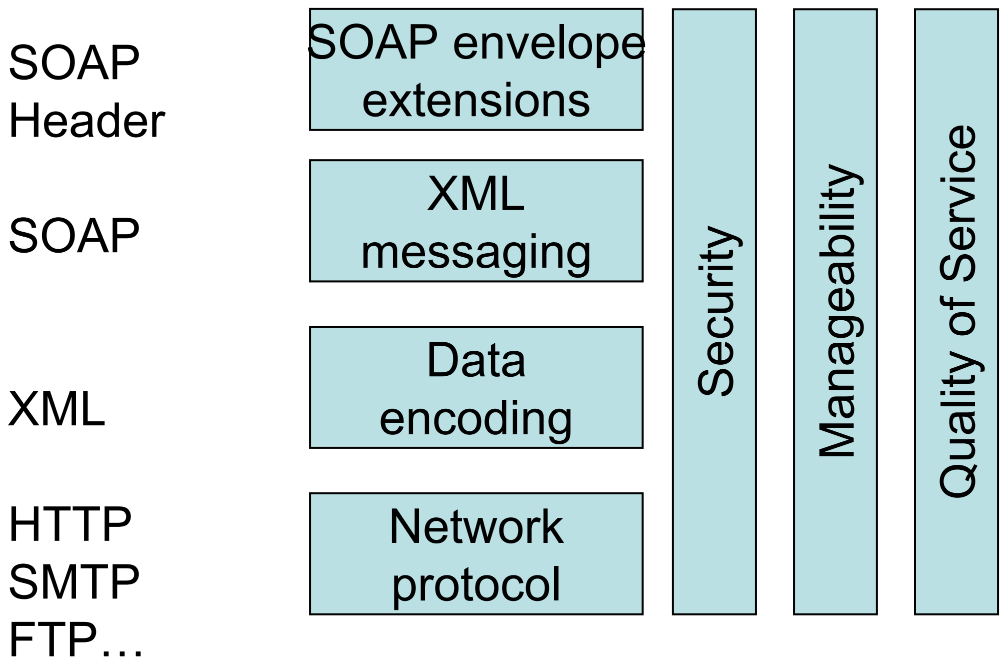
One way message

Query/answer

Notification

Subscription/publication
(possibly with queues)

The wire stack



The technology that determines how a message is sent

What do you need to know to use a service?

- Example: what is the temperature in Paris?
  - Need to know the url of the service
  - That the protocol uses HTTP
  - That it uses SOAP 1.1
  - That request and response use SOAP encoding
  - That request is an RPC with a string as parameter
  - That RPC response is an integer
- In general, more complex: security, authorization, payment, error handling, XML types

Digression: Web service security

- Main functions
  - Confidentiality
  - Authentication
  - Integrity (messages not modified during transport)
  - Non-repudiation
- Main infrastructure
  - Cryptography
  - Public key systems such as RSA

Web service security

- HTTP basic authentication
- SSL: secure socket layer; a protocol for sending encrypted data
- HTTPS = HTTP over SSL: very used
- XML digital signature $\rightarrow$ non repudiation
- XML encryption
  - SSL encrypts the whole message; problem when there are intermediaries
  - XML encryption allows to encrypt selectively

WSDL
Web service definition language

or
what you need to know to use a service

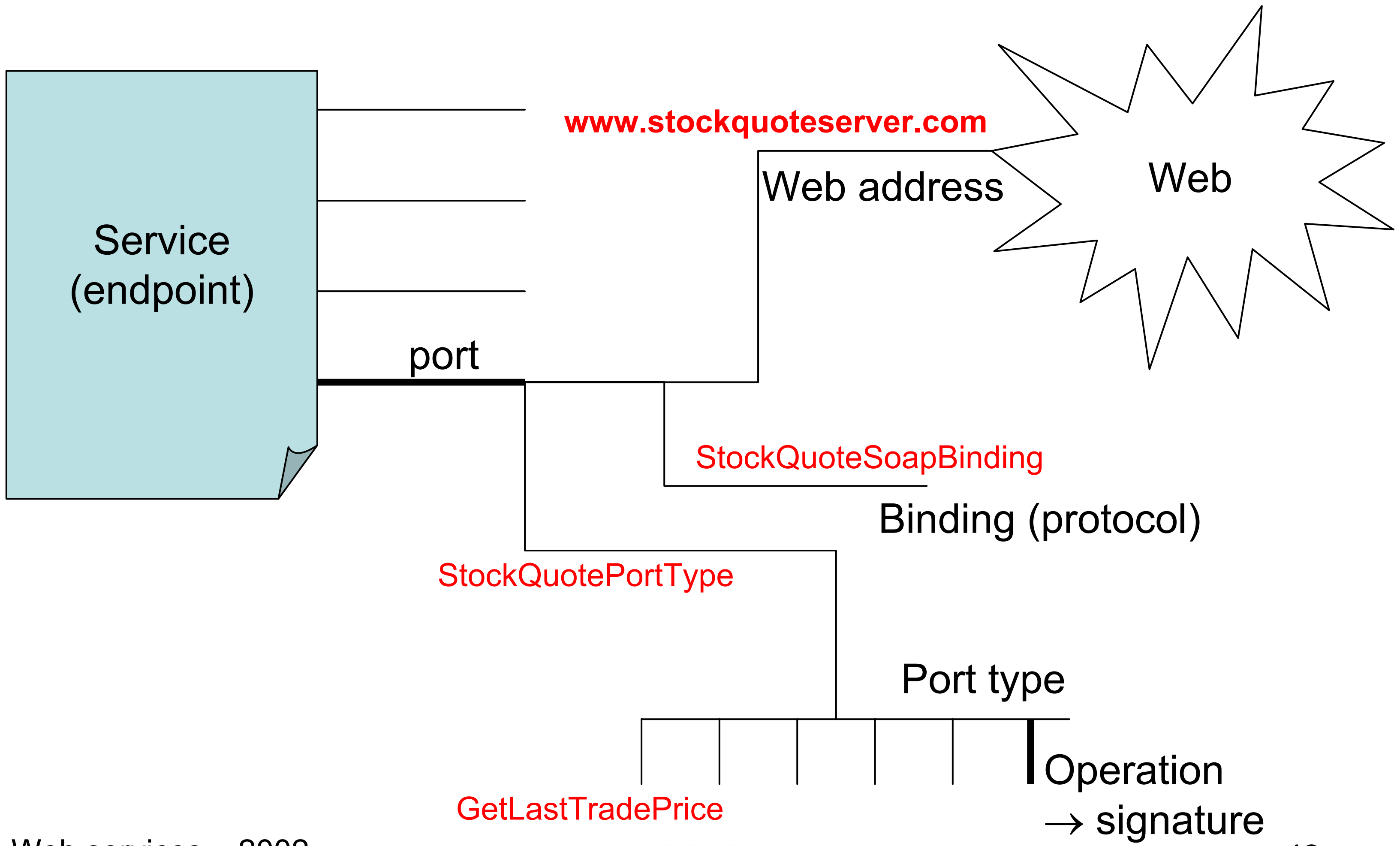
WSDL

- Start 2000: Ariba, IBM, Microsoft
- Version 1.1 submitted to W3C
- XML syntax for describing a service interface
- No session/conversation, transaction

WSDL elements

- Types
- Messages: definition of data that is exchanged
- Operations: method signature
- Port type: collection of operations
- Binding: concrete protocol for a port type
- Port: a web address for a binding
- Service: a collection of ports

An abstract vision of a service

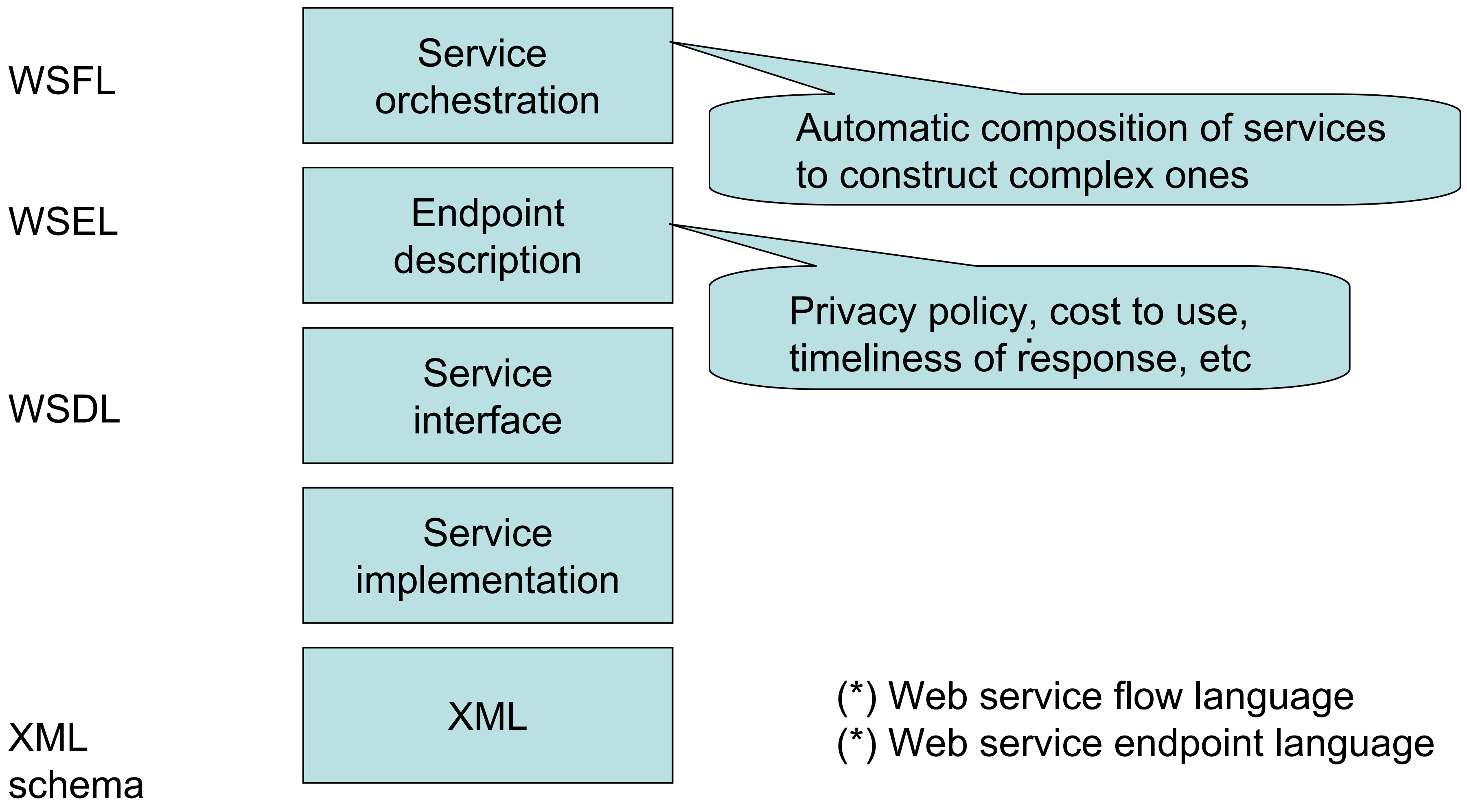


```
<?xml version="1.0"?>
     <definitions name="StockQuote"

     targetNamespace="http://example.com/stockquote.wsdl"
          xmlns:tns="http://example.com/stockquote.wsdl"
          xmlns:xsd1="http://example.com/stockquote.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">
       <types>
         <schema targetNamespace=
              "http://example.com/stockquote.xsd"
            xmlns="http://www.w3.org/2000/10/XMLSchema">
           <element name="TradePriceRequest">
            <complexType> ….
       </types>
```

```
<message name="GetLastTradePriceInput">
   <part name="body"
      element="xsd1:TradePriceRequest"/>
</message>

<message name="GetLastTradePriceOutput">
   <part name="body" element="xsd1:TradePrice"/>
</message>

<portType name="StockQuotePortType">
   <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceInput"/>
      <output message="tns:GetLastTradePriceOutput"/>
   </operation>
</portType>
```

```
  <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
    <soap:binding style="document" transport=
              "http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastTradePrice">
      <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>   <soap:body use="literal"/>  </output>
    </operation>
  </binding>

  <service name="StockQuoteService">
    <documentation>My first service</documentation>
    <port name="StockQuotePort" binding="tns:StockQuoteBinding">
      <soap:address location="http://example.com/stockquote"/>
    </port>
  </service>

</definitions>
```

The service description stack


WSFL
Service orchestration
Automatic composition of services to construct complex ones
WSEL
Endpoint description
Privacy policy, cost to use, timeliness of response, etc
WSDL
Service interface
Service implementation
XML schema
XML

(*) Web service flow language
(*) Web service endpoint language

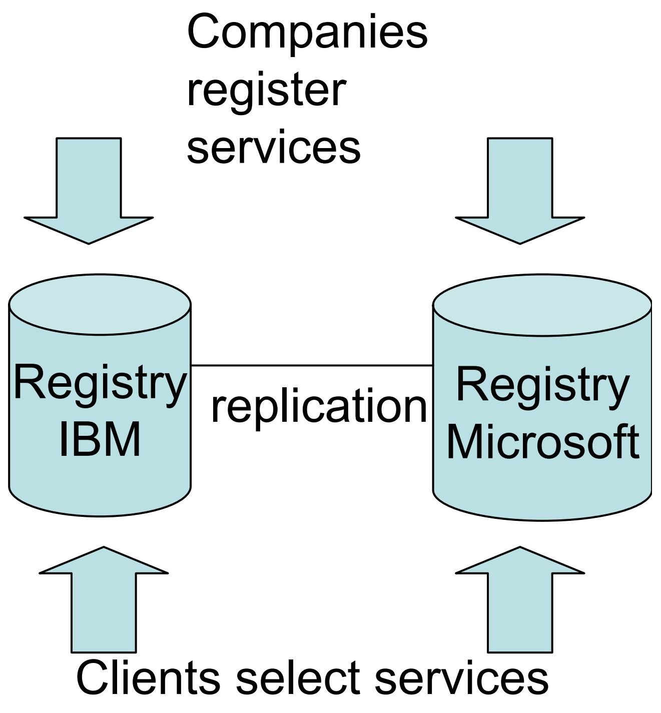
UDDI
Universal Description, Discovery and Integration of services

UDDI

- Where may I find the service I need?
  - Who are the candidates?
  - Which one is the best?
- Core: directories – yellow pages
  - List companies + contact info
  - Classification
  - More information: protocol, cost, quality, contract…
- Who is in charge?
  - E.g.: who controls the categories? Who can publish in the directory?

UDDI (continued)

- Industry consortium with big guys (IBM, Microsoft)
- Means to publish and find
- Lots of noise
- Extremely limited so far
  - Not many services
  - Query language very primitive
  - Information very limited



UDDI (continued)

Business registry: an XML file describing the business and its e-services

- White pages: contact information (address, phone number, etc.)
- Yellow pages: description of business and services based on some ontologies
- Green pages: technical information on the services

Example

```
<businessService businessKey="SevresPratique" serviceKey="Sevres@Ouvaton">
    <description xml:lang="french">  content provider for Sèvres, 92310,
    France</description>
    <bindingTemplate>…%technical information
    <bindingTemplate>…
</businessService>

<bindingTemplate serviceKey="Sevres@Ouvaton" bindingKey="">
    <accessPoint urlType="http">http://sevres-pratique.com</accessPoint>
    <tModelinstanceDetails>…</tModelinstanceDetails>
<bindingTemplate>
```

Tmodel: explains how to interact with a service
- can use WSDL to describe the interface
- can use others, e.g., rosettaNet PIP (e-commerce)

Technical service description

UDDI API

- Publish: requires registration with operator of the registry HTTPS
- Inquiry: search/browse HTTP
- Replication between several registries
- Ontologies: open in UDDI
  - NAICS: industry codes
  - UNSPSC: product and services
  - ISO3166

Digression: electronic business XML ebXML

- 1999: United Nations Center for Trade Facilitations and E-business & Organization for the Advancement of Structured Information Standards
- Also based on XML
- Overlap with the SOAP/WSDL/UDDI approach
- E.g.: ebXML registry and repository; registration of business metadata & UDDI

The field

+ Very active: lots of fun
– Too active → noise: difficult to stay up-to-date
+ Very strong on standards
– Lots of standards that die overnight
+ Very simple (do not get impress by huge documentations that are often content free)
– Very heavy (you have to read huge documentations that are often content free)
+ Lots of free software to play with
– Most of them are not very reliable

Active XML

or *putting XML+SOAP+Xquery to work for data management*

Conclusion

- Distributed data management
  - lots of opportunities because of the combination of XML and Web services
  - **Distributed *ubiquitous* data management**
  - Peer-to-peer data management
- Lots of research issues:
  - Optimization, maintenance: classical
  - Surveillance
  - Discovery of information
  - Integration at web scale: requires more AI

References

Short bibliography

- XML – Extensible Markup Language XML (W3C): http://www.w3.org/XML/
- Xquery – XML Query: http://www.w3.org/XML/Query
- Xschema – XML Schema: http://www.w3.org/XML/Schema
- XPWG – XML Protocol Working Group: http://www.w3.org/2000/xp/Group/
- HTTP – Hypertext Transfer Protocol: http://www.w3.org/Protocols/
- SOAP – Simple Object Access Protocol: http://www.w3.org/TR/SOAP/
- WSDL – Web Services Description Language: http://www.w3.org/TR/wsdl
- OMG – Object Management Group : http://www.omg.org/
- CORBA – Common Object Request Broker Architecture, see OMG
- UDDI – Universal Description, Discovery, and Integration: http://www.uddi.org/
- WSFL – Web Services Flow Language: http://xml.coverpages.org/wsfl.html
- Apache Axis – http://xml.apache.org/axis/
- J2EE – Java 2 Platform, Enterprise Edition: http://java.sun.com/j2ee/

Short bibliography (continued)

- *Building Web Services with Java*: Making Sense of XML, SOAP, WSDL and UDDI, Steve Graham (Editor),
- Ronald L. Rivest, Adi Shamir, Leonard M. Adleman: *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. CACM 21(2): 120-126 (1978)
- Serge Abiteboul, Peter Buneman, Dan Suciu: *Data on the Web: From Relations to Semistructured Data and XML*. 1999
- Serge Abiteboul, Omar Benjelloun, Tova Milo, Ioana Manolescu, Roger Weber, *Active XML: A Data-Centric Perspective on Web Services*: http://osage.inria.fr/verso/PUBLI/display-abstract.php?id=213
- M. Tamer Özsu, Patrick Valduriez: *Principles of Distributed Database Systems*, Second Edition. 1999
- Maarten Van Steen, Andrew S. Tanenbaum, *Distributed Systems: Principles and Paradigms*

Some web service software's

- Web service support for languages
  - Many for Java and C++
    - Apache Axis; Mind electric GLUE
  - SOAP::Lite for perl
  - Sole for Python: SOAP.py
- J2EE: Java 2 Platform Edition Edition
  - BEA, Ioana, IBM, Macromedia
  - Push on web services
- Microsoft .NET – lots of software for web services

Merci