95-702 Distributed Systems

Web Services
Chapter 9 of Coulouris

Web Services

- One of the dominant paradigms for programming distributed systems.
- Enables business to business integration. (Suppose one organization uses CORBA and another uses .NET) No problem!
- Enables service oriented architecture (SOA).
- Adopted by the grid computing community.
- May exist internally to an organization or externally (in the cloud).

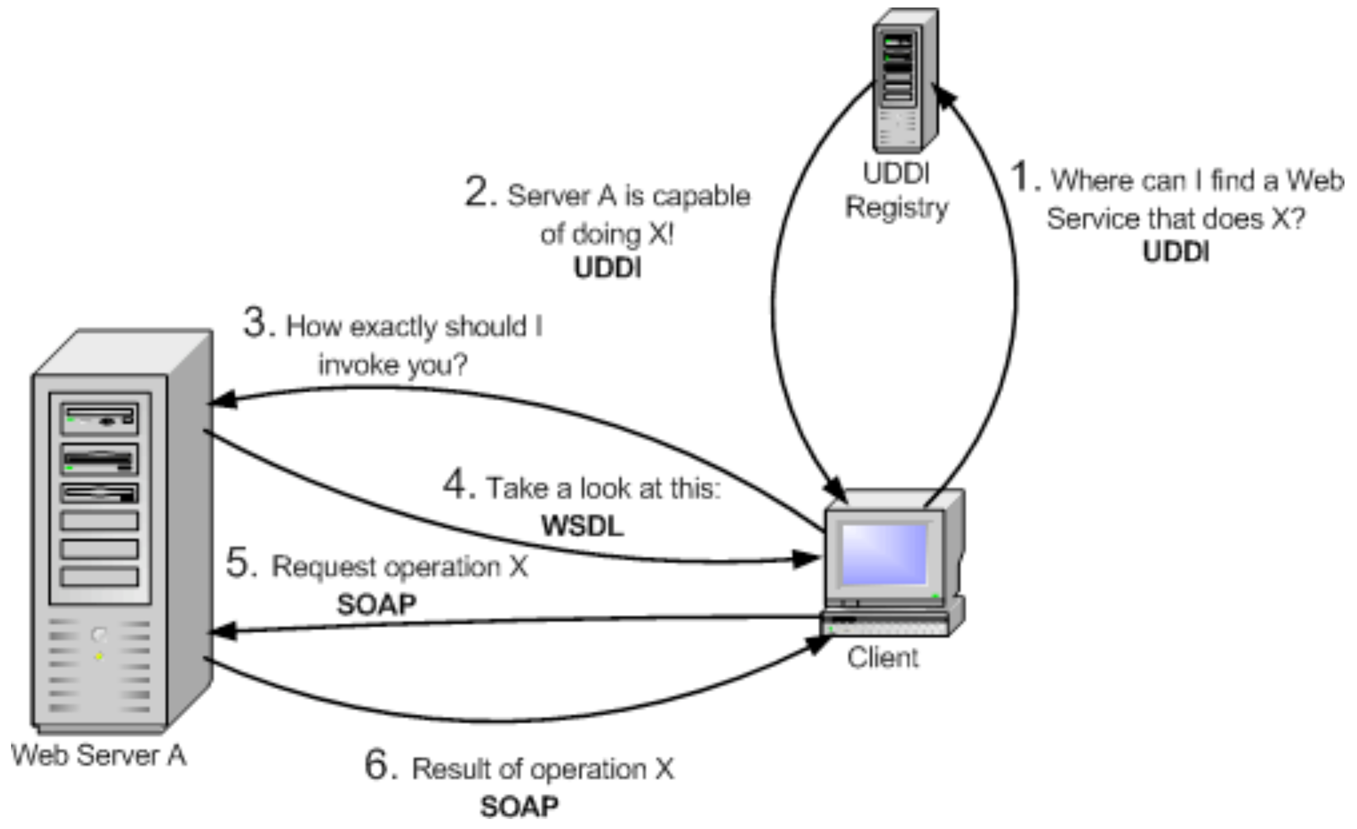
What are Web Services?

- Web Services began life when Bill Gates introduced BizTalk in 1999.
- BizTalk was later renamed .NET.
- The idea: "to provide the technologies to allow software in different places, written in different languages and resident on different platforms to connect and interoperate." From "Programming the World Wide" by Sebesta

Two Approaches

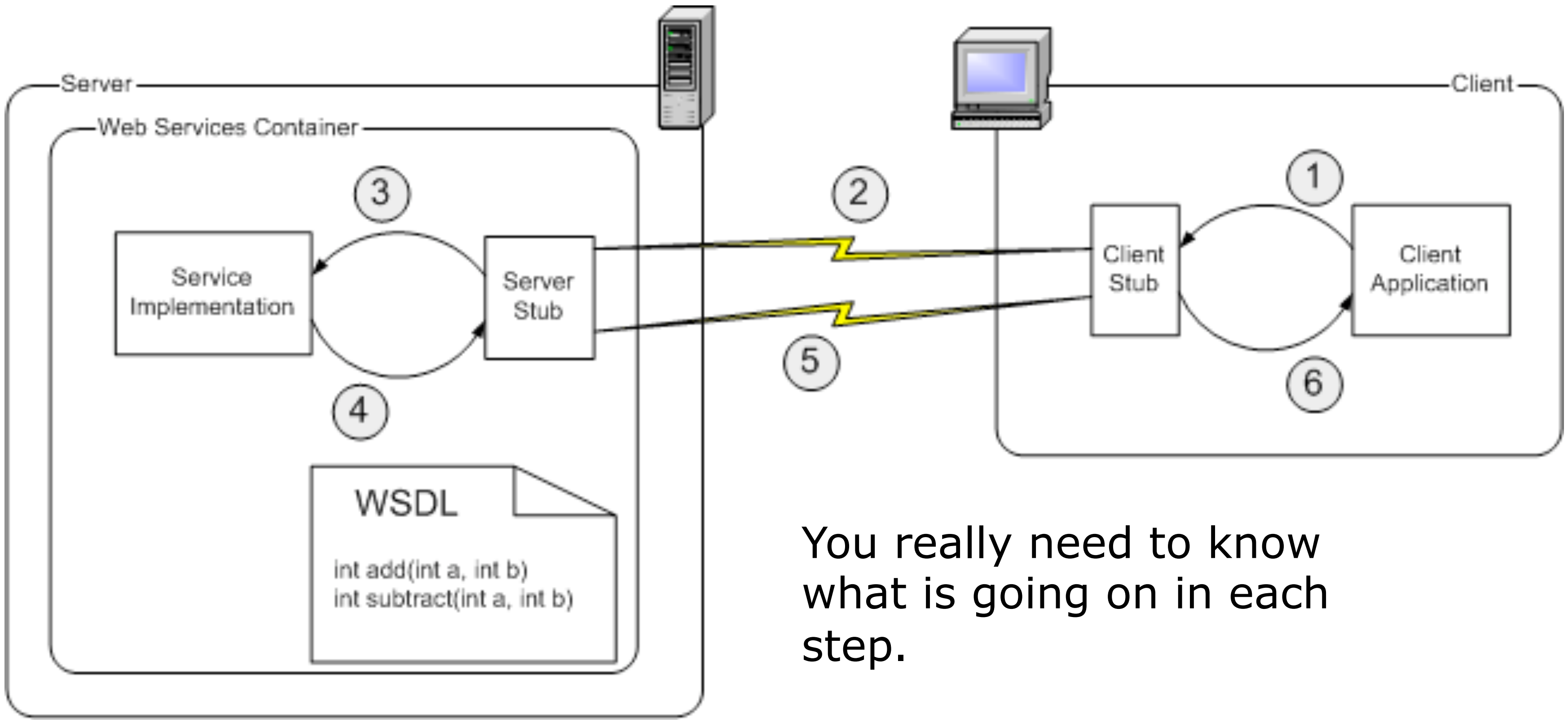
- SOAP Based (WS-*) Web Services
- REST style web services
- Today’s class:
  - Lecture and one short lab with a SOAP web service.
  - Principles of REST

A SOAP Based System In A Nutshell



From Globus.org
(Grid computing)

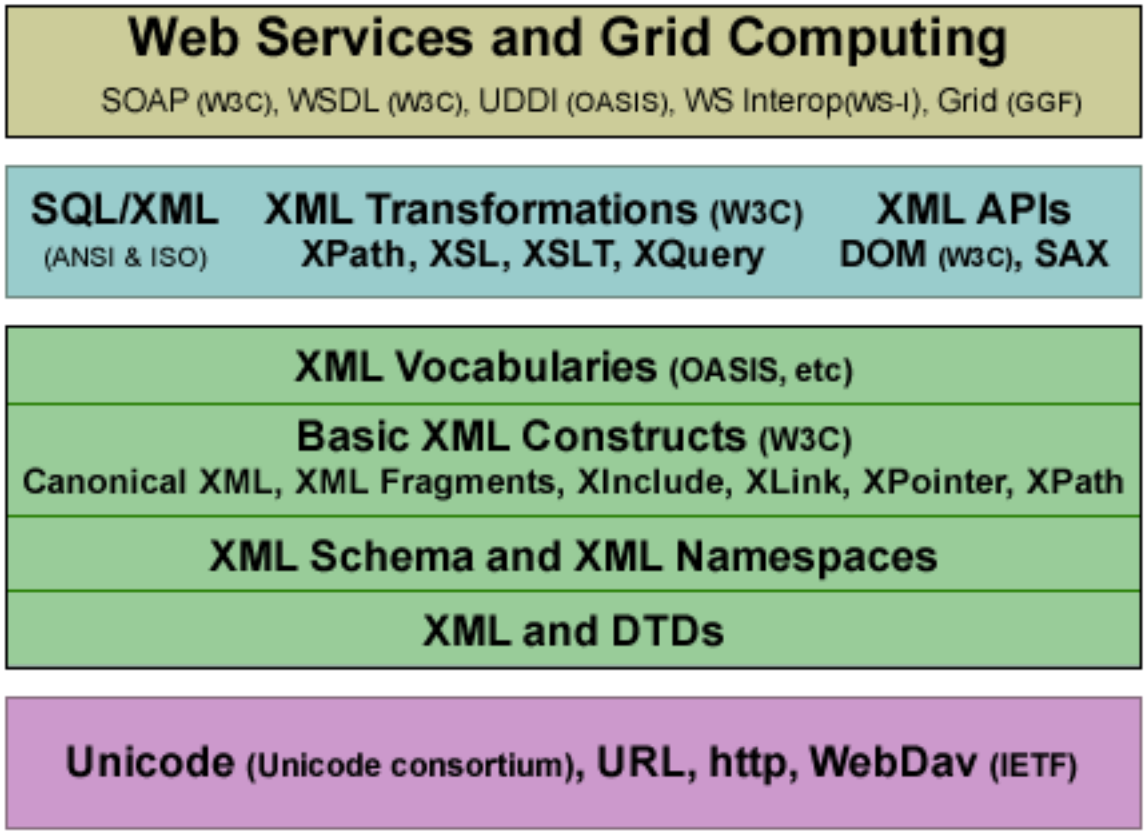
With Stubs



You really need to know what is going on in each step.

From Globus.org
(Grid computing)

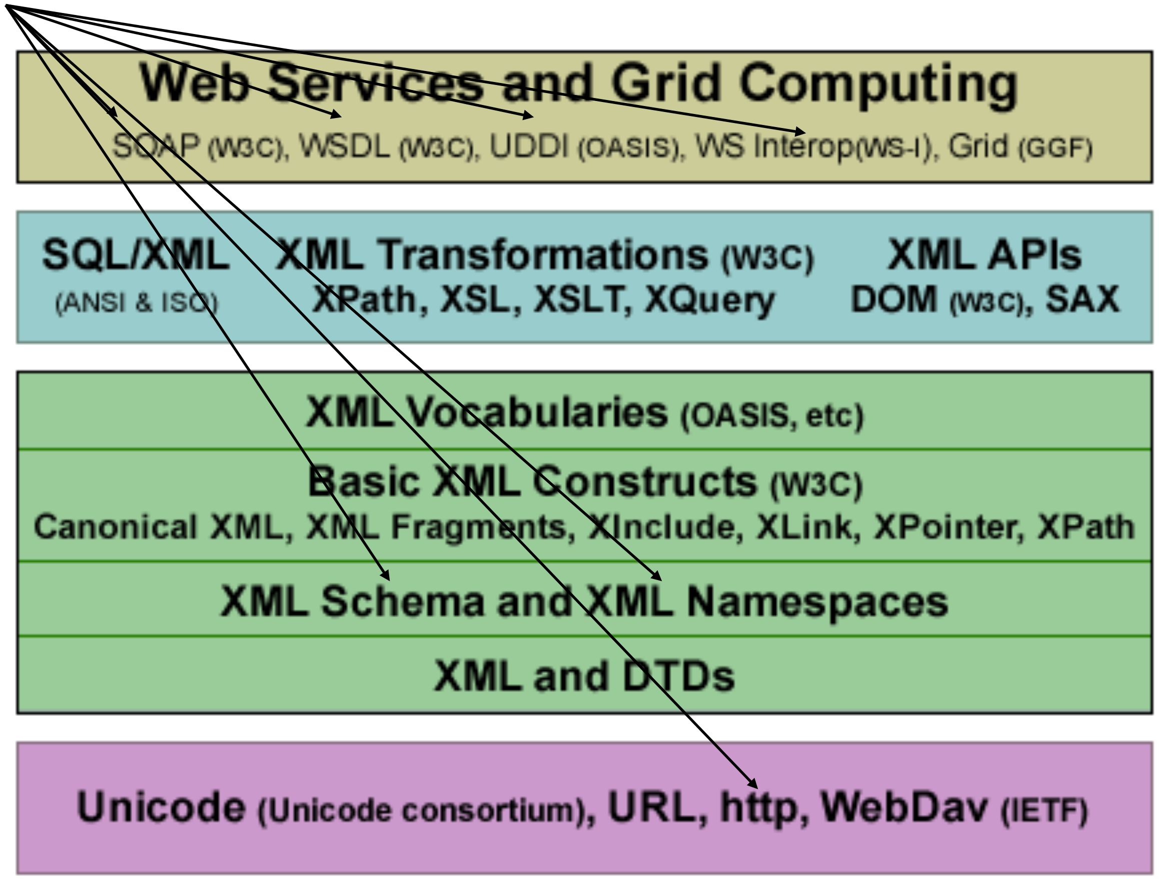
Some Important Standards



Web Services and Grid Computing
SOAP (W3C), WSDL (W3C), UDDI (OASIS), WS Interop(WS-I), Grid (GGF)
SQL/XML
(ANSI & ISO)
XML Transformations (W3C)
XPath, XSL, XSLT, XQuery
XML APIs
DOM (W3C), SAX
XML Vocabularies (OASIS, etc)
Basic XML Constructs (W3C)
Canonical XML, XML Fragments, XInclude, XLink, XPointer, XPath
XML Schema and XML Namespaces
XML and DTDs
Unicode (Unicode consortium), URL, http, WebDav (IETF)

Some Important Standards

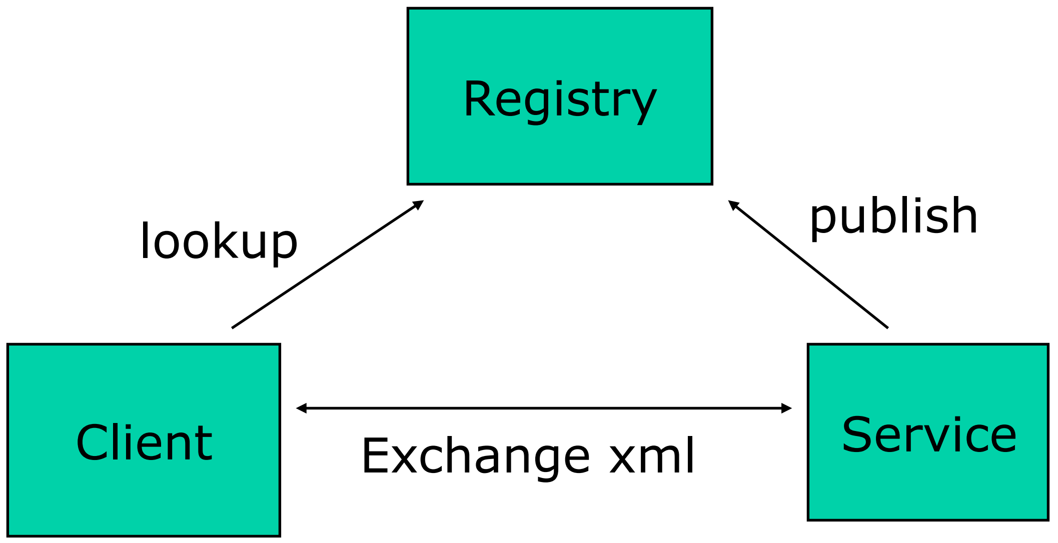
Very important with respect to XML web services.


Web Services and Grid Computing
SOAP (W3C), WSDL (W3C), UDDI (OASIS), WS Interop(WS-I), Grid (GGF)
SQL/XML
(ANSI & ISO)
XML Transformations (W3C)
XPath, XSL, XSLT, XQuery
XML APIs
DOM (W3C), SAX
XML Vocabularies (OASIS, etc)
Basic XML Constructs (W3C)
Canonical XML, XML Fragments, XInclude, XLink, XPointer, XPath
XML Schema and XML Namespaces
XML and DTDs
Unicode (Unicode consortium), URL, http, WebDav (IETF)

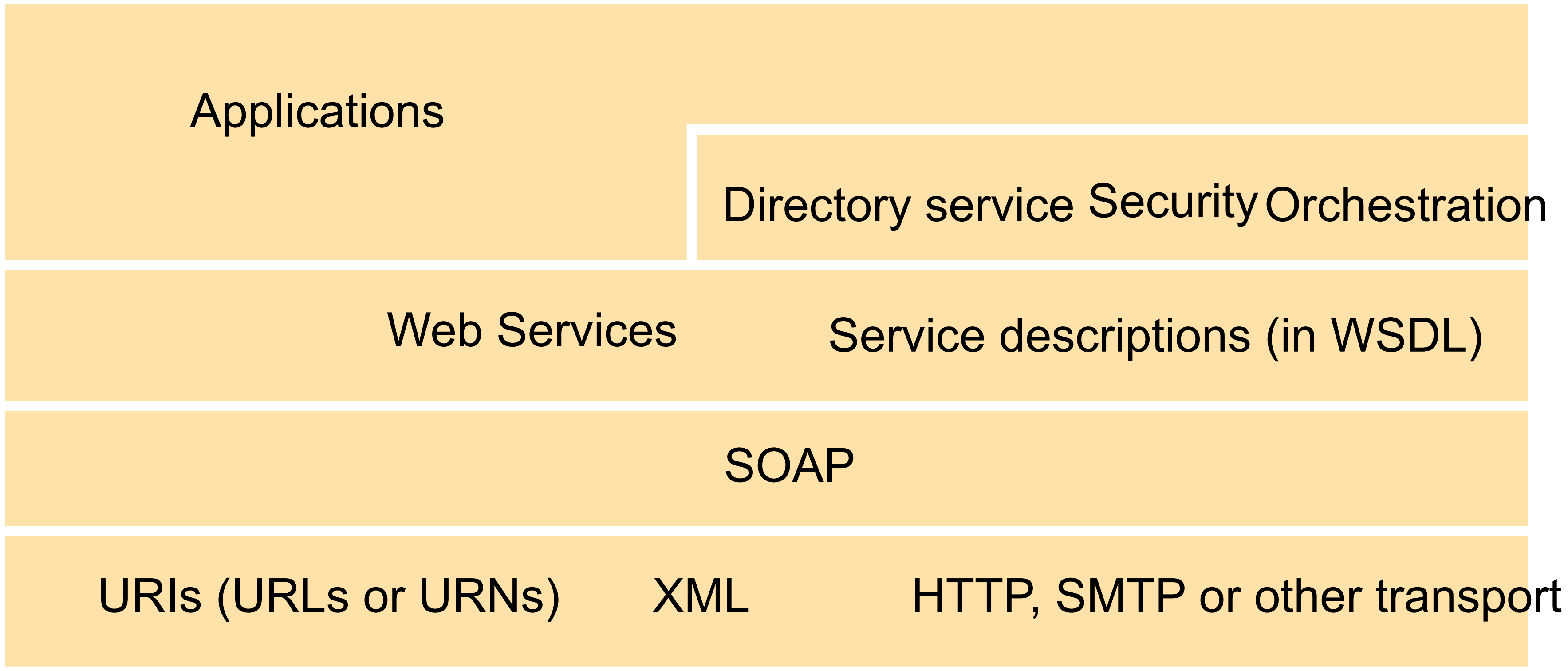
SOAP Style Web Services

- Provide service interfaces.
- Communicate using request and reply messages made of SOAP or some other XML document.
- Have an Interface Definition Language (IDL) called WSDL (Web Service Definition Language)
- May be looked up in a web service UDDI registry (Universal Directory and Discovery Service).
- Are language independent.
- May be synchronous or asynchronous.

Web Services



Web Services Infrastructure and Components



Communication Patterns

- In general, web services use either a **synchronous request-reply** pattern of communication with their clients or they communicate by **asynchronous messages**.
- The client does not block on asynchronous calls. Do you block when you are expecting an important phone call? If not then you are planning on handling the call asynchronously.
- To allow for a variety of patterns, SOAP is based on the packaging of single one-way messages.
- SOAP is used to hold RPC style parameters or entire documents.
- SOAP may be used over different transports (SMTP, TCP, UDP, or HTTP)

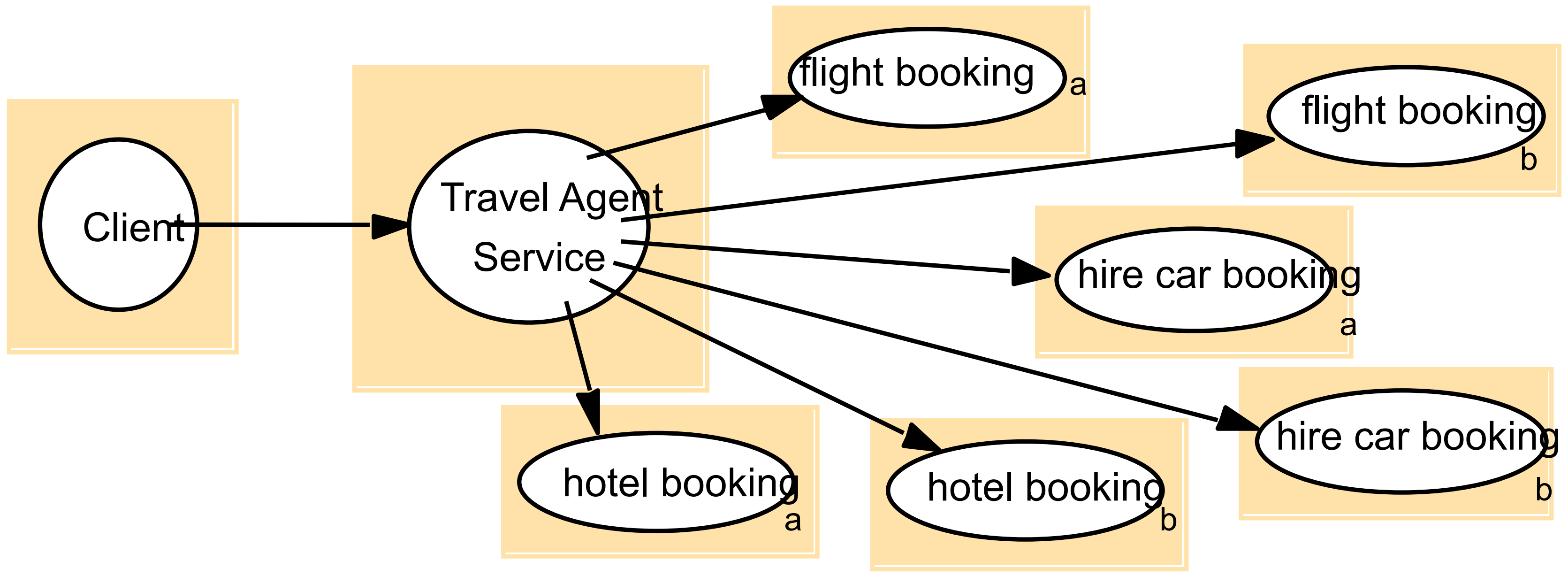
Service References

- URI's are **Uniform Resource Identifiers**.
- URL's are **Uniform Resource Locator** URI's that include location information. Thus, resources pointed to by URL's are hard to move.
- URN's are **Uniform Resource Name** URI's that include no location information.
- A URN lookup service may be used to determine a URL from a URN.
- URL's are the most frequently used form of URI.

Examples (the third is from Wikipedia):

1. URL: http://www.cmu.edu/service
2. URN: urn:ISBN:0-111-2345-6
3. "you can find urn:ietf:rfc:3187 (URN) over at http://tools.ietf.org/html/rfc3187.html (URL)."

Web Service Composition (Mashups)

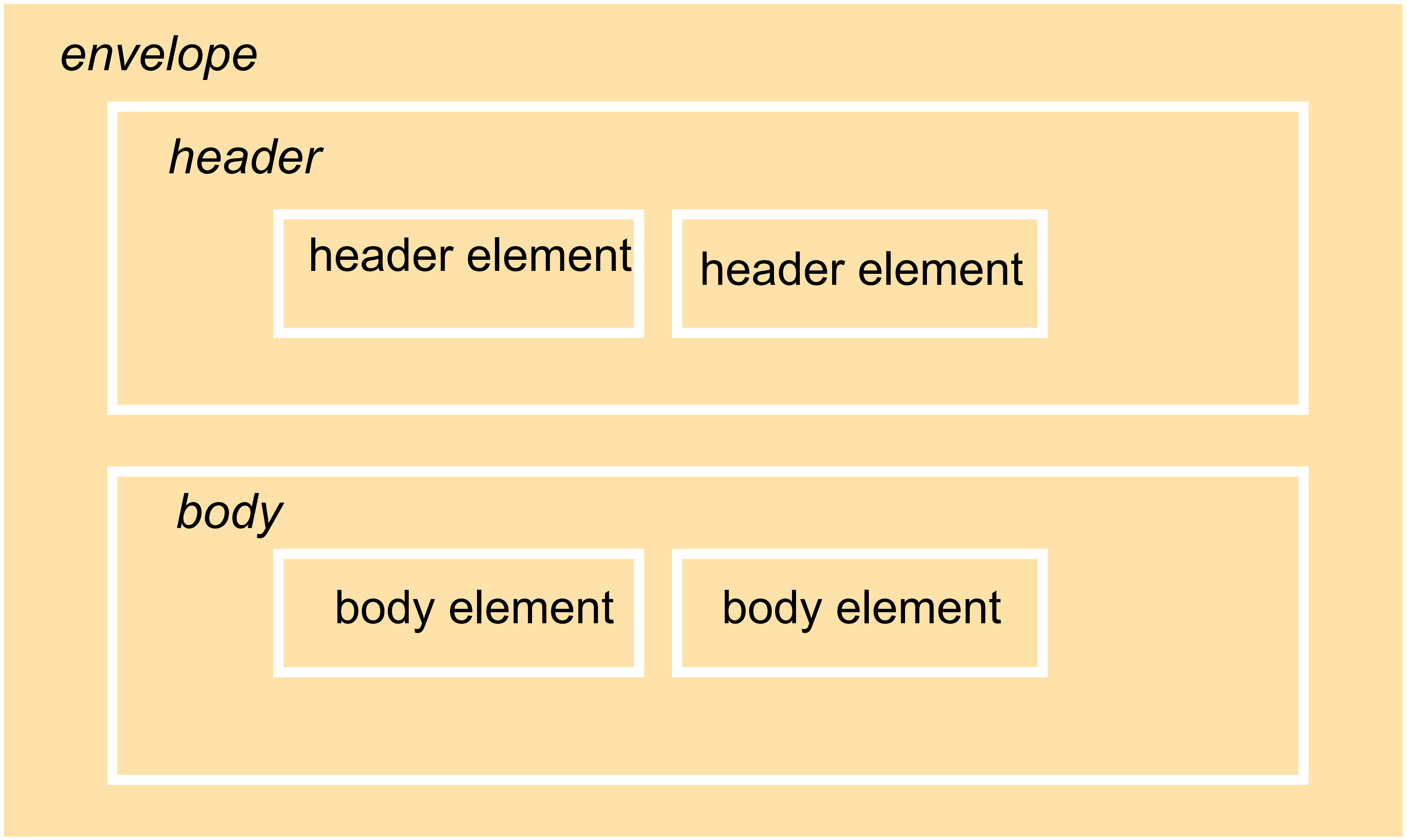


What concerns are not shown? This is an important list: Transactions, Security (privacy, identification, authentication, authorization), Reliability, Orchestration tooling, Interoperability through Standards, RPC or Messaging, Service Level agreements

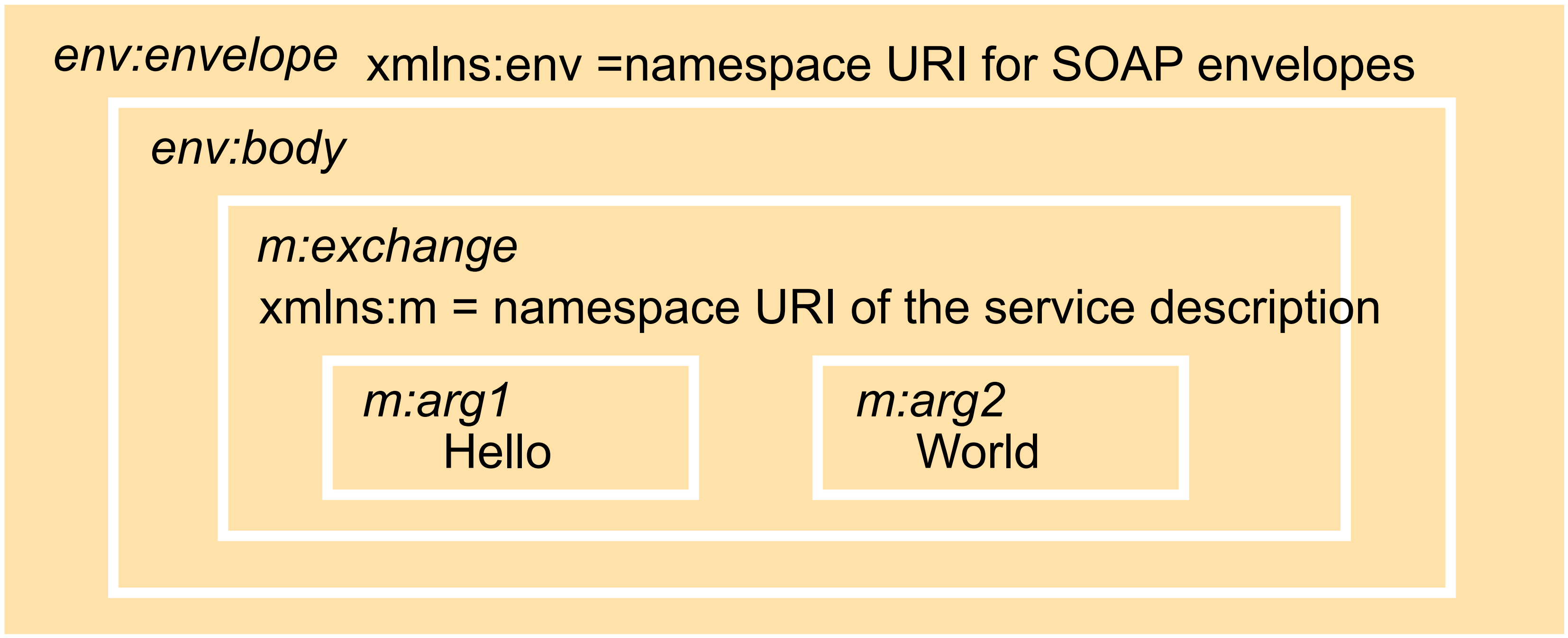
SOAP

- Defines a scheme for using XML to represent the contents of request and reply messages as well as a scheme for the communication of XML documents.
- It is intended that a SOAP message can be passed via intermediaries on the way to the computer that manages the resources to be accessed.
- The intermediaries may process the SOAP to provide security or transaction support as well as other services.
- Typically, the SOAP header is processed by intermediaries and the SOAP body holds the request or reply.

SOAP Envelope

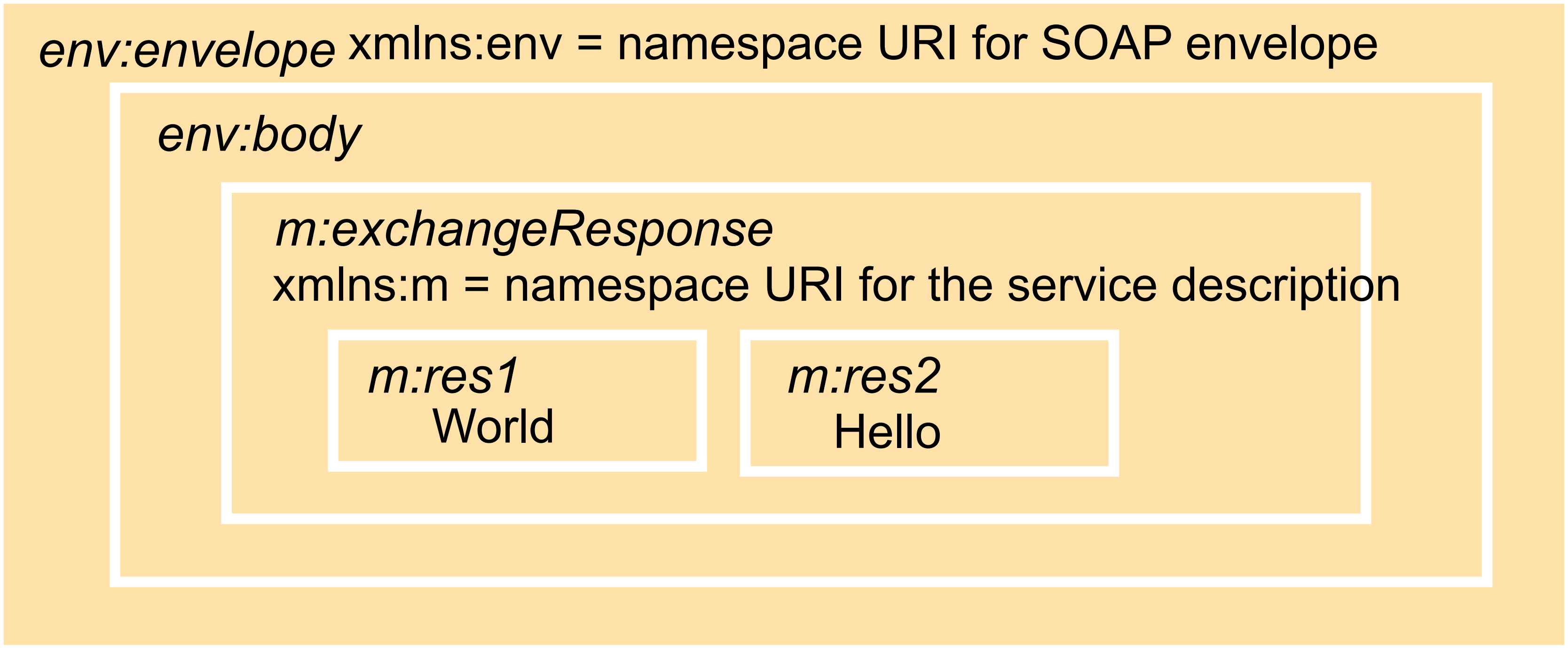


Request Without Headers

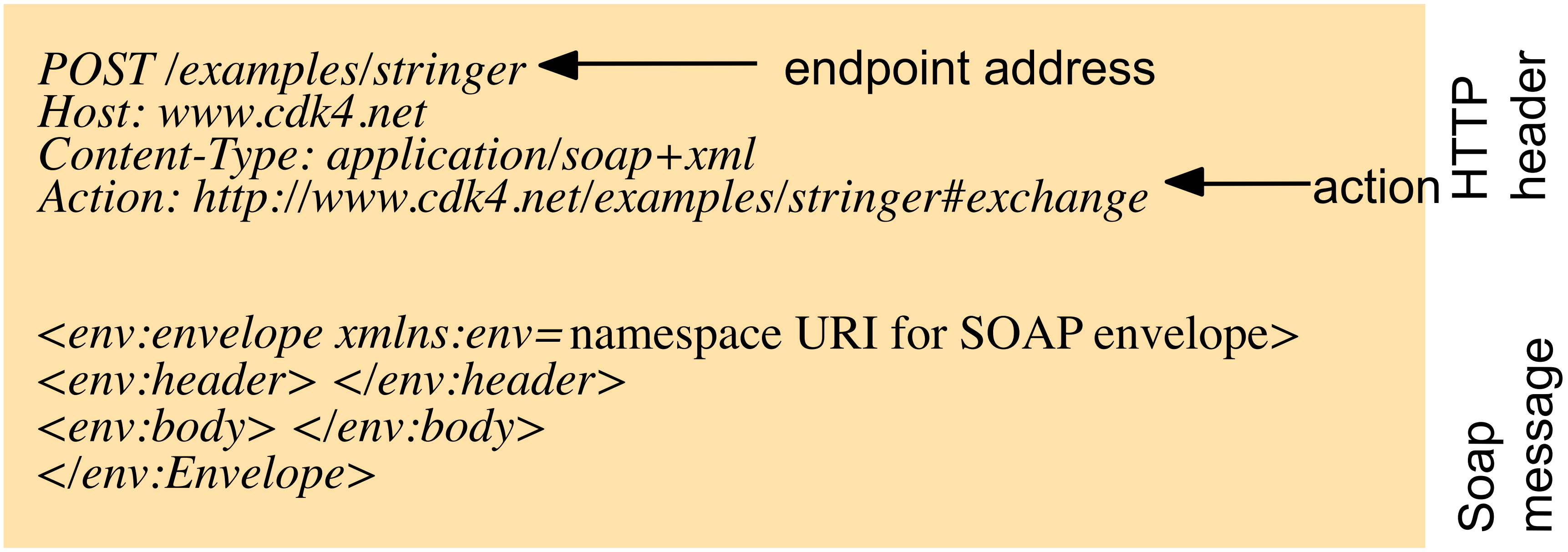

env:envelope xmlns:env =namespace URI for SOAP envelopes
env:body
m:exchange
xmlns:m = namespace URI of the service description
m:arg1
Hello
m:arg2
World

Why is envelope in a different namespace than arg1?
Consider GreenBayPackers:MikeMcCarthy and
CMU:MikeMcCarthy

Corresponding Reply


env:envelope xmlns:env = namespace URI for SOAP envelope
env:body
m:exchangeResponse
xmlns:m = namespace URI for the service description
m:res1
World
m:res2
Hello

HTTP POST Example



```
POST /examples/stringer
Host: www.cdk4.net
Content-Type: application/soap+xml
Action: http://www.cdk4.net/examples/stringer#exchange


<env:envelope xmlns:env=namespace URI for SOAP envelope>
<env:header> </env:header>
<env:body> </env:body>
</env:Envelope>
```

A transport protocol is required to send a SOAP document to its destination.

Other transports may be used. WS-Addressing may be used to include destination and source. Thus, different protocols might be used over different parts of the route of a message.

REST Style WS

POST /examples/stringer
Host: www.cdk4.net

$\longleftarrow$ Use a URI and an HTTP method to select what needs to be done.

Drop the SOAP and use name value pairs in the request. Use XML or JSON for the response. Don't provide a new set of operations - use HTTP methods instead. Use the same set of principles that made the web go viral!

WS-Addressing

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
           xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
    <S:Header>
      <wsa:MessageID>
         uuid:6B29FC40-CA47-1067-B31D-00DD010662DA
      </wsa:MessageID>
      <wsa:ReplyTo>
          <wsa:Address>http://business456.example/client1</wsa:Address>
       </wsa:ReplyTo>
       <wsa:To>http://fabrikam123.example/Purchasing</wsa:To>
       <wsa:Action>http://fabrikam123.example/SubmitPO</wsa:Action>
    </S:Header>
  <S:Body>
   ...
  </S:Body>
</S:Envelope>
```

Address information included within the document rather than only being specified by the transport. What does this buy us?

Distributed Objects?

At first glance, the interaction between client and server seems like RMI. We will look at RMI soon. RMI short course : OOP on more than one machine.

But, RMI permits the creation of **remote objects**. These may then be accessed via remote references.

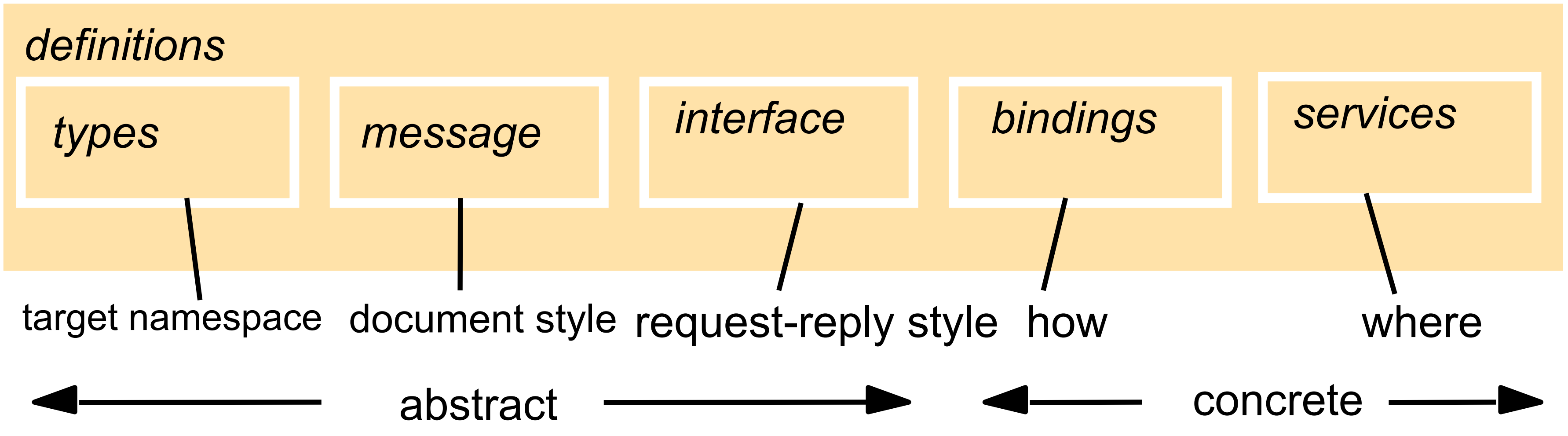
Web services may create and use objects but never return a remote reference to a remote object. **A web service is a single set of procedures**.

Web services are simpler than distributed objects. Simple is good.

Service Descriptions

- The primary means of describing a web service is by using WSDL (the Web Services Description Language).
- XML Schema may be used to describe the input and output parameters.
- WSDL describes the operations and makes use of XML Schema to describe an exchange of messages.
- A Service Description (WSDL document) is an IDL (interface definition language) plus it contains information on how and where the service may be accessed.
- It contains an abstract part and a concrete part. The abstract part is most like a traditional interface. The concrete part tells us how and where to access the service.

The Main Elements in a WSDL Description



A binding is a choice of protocols.
A service holds an endpoint address.
Client or server side code may be generated automatically from the WSDL.
A WSDL document may be accessed directly or indirectly through a registry like UDDI (Universal Directory and Discovery Service).

WSDL

- A message exchange is called an *operation.*
- Related operations are grouped into *interfaces.*
- A *binding* specifies concrete details about what goes on the wire.
- WSDL is an Interface Definition Language (IDL).

WSDL

- Describes the contract between applications.
- Can be automatically generated from a collection of Java or C# classes.
- Can be read by utilities that generate client side proxy code or server side skeletons.
- See wsimport (JDK 6.0) or wsdl.exe on the Microsoft side. In Netbeans just drag and drop a web reference.

Lab Activity One

- Write a SOAP based web service that computes the sum of two integers.
- Write a web service client that asks a user for two integers and makes use of the service to compute the sum.

Lab Activity (1)

(1)New Project/Java Web/Web Application/Name it
ArithmeticServiceProject/Next/Finish

(2) Right Click Project/New Web Service
Name: ArithmeticService
Package: edu.cmu.andrew.YOURID
Select: Create web service from scratch/Finish

(3) Use design mode to create a new operation called add that takes two integer arguments, adds them and returns an integer result.

(4) Use source mode to complete the code in the add method.

Lab Activity (2)

(5) Right click the project. Save, build and deploy the web service.

(6) Expand the Web Service node. Right click ArithmeticService and select Test Web Service.

(7) Use the browser to visit the service and view the SOAP request and response.

(8) View the WSDL. Copy the WSDL URL. We will need it to generate the client side proxy.

(9) Build a new project. This project will hold the web service client.

(10) New Project/Java Application/Name it WSClientProject.

(11) Right Click Project and select New Web Service Client.

(12) Select WSDL URL and paste the WSDL's URL.

Lab Activity (3)

(13) There should now be a web service reference node under your client project. Expand it down to the add method. Drag the add method into your client.

(14) Complete your client side code:

```
public static void main(String[] args) {
    System.out.println("Enter two ints to be added separated by a newline");
    Scanner sc=new Scanner(System.in);
    String x =sc.nextLine();
    String y = sc.nextLine();
    int a = Integer.parseInt(x);
    int b = Integer.parseInt(y);
    System.out.println(a + "+" + b);
    int c = add(a,b);                       // this add method was generated from
                                            // the WSDL
    System.out.print(c);
 }
```

Lab Activity (4)

(15) Right click the client project. Select clean and build and then run.

(16) Enter an int and hit return. Enter a second int and hit return.

Quiz questions:

a) How many SOAP messages were transferred on the wire?

b) What use was the WSDL?

c) Is this an interoperable solution?

d) Is this secure from Even and Mallory?

e) Is this synchronous or asynchronous?

f) Would this service handle concurrent visitors?

A Look At REST

- So far we have:

  (1) Created a SOAP based web service.
  (2) Tested it and retrieved its WSDL.
  (3) Generated code based on the WSDL.
  (4) Called that code from a client.

- Let’s look at the REST design philosophy…

REST

- REpresentational State Transfer
- Roy Fielding's doctoral dissertation (2000)
- Fielding (along with Tim Berners-Lee) designed HTTP and URI's.
- The question he tried to answer in his thesis was "Why is the web so viral"? What is its architecture? What are its principles?

Notes from "RESTFul Java with JAX-RS" by Bill Burke.

REST Architectural Principles

- The web has addressable resources.
  Each resource has a URI.
- The web has a uniform and constrained interface.
  HTTP, for example, has a small number of methods. Use these to manipulate resourses.
- The web is representation oriented – providing diverse formats.
- The web may be used to communicate statelessly – providing scalability
- Hypermedia is used as the engine of application state.

Understanding REST

- REST is not protocol specific. It is usually associated with HTTP but its principles are more general.
- SOAP and WS-* use HTTP strictly as a transport protocol.
- But HTTP may be used as a rich application protocol.
- Browsers usually use only a small part of HTTP.
- HTTP is a synchronous request/response network protocol used for distributed, collaborative, document based systems.
- Various message formats may be used – XML, JSON,..
- Binary data may be included in the message body.

Principle: Addressability

- Addressability (not restricted to HTTP)
  Each HTTP request uses a URI.
  The format of a URI is well defined:

  scheme://host:port/path?queryString#fragment

  The scheme need not be HTTP. May be FTP or HTTPS.
  The host field may be a DNS name or a IP address.
  The port may be derived from the scheme. Using HTTP implies port 80.
  The path is a set of text segments delimited by the "/".
  The queryString is a list of parameters represented as name=value pairs. Each pair is delimited by an "&".
  The fragment is used to point to a particular place in a document.

  A space is represented with the '+' characters. Other characters use % followed by two hex digits.

Principle: Uniform Interface (1)

- A uniform constrained interface:

  - No action parameter in the URI

  - HTTP
    GET - read only operation
    - idempotent (once same as many)
    - safe (no important change to server's state)
    - may include parameters in the URI
      http://www.example.com/products?pid=123

Principle: Uniform Interface (2)

HTTP

PUT - store the message body
- insert or update
- idempotent
- not safe

Principle: Uniform Interface (3)

HTTP

POST - Not idempotent
- Not safe
- Each method call may modify the resource in a unique way
- The request may or may not contain additional information
- The response may or may not contain additional information
- The parameters are found within the request body (not within the URI)

Principle: Uniform Interface (4)

HTTP
DELETE - remove the resource
- idempotent
- Not safe
- Each method call may modify the resource in a unique way
- The request may or may not contain additional information
- The response may or may not contain additional information

HTTP HEAD, OPTIONS, TRACE and CONNECT are less important.

Principle: Uniform Interface (5)

Does HTTP have too few operations?

Note that SQL has only four operations: SELECT, INSERT, UPDATE and DELETE

JMS and MOM have, essentially, two operations: SEND and RECEIVE

SQL and JMS have been very useful.

What does a uniform interface buy?

Familiarity

- We do not need a general IDL that describes a variety of method signatures.
- We already know the methods.

Interoperability

- WS-* has been a moving target.
- HTTP is widely supported.

Scalability

- Since GET is idempotent and safe, results may be cached by clients or proxy servers.
- Since PUT and DELETE are both idempotent neither the client or the server need worry about handling duplicate message delivery.

Principle: Representation Oriented(1)

- Representations of resources are exchanged.
- GET returns a representation.
- PUT and POST passes representations to the server so that underlying resources may change.
- Representations may be in many formats: XML, JSON, YAML, etc., …

Principle: Representation Oriented(2)

- HTTP uses the CONTENT-TYPE header to specify the message format the server is sending.
- The value of the CONTENT-TYPE is a MIME typed string. Versioning information may be included.
- Examples:

    text/plain

    text/html

    application/vnd+xml;version=1.1
- “vnd” implies a vendor specific MIME type

Principle: Representation Oriented(3)

- The ACCEPT header in content negotiation.
- An AJAX request might include a request for JSON.
- A Java request might include a request for XML.
- Ruby might ask for YAML.

Principle: Communicate Statelessly

- The application may have state but there is no client session data stored on the server.
- If there is any session-specific data it should be held and maintained by the client and transferred to the server with each request as needed.
- The server is easier to scale. No replication of session data concerns.

Principle: HATEOAS

- **Hypermedia as the Engine of Application State**
- Hypermedia is document centric but with the additional feature of links.
- With each request returned from a server it tells you what interactions you can do next as well as where you can go to transition the state of your application.
- Example:

```
<order id = “111”>
    <customer>http://…/customers/3214
    <order-entries>
        <order-entry>
            <qty>5
            <product>http://…/products/111
```

Principle: HATEOS

- From Wikipedia:
- A REST client enters a REST application through a simple fixedURL. All future actions the client may take are discovered within resource representations returned from the server. The media types used for these representations, and the link relations they may contain, are standardized. The client transitions through application states by selecting from the links within a representation or by manipulating the representation in other ways afforded by its media type. In this way, RESTful interaction is driven by hypermedia, rather than out-of-band information.[1]

Principle: HATEOS

- Knock Knock Joke Example
- Netbeans RESTKnocker

Bing Maps Using REST

- Visit the URL (using my key):

http://dev.virtualearth.net/REST/v1/Locations/New%20York?
output=xml&key=AqMWEeRufD1Ch2uhYsyD10OPbLGMs_GATcB8Xd8trcvybpNuDRcMo6U6uVCqOa
ra

- How does your browser react?
- Why?
- Change XML to JSON.
- How does your browser react?
- Why?

WeatherBug API

- Here is a nice description of the weather bug REST style API:
- http://developer.weatherbug.com/docs/read/WeatherBug_API_JSON
- WeatherBug also provides a SOAP based service.