Web-services

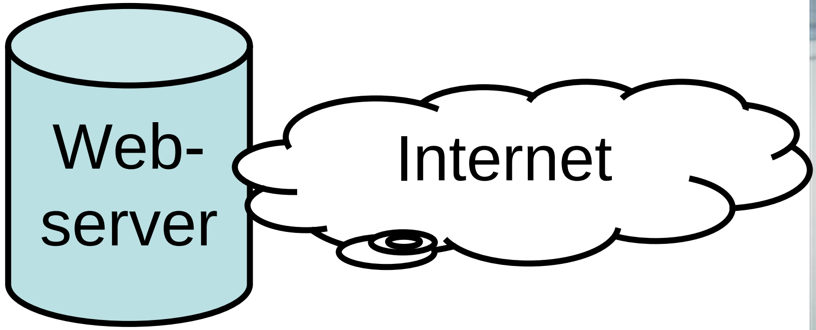
Brian Nielsen

bnielsen@cs.aau.dk

Why Web Services?

Today's Web

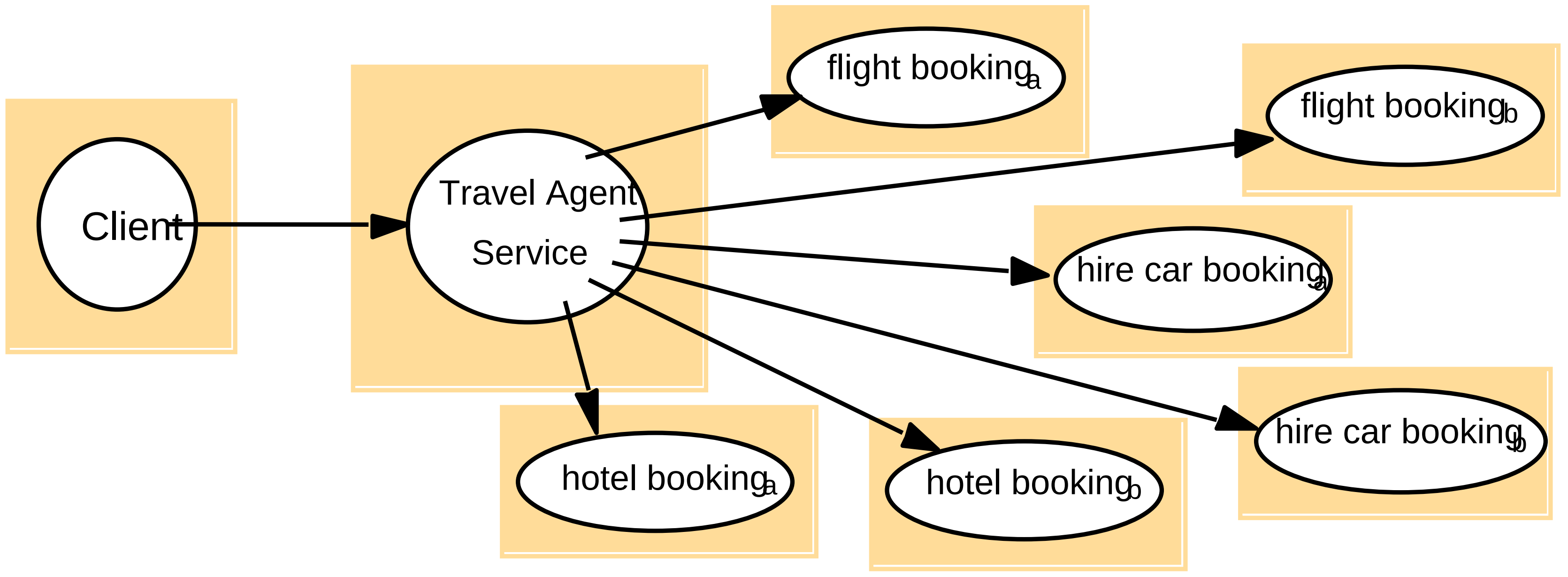
- Web designed for application to human interactions
  - Information sharing: a distributed content library.
  - Enabled Business-to-costumer e-commerce
  - Non-automated B2B interactions
- How did it happen?
  - Built on very few standards: http + html
  - very few assumptions made about computing platforms
  - Result was ubiquity (Existence everywhere at the same time)



What’s next?

- The Web is everywhere. There is a lot more we can do!
  - E-marketplaces
  - Open, automated business-to-business e-commerce
  - *Business process integration on the Web*
  - *Resource sharing, distributed computing.*
  - Pervasive Embedded Computing
- Current approach is *ad-hoc*
  - Proprietary protocols
  - e.g., application-to-application interactions with HTML forms.
- Goal:
  - **Enable the creation of applications that are built by combining loosely coupled and interoperable services**
  - **enabling systematic application-to-application interaction on the Web**

Combination of web services

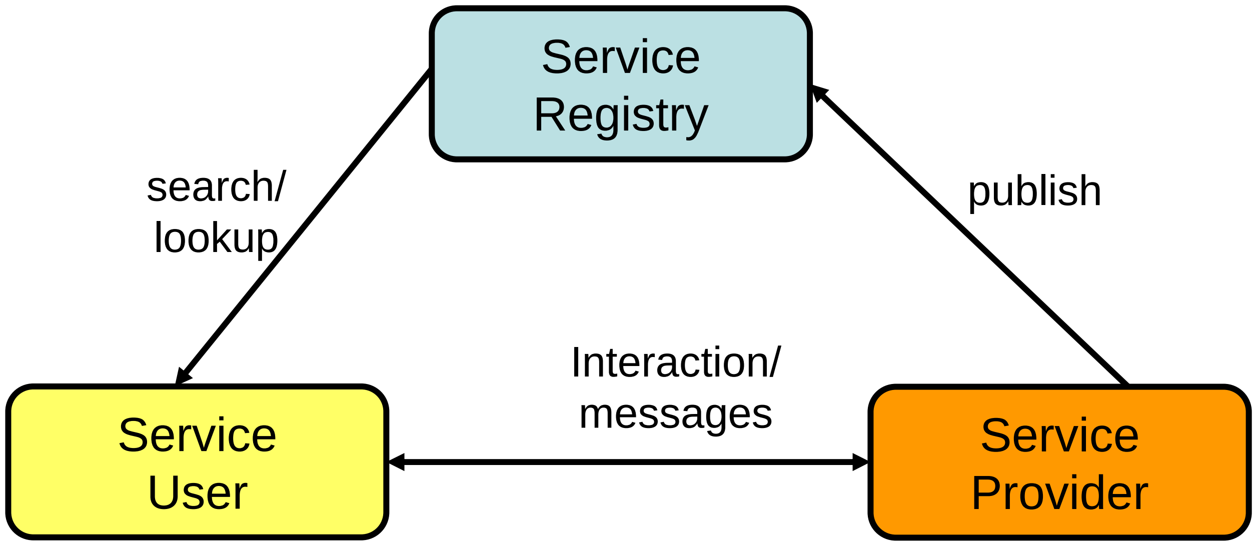


Travel agent scenario

1. The client asks the travel agent service for information about a set of services; for example, flights, car hire and hotel bookings.
2. The travel agent service collects prices and availability information and sends it to the client, which chooses one of the following on behalf of the user:
   (a) refine the query, possibly involving more providers to get more information, then repeat step 2;
   (b) make reservations;
   (c) quit.
3. The client requests a reservation and the travel agent service checks availability.
4. Either all are available;
   or for services that are not available;
      either alternatives are offered to the client who goes back to step 3;
      or the client goes back to step 1.
5. Take deposit.
6. Give the client a reservation number as a confirmation.
7. During the period until the final payment, the client may modify or cancel reservations

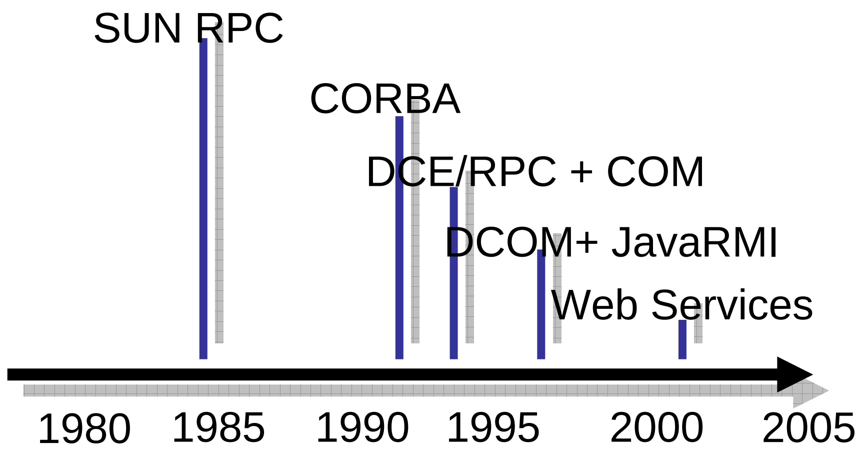
Service Oriented Architecture

- ***Develop large scale applications from (distributed) collections of smaller loosely-coupled service providers***



What are Web Services

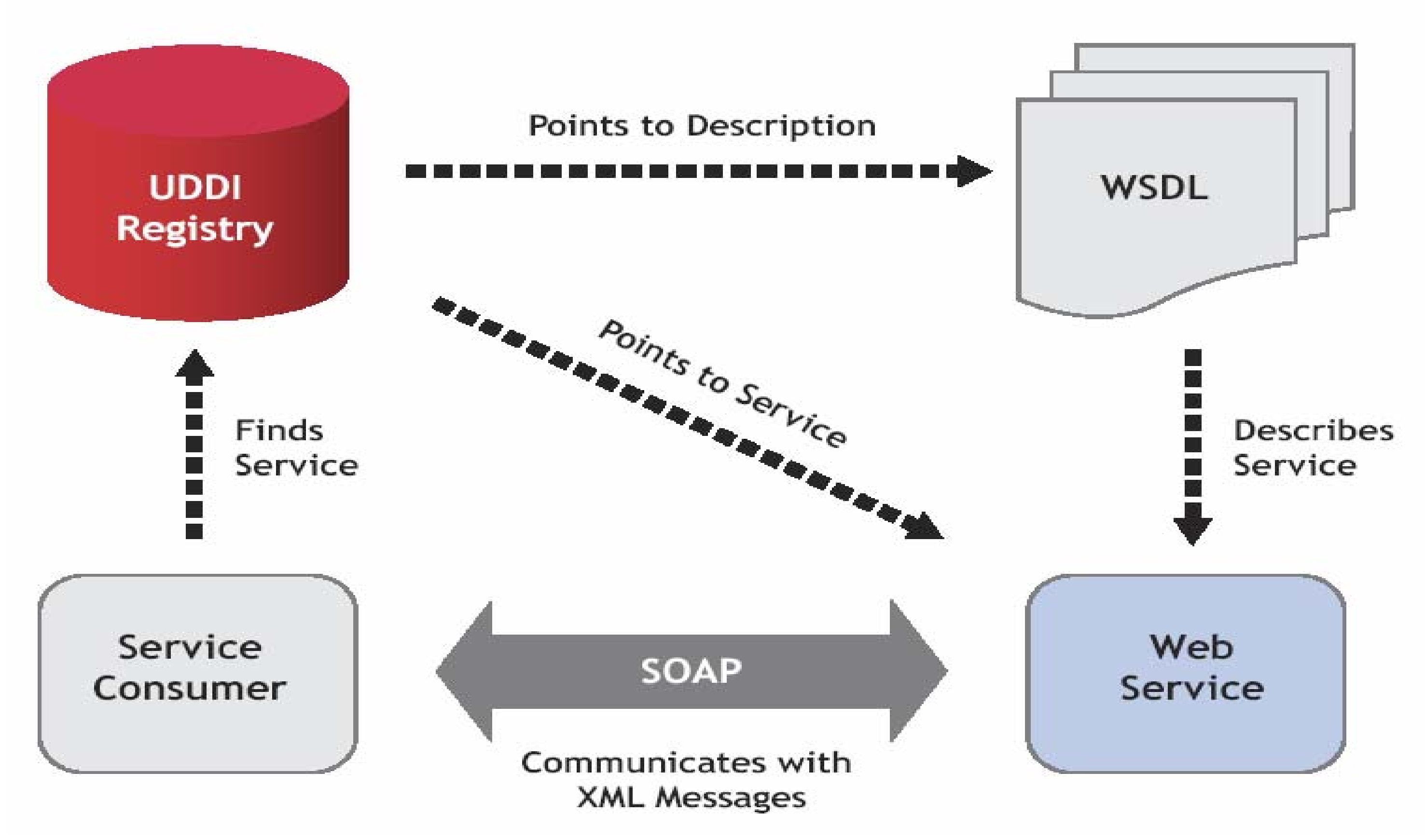
- ***"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network."***
- Distributed computing for the Web:



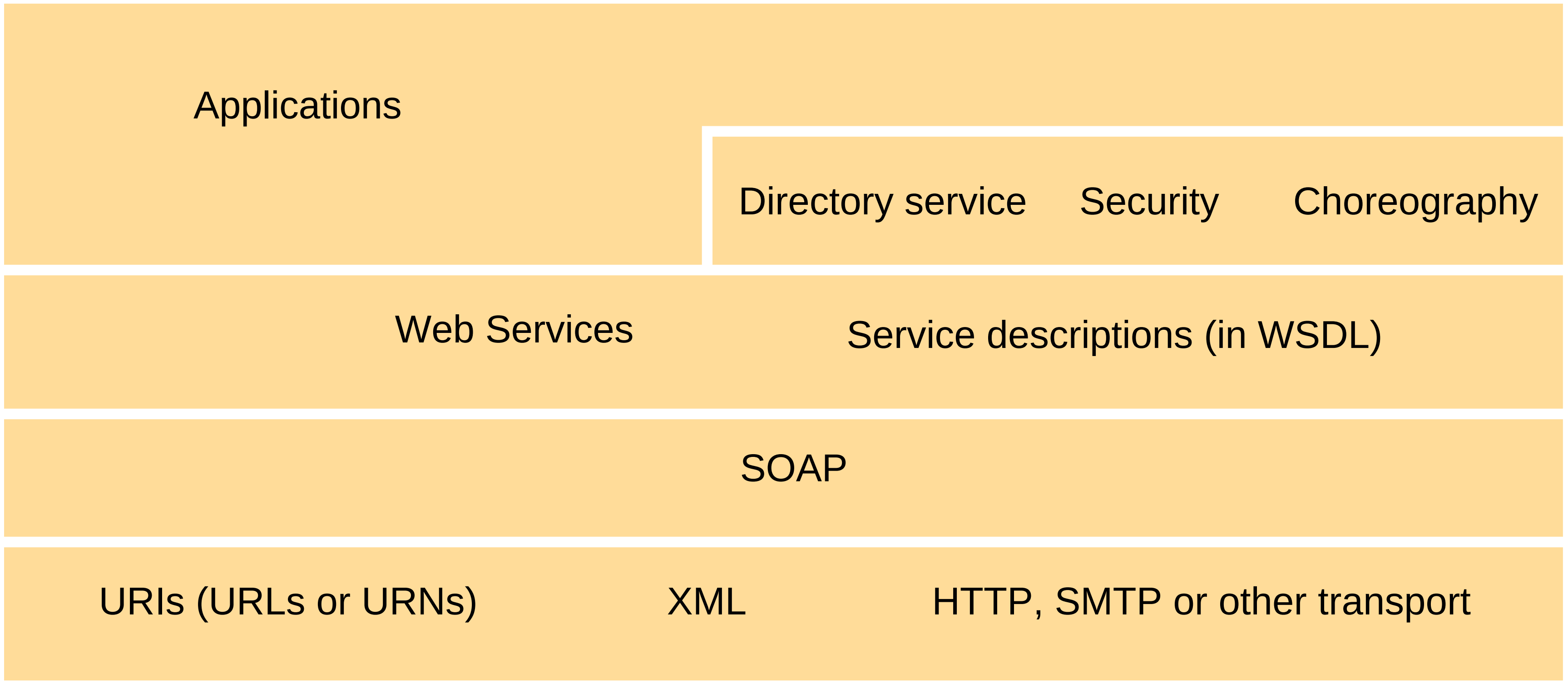
  - Internet-wide
  - Platform neutral, open Standards
  - Based on ubiquitous software (XML, HTTP)

Web Services Components

Web-based Service Oriented Architecture

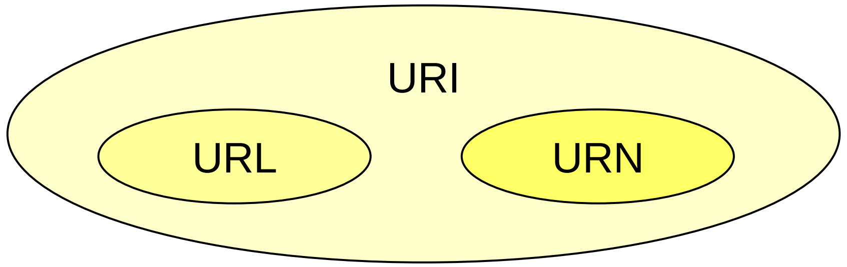


Web services infrastructure and components



Recall XML

URI / URL / URN



- URI (Uniform Resource Identifier): a general ressource identifier, whose value may be either URL or URN
  - URL: includes resource location information
    - *http://www.cdk4.net/person*
  - URN: (Uniform Resource Names): location independent, rely on lookup service to map them onto the URLs of resources
    - *urn:isbn:0-321-26354-5*

XML

- XML (extensible markup language) is defined by the World Wide Web Consortium (W3C)
- Both XML and HTML were derived from SGML (Standardized Generalized Markup Language)
  - HTML: tags specify how a browser displays the text
  - XML: tags **describe the logical structure of the data**
- XML is extensible: users can define their own tags (HTML uses a fixed set of tags)
- Generic tools:
  - parsing, validating, querying, translating, …

XML Definition of a Person

```
<person id="123456789">
            <name>Smith</name>
            <place>London</place>
            <year>1934</year>
            <!-- a comment -->
</person >
```

- Element: *<name>Smith</name>*
- Attribute: *id="123456789“*
- items represented as elements or attributes:
  - An element is generally a container for data
  - An attribute is used for labeling that data

Name Spaces

- XML name spaces: provide scoping of names to avoid name-clashes
- URIs are used to identify namespaces (specified in xmlns attribute)
- URI is a cheap way of getting unique names
  - Developer controls hierarchy under the given URI.
  - Doesn’t necessarily point to anything
- In the example, *pers* is shorthand for http://www.cdk4.net/person

```
<person pers:id="123456789" xmlns:pers = "http://www.cdk4.net/person“ >
        <pers:name> Smith </pers:name>
        <pers:place> London </pers:place >
        <pers:year> 1934 </pers:year>
</person>
```

XML-Schema

- A schema defines the legal structure (grammar) of an XML document
  - elements and attributes that can appear in a document,
  - how the element are nested and the number of elements,
  - whether an element is empty or can include text.
- For each element, it defines the type and default value
- Schema Languages: DTD, XML-Schema
- An XML document may be validated against a schema
- May be transformed (XSLT) Navigated / queried /Language bindings

```
<xsd:schema  xmlns:xsd = URL of XML schema definitions   >
    <xsd:element name= "person" type ="personType" />
        <xsd:complexType name="personType">
            <xsd:sequence>
                <xsd:element name = "name"  type="xs:string"/>
                <xsd:element name = "place"  type="xs:string"/>
                <xsd:element name = "year"  type="xs:positiveInteger"/>
            </xsd:sequence>
            <xsd:attribute name= "id"   type = "xs:positiveInteger"/>
        </xsd:complexType>
</xsd:schema>
```

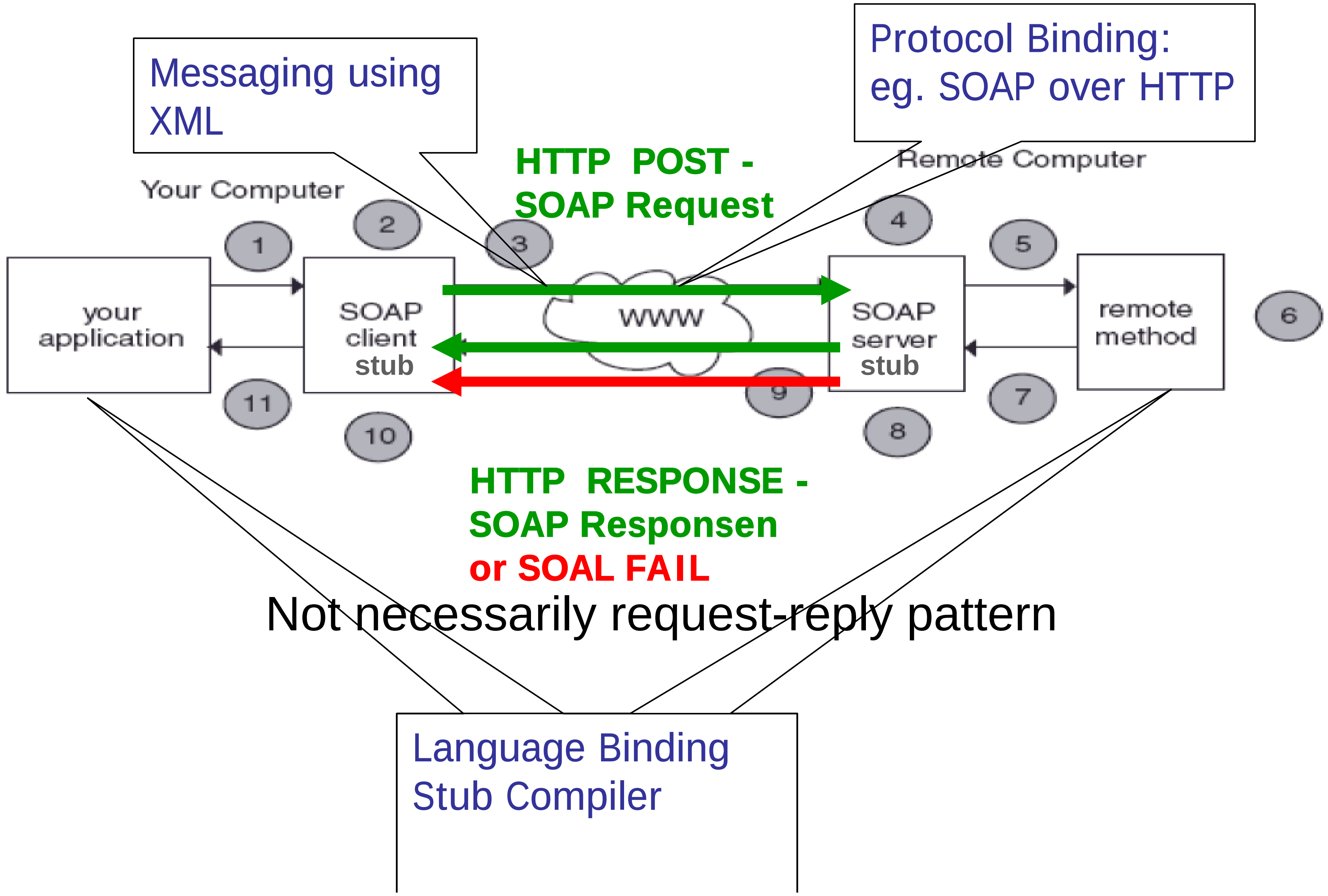
SOAP

SOAP

- Used to mean **Simple Object Access Protocol**
- From SOAP 1.2 > SOAP is no longer an acronym

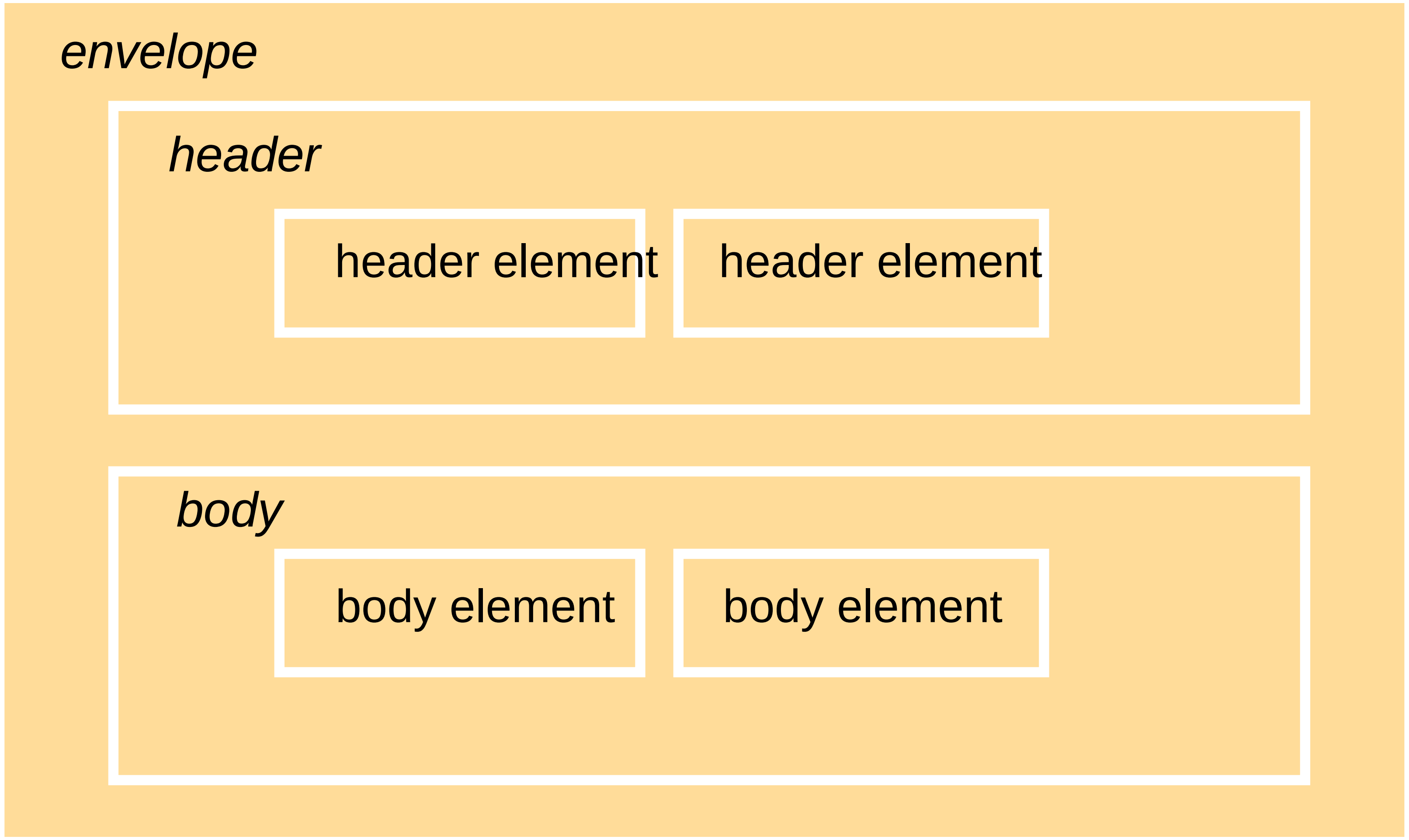
- XML-based
  - uses XML to represent the contents of request and reply messages
  - Platform independent, language independent
- Transport:
  - HTTP
  - SMTP, FTP, TCP or UDP, (Jabber)
- Language Binding:
  - SOAP APIs available for many programming languages,
  - Java, Javascript, Perl, Python, .NET, C, C++, C#, and VB
  - Programmers do not normally need to concern how SOAP uses XML to represent messages and HTTP to communicate them

Web Services using SOAP



Not necessarily request-reply pattern

SOAP message Enveloping



SOAP Message

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV
  ="http://schemas.xmlso
  ap.org/soap/envelope/"
  >

    < SOAP-ENV:Header>
     ...
    </ SOAP-ENV:Header>

    < SOAP-ENV:Body>
      ...
    </ SOAP-ENV:Body>
   ...
</ SOAP-ENV: Envelope>
```

Envelope

- specifies global settings, eg encoding.

Header

- Optional
- Ultimate destination
- encryption
- routing & delivery settings
- authentication/authorisation information
- transaction context
- Other data extensions

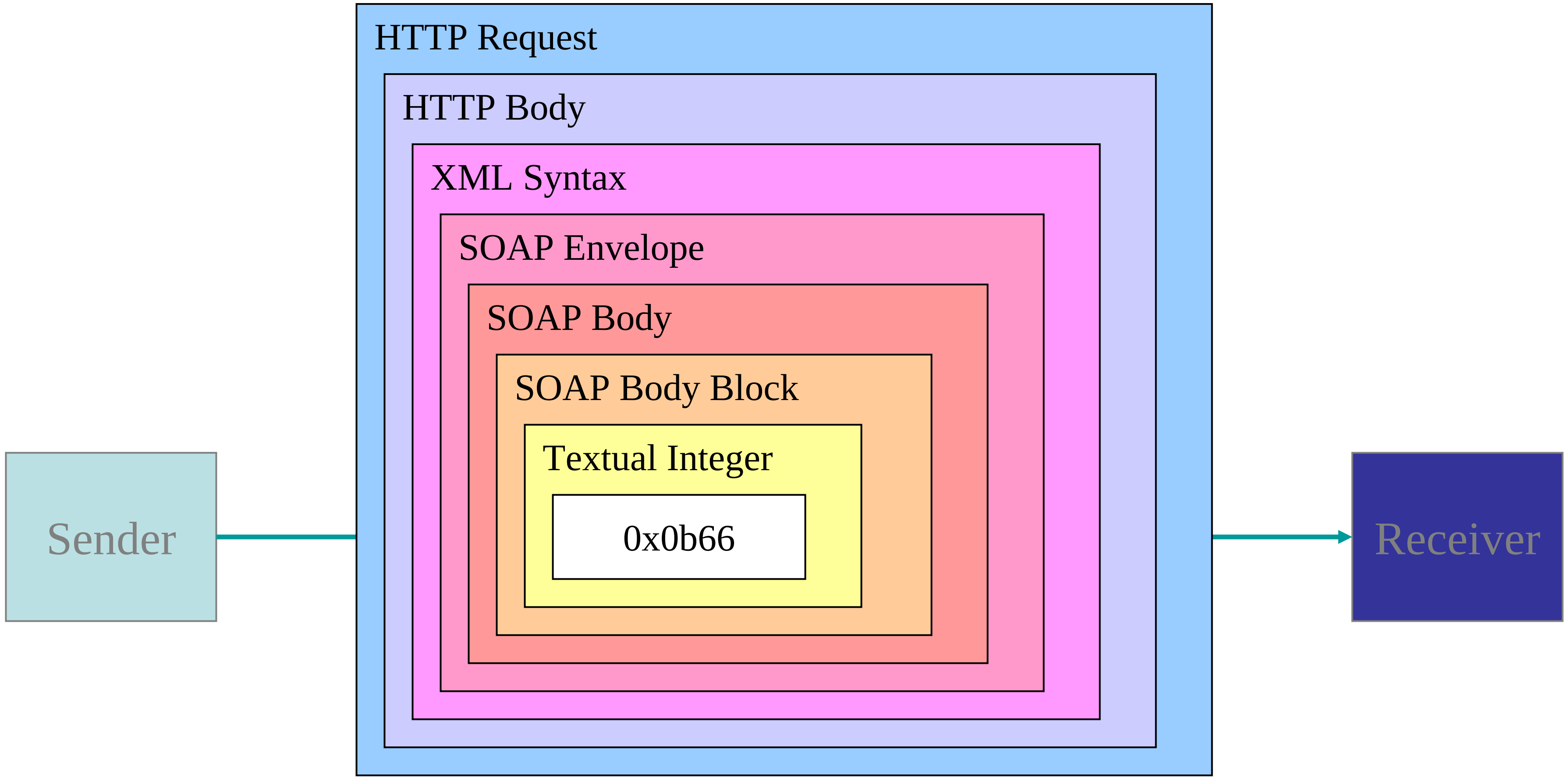
Body

- required
- data or message to be processed
- can contain anything that can be expressed in XML
- containing as many child nodes as required

IP 2 Location demo

1. http://www.fraudlabs.com/
2. http://www.fraudlabs.com/docs/IP2Proxy_Web_Service_Documentation.pdf (p10)
3. http://ws.fraudlabs.com/ip2locationwebservice.asmx?wsdl

(In)Efficiency



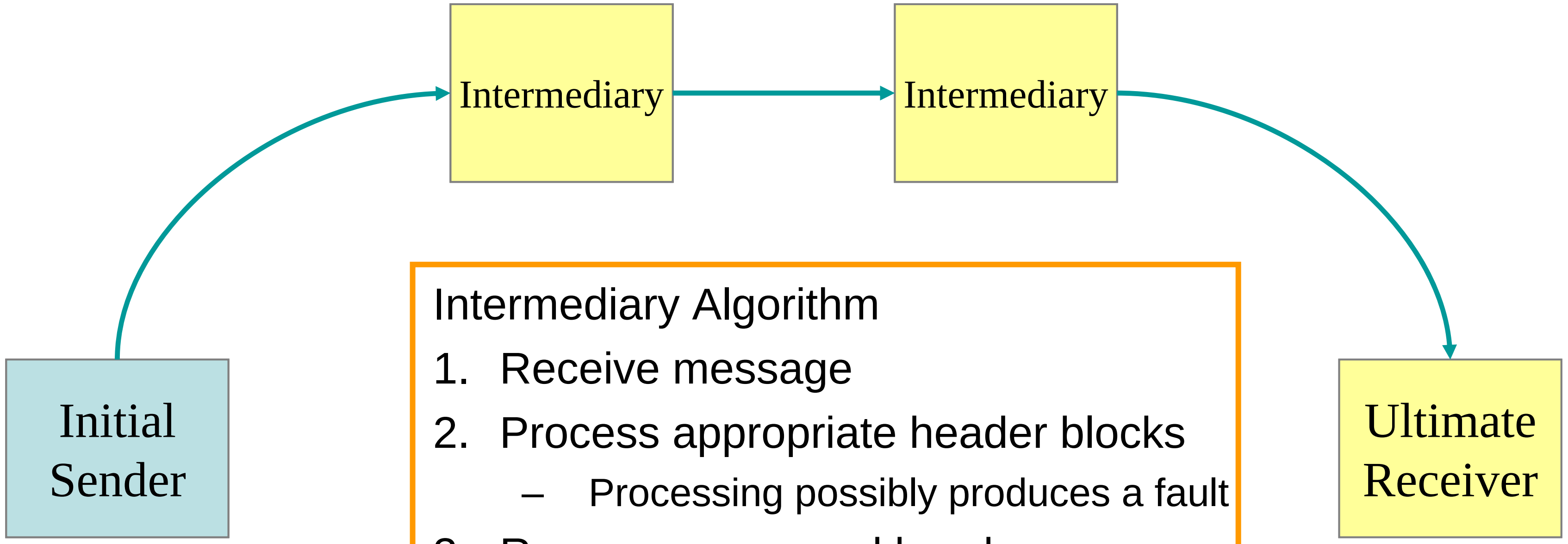
•SOAP requests may be 14 times longer than CORBA

•SOAP requests may take 882 times as long as CORBA

SOAP Nodes

Soap message may be destined to a set of intermediary nodes as well as an ultimate receiver

- En/de-cryption, compression, load-balancing, access control, auditing, routing, monitoring



Intermediary Algorithm

1. Receive message
2. Process appropriate header blocks
   - Processing possibly produces a fault
3. Remove processed headers
4. Add new headers
5. Send message

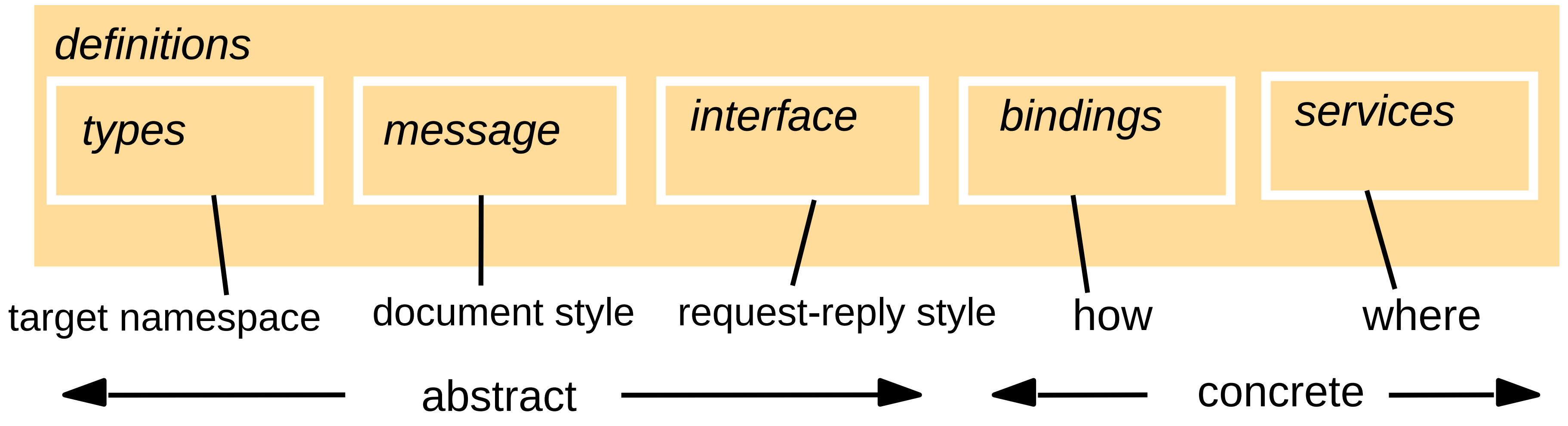
WSDL

Web Services Description Language

- WSDL (“Whistle”) W3 standard
- XML-based language describing:
  - What functionality is provided?
  - How should it be accessed?
  - Where is the service located?

1. Implementation language independent interface description
2. Allows advertisement of service descriptions, enables dynamic discovery and binding of compatible services.
   - Used in conjunction with UDDI registry
3. Generate compatible client and server stubs.
   - wsdl2java
   - Java2wsdl
4. **Allows industries to define standardized service interfaces.**

WSDL description

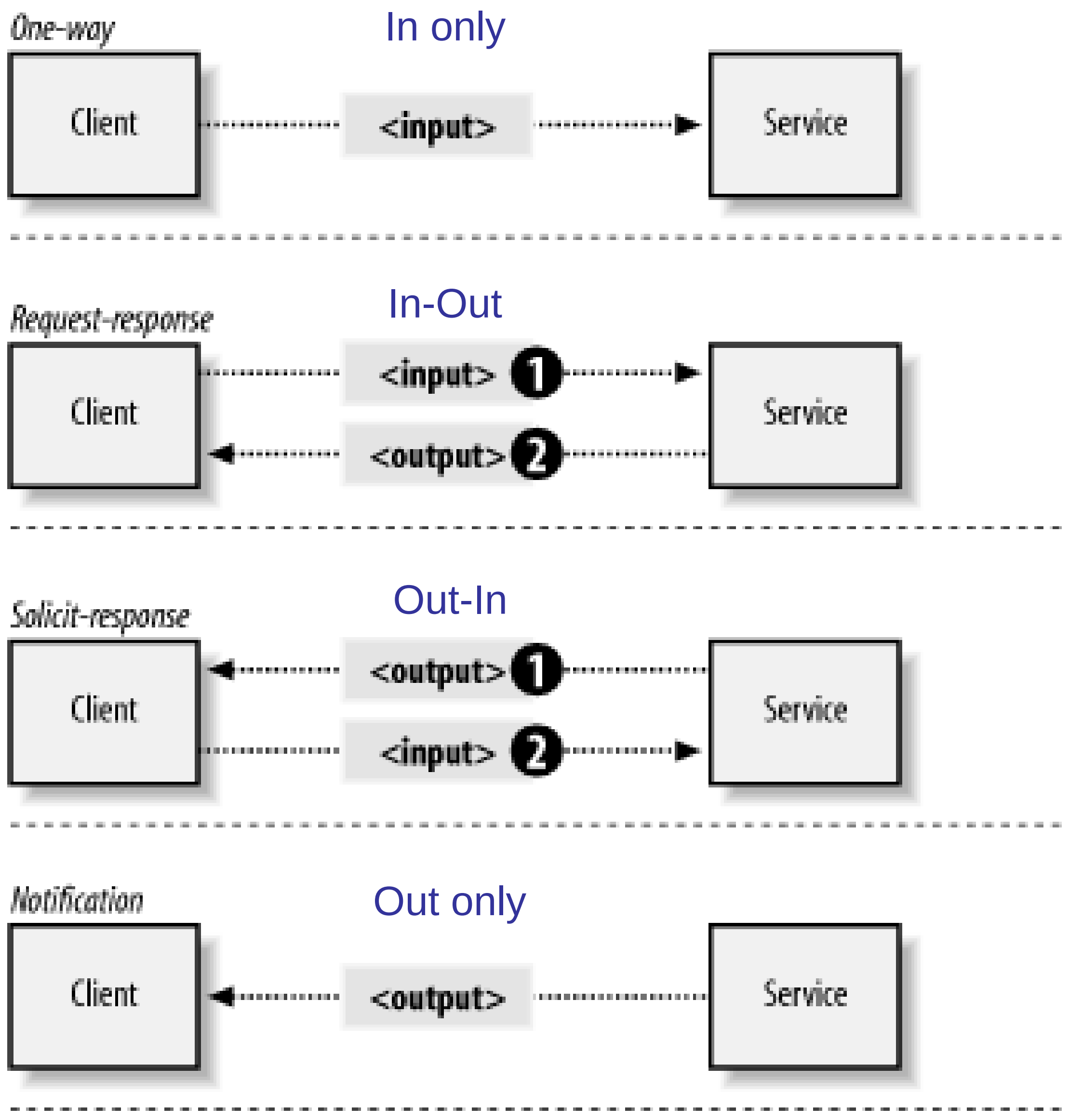


- Types: XML schema describing the used data types
- Message: The structure of the messages exchanged
- Interface: Information describing all publicly available functions
- Bindings: Information about the transport protocol to be used
- Services: Address information for locating the specified service

WSDL Operation patterns

1. One-way
2. Request-response
3. Solicit-response
4. Notification

Robust Versions of in-only and out-only gives failure response



IP 2 Location demo

1. http://www.fraudlabs.com/
2. http://www.fraudlabs.com/docs/IP2Proxy_Web_Service_Documentation.pdf (p10)
3. http://ws.fraudlabs.com/ip2locationwebservice.asmx?wsdl

Recipe Server

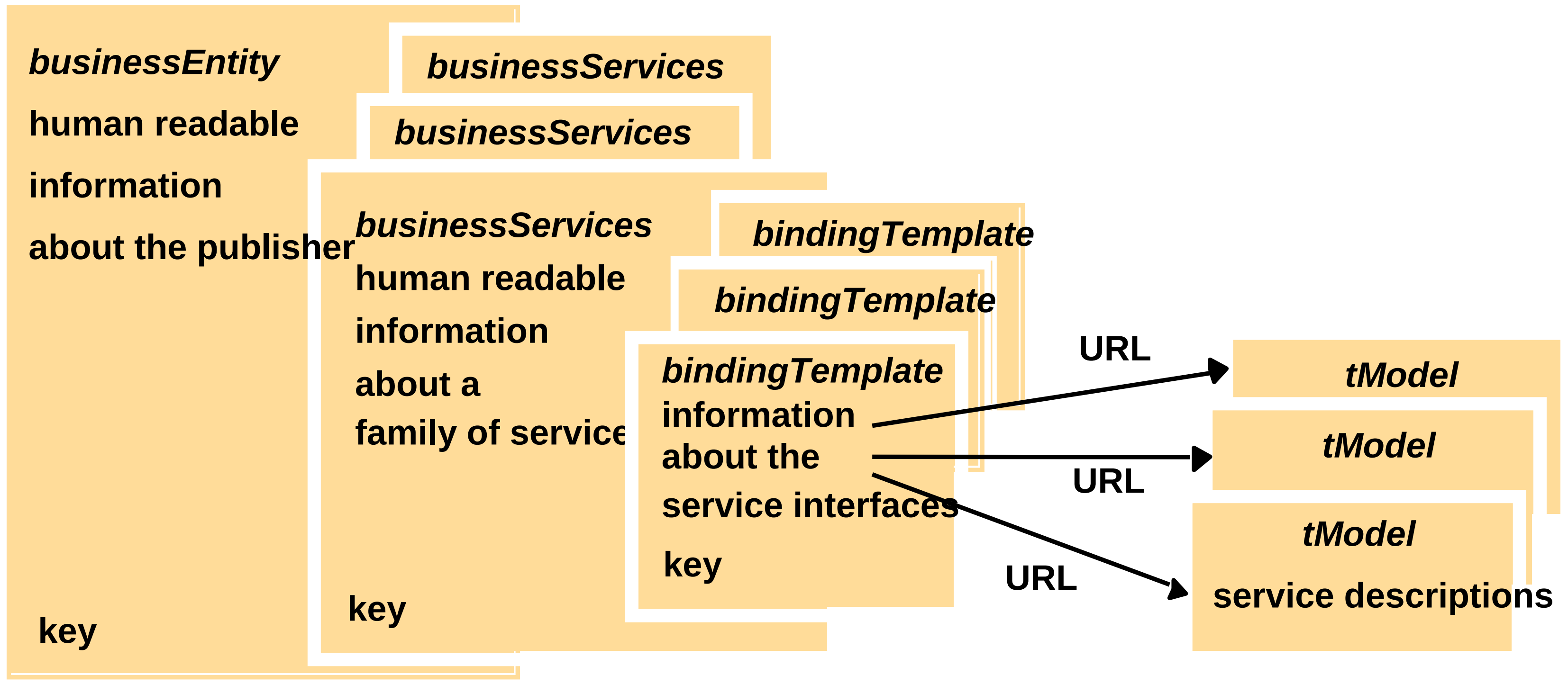
- From “An introduction to XML and Web Technologies” by Anders Møller & Michael I. Schwartzbach
  - http://www.brics.dk/ixwt/examples/recipeserver.wsdl
  - http://www.brics.dk/ixwt/examples/recipes.xsd
  - http://www.brics.dk/ixwt/examples/recipes.xml
  - http://www.brics.dk/ixwt/examples/recipes.xsl
- Operations on recipe collection
  - getRecipe: returns the collection of recipes stored at the server
  - lockRecipe: obtains lock of recipe ID (or fails)
  - writeRecipe: upload recipe
  - unlockRecipe: Releases given lock

UDDI

UDDI

- Universal Description, Discovery and Integration
- A UDDI Server acts as a registry for Web Services and makes them searchable.
  - White pages (general information)
  - Yellow pages (categories of services)
  - Green pages (business rules)
- Accessible as web service and html
- http://soapclient.com/uddisearch.html

The main UDDI data structures

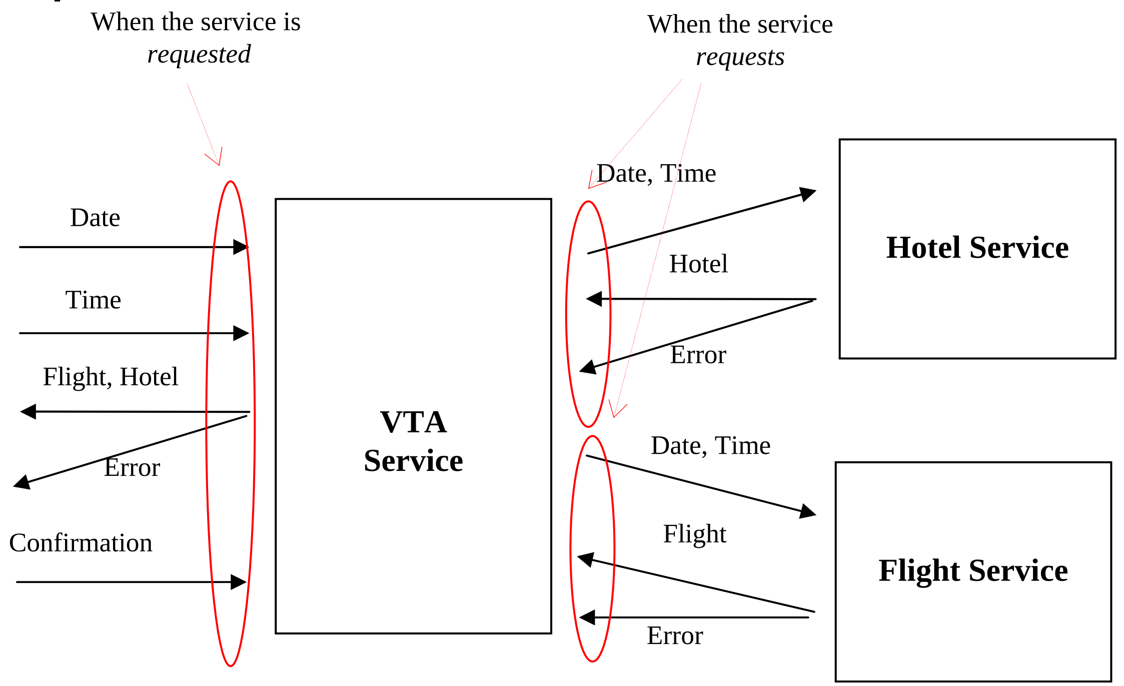


WS-Extensions

- WS-security, WS-choreography
- WS-flow, WS-reliability, WS-transactions, WS-membership, WS-CDL, WS-BPEL, WS-Events, WS-Policy, WS-Routing,…
- “WS*”
  - http://www.dotnet-collective.com/ws_extensions.html

Choreography & Orchestration

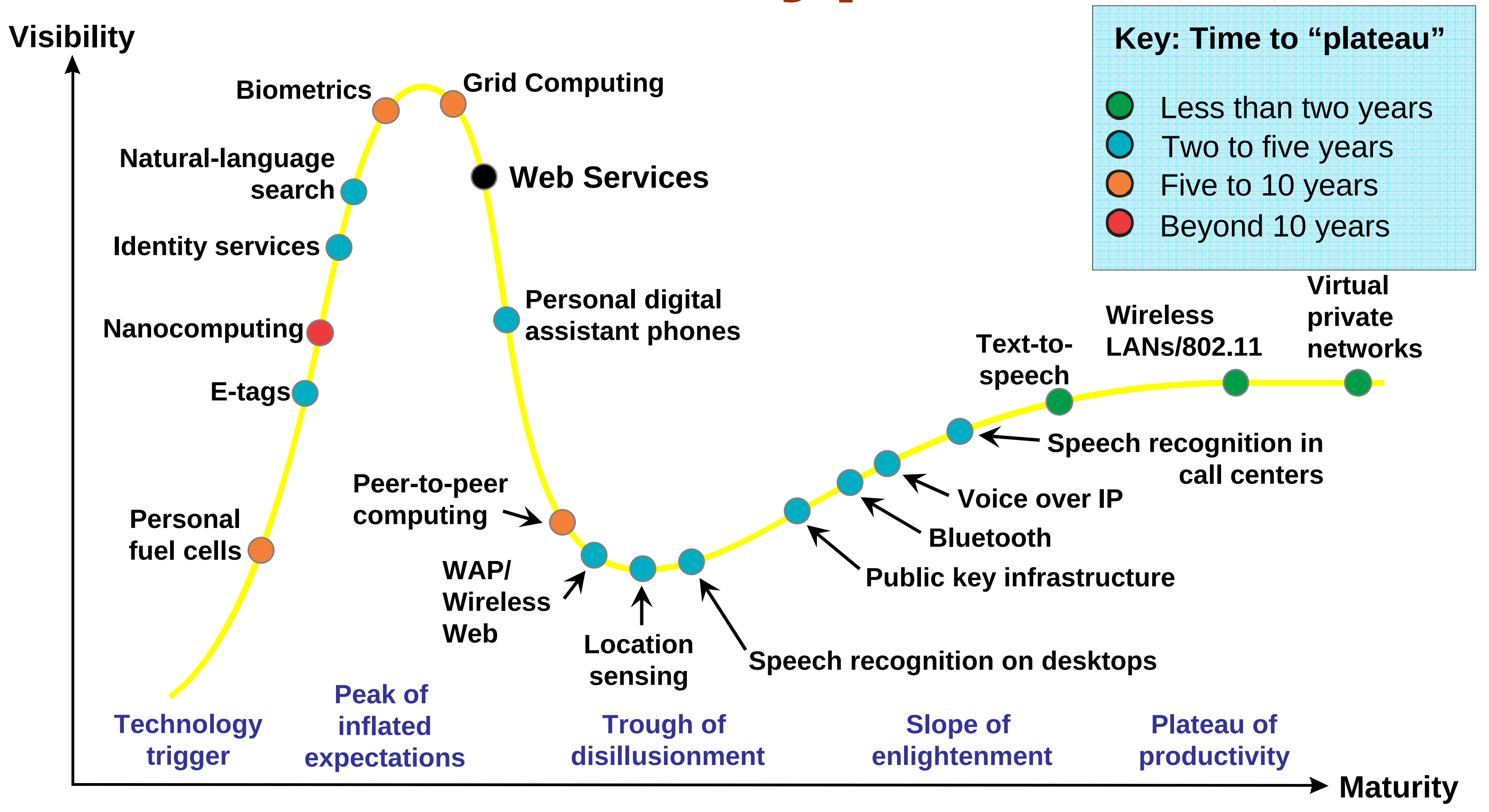
- VTA example:



- **Choreography =** how to interact with the service to consume its functionality
- **Orchestration =** how service functionality is achieved by aggregating other Web Services

Outlook

Gartner's 'Hype' Curve



Source: Gartner Group June 2002