eGee

Enabling Grids for E-science in Europe

www.eu-egee.org

Introduction to Web Services

Objectives

- Architecture

- Standards
  - XML Schema
  - SOAP
  - WSDL
  - UDDI

- Context for Web Services

Section

INTRODUCTION

The concept of web services

- Web services is a messaging system which allows communication between objects.

- Messages can be synchronous or asynchronous.

- This system is loosely coupled
  (ie. Services should not be dependent on each other).

W3C view of Web Services

- *The World Wide Web is more and more used for application to application communication.*

- *The programmatic interfaces made available are referred to as **Web services**.*
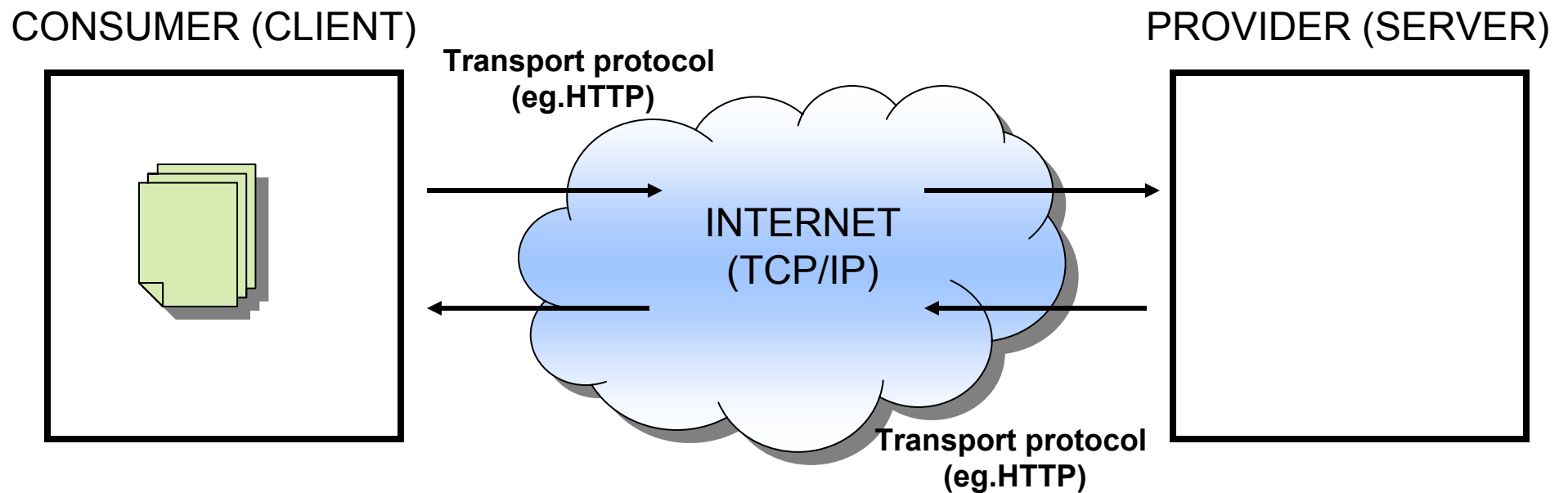
- http://www.w3.org/2002/ws/

Web Services

- Web services are
  - Applications that enable remote procedure calls over a network or the Internet often using XML and HTTP

- Benefits
  - This allows us to hide the details of how a service is implemented; only URL and data types are required
  - It is largely irrelevant to the client whether the service is developed with Java or ASP.NET or if it is running on Windows, Linux or any other platform
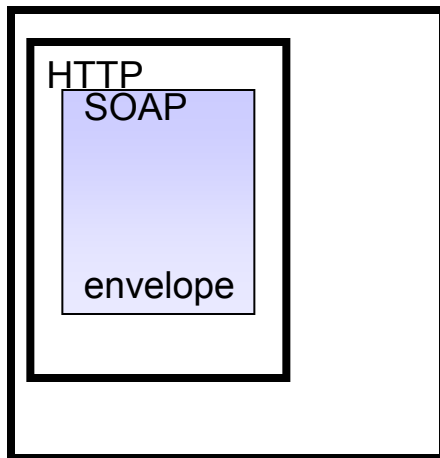
W3C Web Services glossary

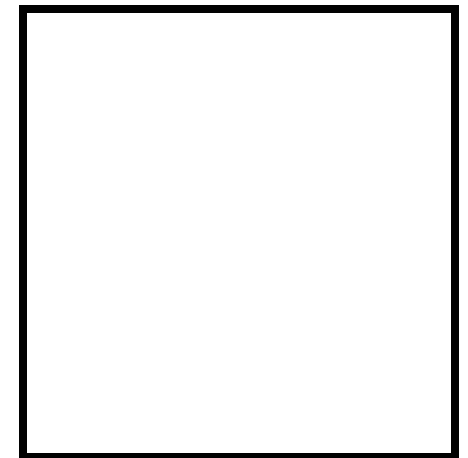- http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

Architecture

Web services architecture overview

Consumer (1)

CONSUMER (CLIENT)

Transport protocol (eg.HTTP)

INTERNET (TCP/IP)

PROVIDER (SERVER)

Transport protocol (eg.HTTP)

eGee
Enabling Grids for
E-science in Europe
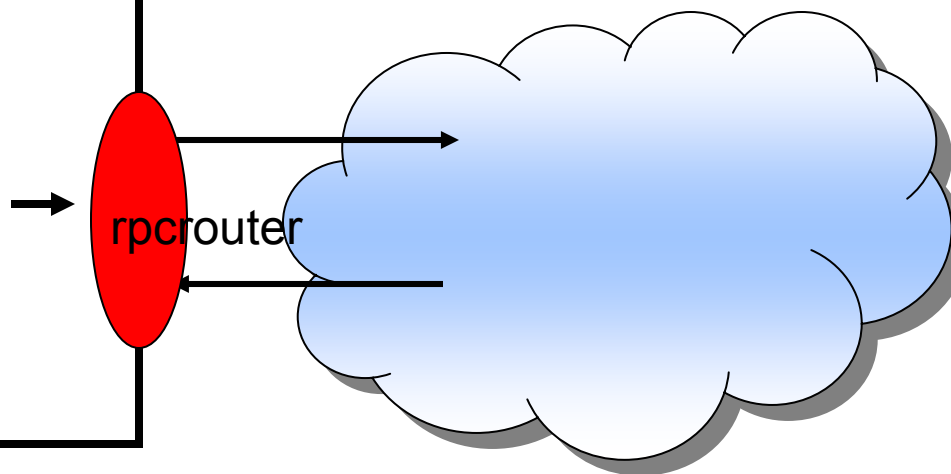
Consumer (2)

CONSUMER (CLIENT)

PROVIDER (SERVER)

HTTP
SOAP

envelope

Consumer (3)

CONSUMER (CLIENT)

PROVIDER (SERVER)

HTTP

SOAP

rpcrouter

envelope
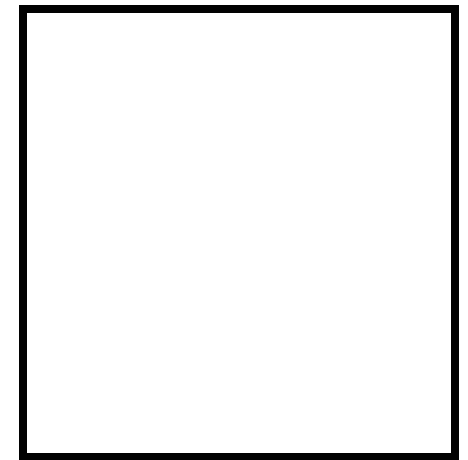
Web services architecture overview Provider (1)

eGee
Enabling Grids for
E-science in Europe

CONSUMER (CLIENT)

PROVIDER (SERVER)

Container
(eg. Tomcat)

Servlet

Web services architecture overview Provider (2)

CONSUMER (CLIENT)
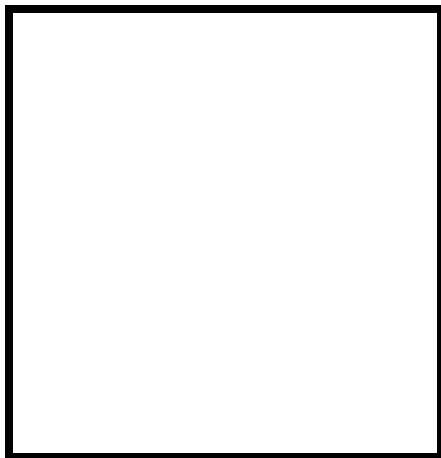
PROVIDER (SERVER)

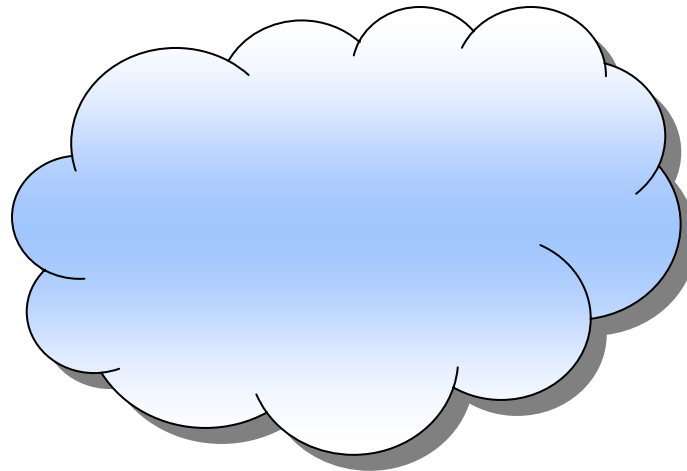Web server
(eg. Apache)

Container
(eg. Tomcat)

Servlet
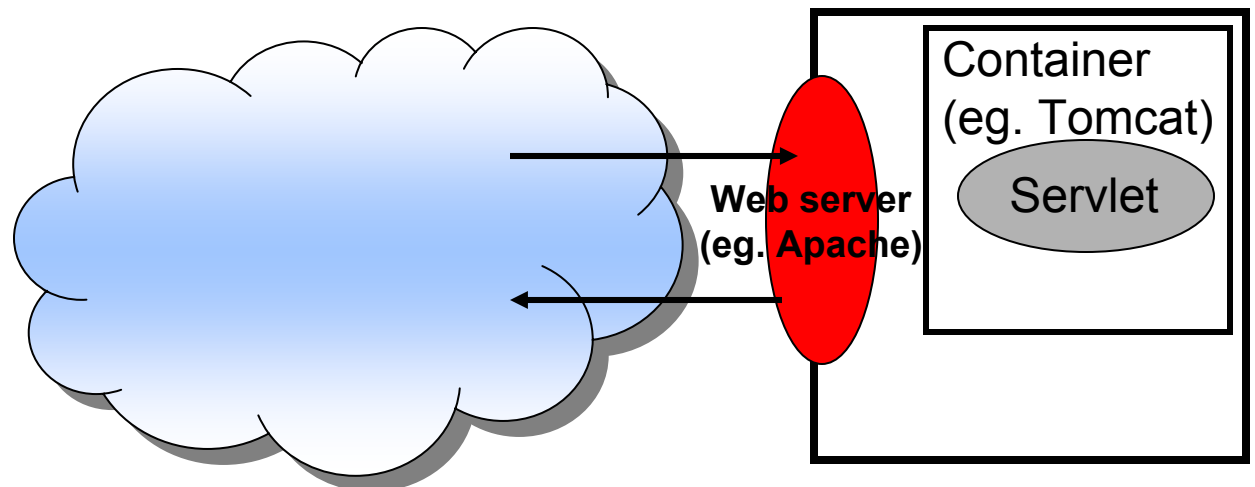
Web services architecture overview
Message transport (1)

CONSUMER (CLIENT)

PROVIDER (SERVER)

HTTP

SOAP

envelope

Container
(eg. Tomcat)

Servlet

Web services stack
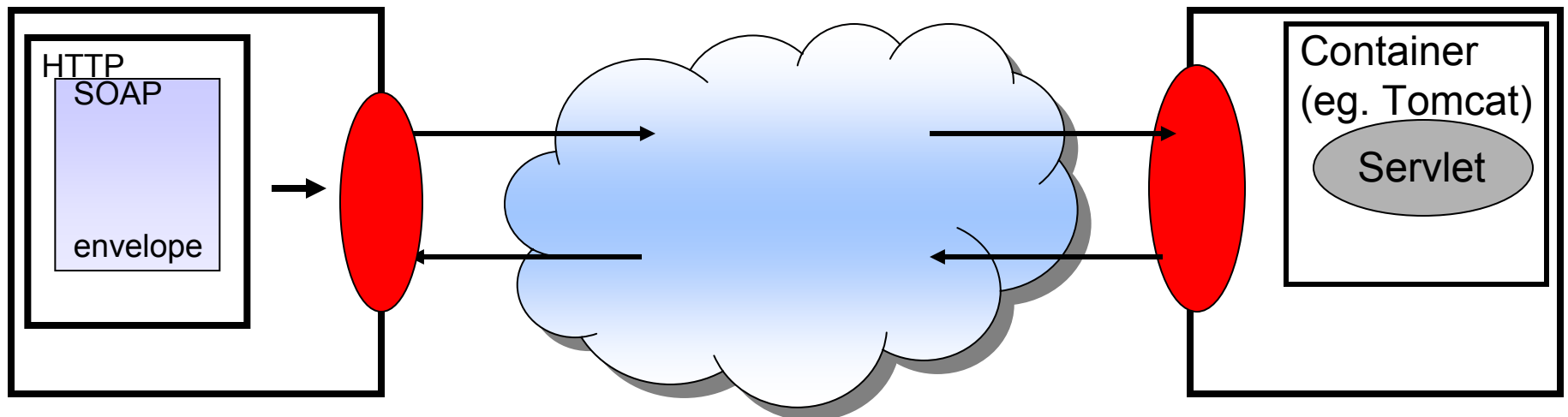
Application	Application (servlet)
rpcrouter	Web server
SOAP	SOAP
HTTP	HTTP
TCP/IP	TCP/IP
Infrastructure (Data link)	Infrastructure (Data link)
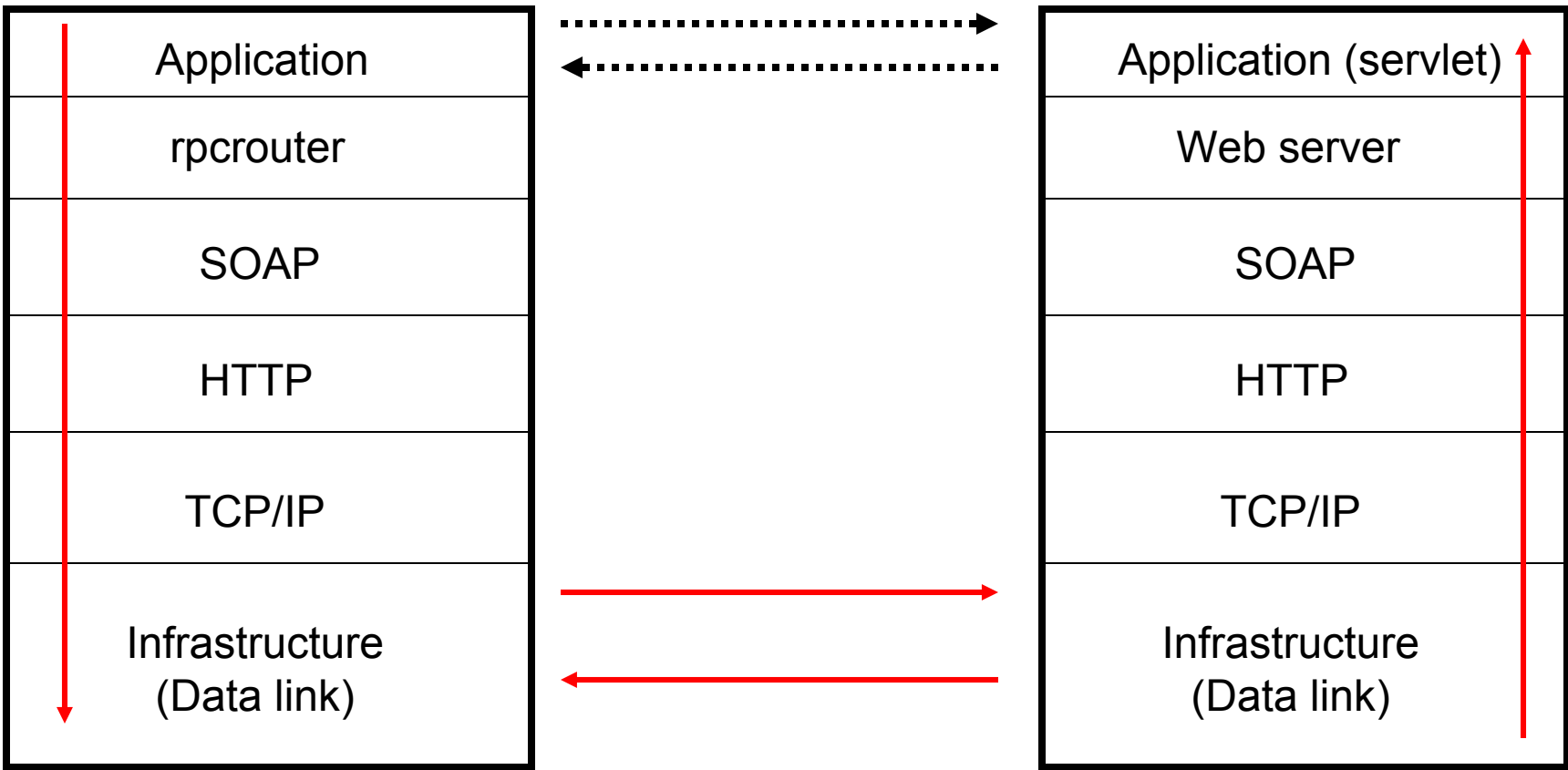
Business mail analogy

- The SOAP envelope is analogous to a business letter with an address **within** a distant organisation. This gives the information needed to get it from the sender's building to the recipient's building.

- The transport protocol is analogous to the carrier used for transport between buildings. (eg. FedEx.)

- The web server and container act like the local services for the recipient which place the message in his/her pigeon-hole.
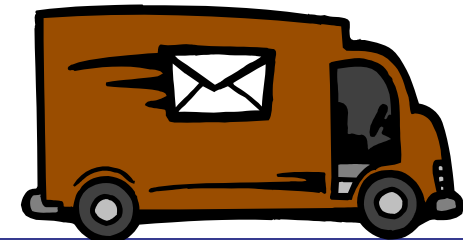
Protocols

Communication and standards

- Efficient (or indeed any) communication is dependent on a shared vocabulary and grammar.

- Because web services deals with inter-organisation communication these must be universal standards.

Underlying standards

- The basic standards for web services are:

- XML (Extensible Markup Language)

- SOAP (simple object access protocol)

- WSDL (web services description language)

- UDDI (universal description, discovery and integration)

The state of standards

- XML 1.0 fairly stable, although Schema are in the process of replacing DTDs (currently Schema 1.1 being worked on).
- SOAP 1.2
- WSDL 2.0 (coming out, 1.2 current)
- UDDI version 3 (Aug 2003)
- BPEL 1.1 (Business Process Execution Language)
- choreography description language (web services work flows)

    started January 2003.

Standards are still volatile and in the process of development.

Web Services Architecture

- Web Services involve three major roles
  - Service Provider
  - Service Registry
  - Service Consumer

- Three major operations surround web services
  - Publishing – making a service available
  - Finding – locating web services
  - Binding – using web services

Making a service available (1)

- In order for someone to use your service they have to know about it.

- To allow users to discover a service it is published to a registry (UDDI).

- To allow users to interact with a service you must publish a description of it's interface (methods & arguments).

- This is done using WSDL.

Making a service available (2)

- Once you have published a description of your service you must have a host set up to serve it.

- A web server is often used to deliver services (although custom application – application communication is also possible).

- This is functionality which has to be added to the web server. In the case of the apache web server a 'container' application (Tomcat) can be used to make the application (servlet) available to apache (deploying).

The old transfer protocols are still there.

- Like the grid architecture web services is layered on top of existing, mature transfer protocols.

- HTTP, SMTP are still used over TCP/IP to pass the messages.

- Web services, like grids, can be seen as a functionality enhancement to the existing technologies.

XML

- All Web Services documents are written in XML

- XML Schema are used to define the elements used in Web Services communication

SOAP

- Actually used to communicate with the Web Service

- Both the request and the response are SOAP messages

- The body of the message (whose grammar is defined by the WSDL) is contained within a SOAP "envelope"

- "Binds" the client to the web service

WSDL

- Describes the Web Service and defines the functions that are exposed in the Web Service

- Defines the XML grammar to be used in the messages
  - Uses the W3C Schema language

UDDI

- UDDI is used to register and look up services with a central registry

- Service Providers can publish information about their business and the services that they offer

- Service consumers can look up services that are available by
  - Business
  - Service category
  - Specific service

XML

What is XML

- XML stands for extensible markup language

- It is a hierarchical data description language

- It is a sub set of SGML a general document markup language designed for the American millitary.

- It is defined by w3c.

How does XML differ from HTML?

- HTML is a presentation markup language – provides no information about content.

- There is only one standard definition of all of the tags used in HTML.

- XML can define both presentation style and give information about content.

- XML relies on custom documents defining the meaning of tags.

What is a Schema?

- A schema is the definition of the meaning of each of the tags within a XML document.

- *Analogy: A HTML style sheet can be seen as a limited schema which only specifies the presentational style of HTML which refers to it.*

- *Example: in HTML the tag <strong> pre-defined. In XML you would need to define this in the context of your document.*

A minimal XML document

value

```
<?xml version="1.0" ?>
    <document name="first">Jim</document>
```

A tag

An attribute

Closing tag
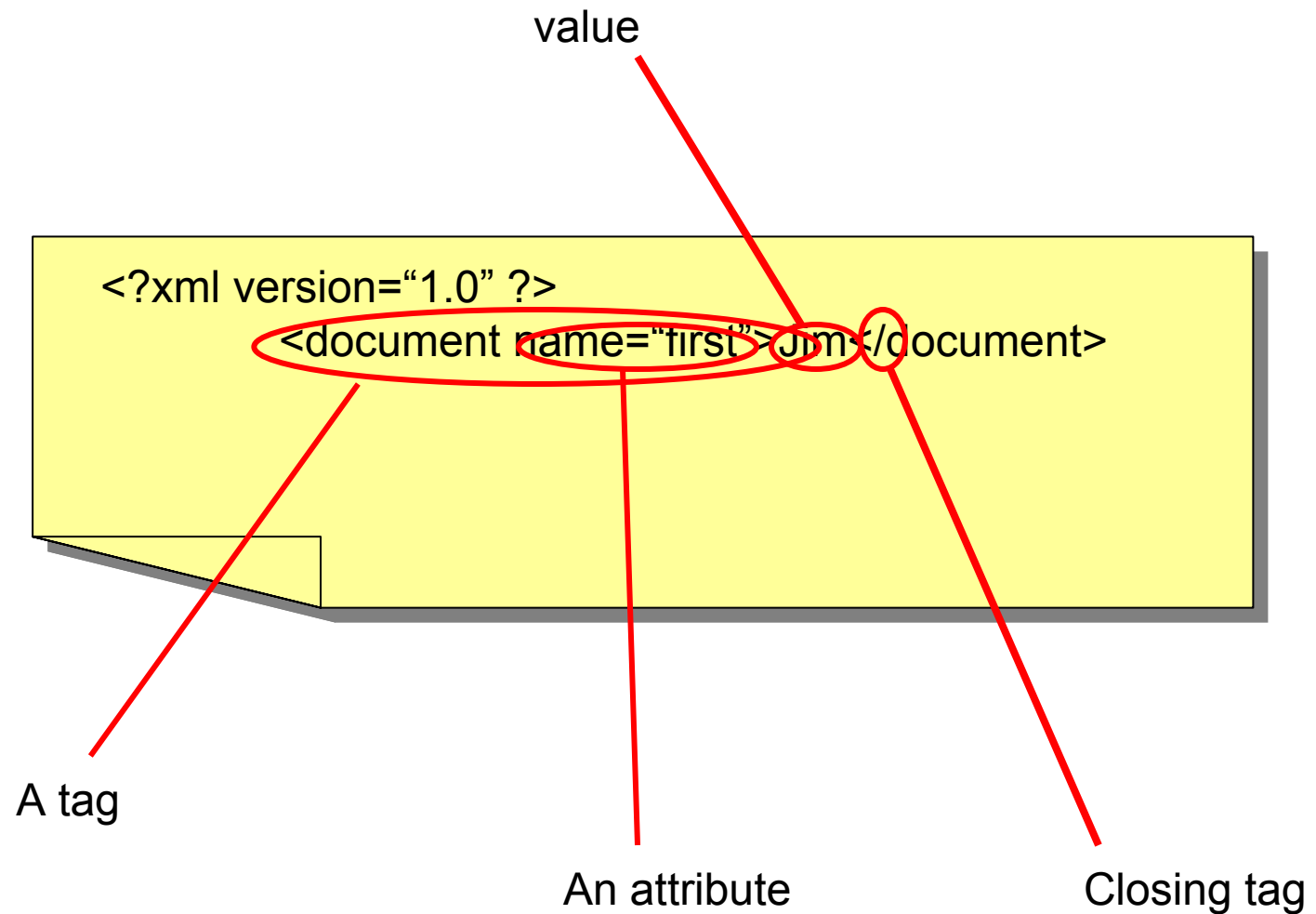
Valid and well formed

- A correct XML document must be both valid and well formed.

- Well formed means that the syntax must be correct and all tags must close correctly (eg <…> </…>).

- Valid means that the document must conform to some XML definition (a DTD or Schema).

(Otherwise there can be no definition of what the tags mean)

Using namespaces in XML

- To fully qualify a namespace in XML write the namespace:tag name. eg.

    <my_namespace:tag> </my_namespace:tag>

- In a globally declared single namespace the qualifier may be omitted.

- More than one namespace:

    <my_namespace:tag> </my_namespace:tag>

    <your_namespace:tag> </your_namespace:tag>

  can co-exist if correctly qualified.

Namespaces in programming languages

- In C/C++ defined by #includes and classes (eg. myclass::variable).

- In PERL defined by package namespace, $local and $my (eg. myPackage::variable).

- In JAVA defined by includes and package namespace (eg. java.lang.Object)

- **Defines the scope of variables**

Schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
xmlns="document" >
<xs:element name = "DOCUMENT">
            <xs:element name="CUSTOMER"> </xs:element>
</xs:element>
</xs:schema>
```

Simple schema
saved as order.xsd

```
<?xml version="1.0"?>
<DOCUMENT xmlns="document"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
Xsi:schemaLocation="order.xsd">
<DOCUMENT>
            <CUSTOMER>sam smith</CUSTOMER>
            <CUSTOMER>sam smith</CUSTOMER>
</DOCUMENT>
```

XML document
derived from
schema.

Section

SOAP

Request Response Web Services

- Currently the most common implementation of Web Services

- Work in a very simple 'request – response' paradigm

- For Example:
    - A Weather Service– simple request for weather in an area, simple response with the weather report
    - An Airline special offers service – travel agents would simply make requests for latest offers and would receive the offers as a response

SOAP messages

- SOAP provides a standard 'envelope' within which a message can be delivered.

- SOAP is mechanism (protocol) for transferring information (messages) between applications which may be widely distributed.

- SOAP says nothing about the content of the message – the sender and the receiver must understand the message for themselves.

- SOAP is part of a communication stack.

SOAP Structure(1)

- Each SOAP message will have:
  - An `Envelope`
  - A `Header` (optional)
  - A `Body`
  - The `Body` may contain a `Fault` element

Transport protocol

MIME header

SOAP ENVELOPE

SOAP HEADER

SOAP BODY

FAULT
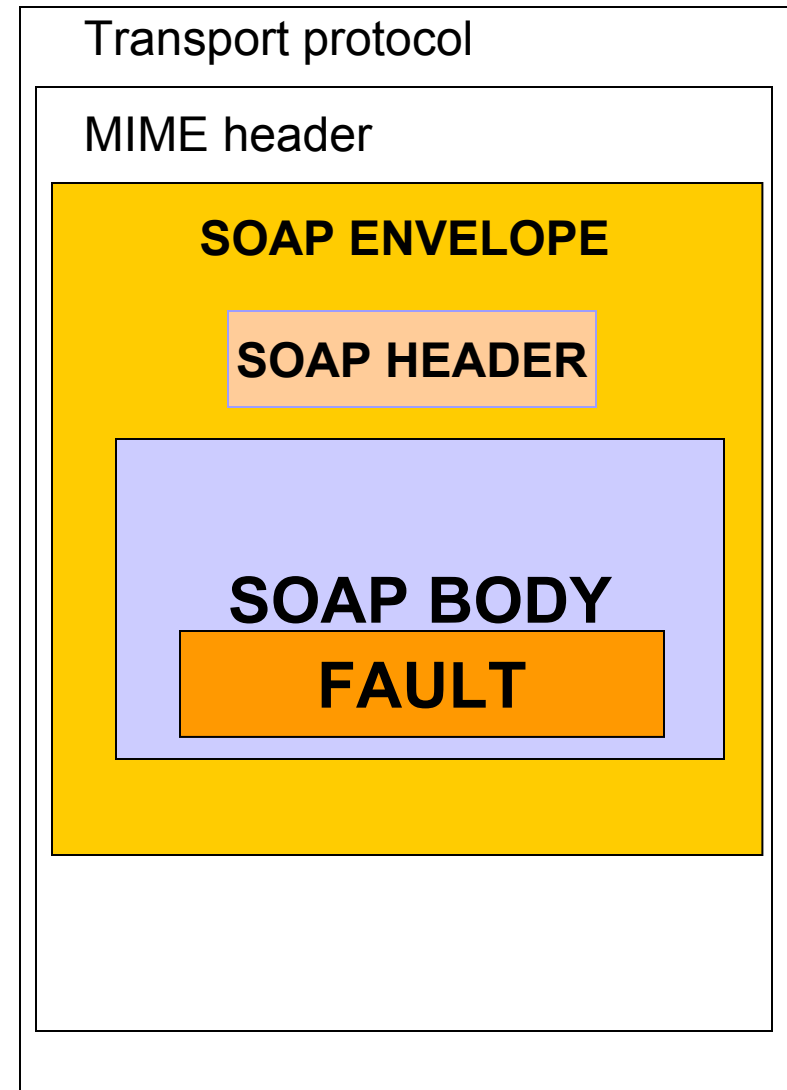
SOAP Structure(2)

- The envelope wraps the entire soap document

- The header contains allows additional information to be passed as well as the body of the document – e.g. authentication

- The body element contains the core of the SOAP document – this will contain either the RPC call or the XML message itself

- The fault information will contain any exception information

Anatomy of a SOAP message

```
<?xml version='1.0' encoding='UTF-8'?>

<SOAP-ENV:Envelope xmlns:SOAP_ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3c.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3c.org/1999/XMLSchema">

        <SOAP-ENV:Header>

        </SOAP-ENV:Header

        <SOAP_ENV:Body>

        </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

SOAP protocol binding

```
SOAPAction = "urn:soaphttpclient-action-uri"

Host = localhost

Content-Type = text/xml; charset=utf-8

Content-Length = 701
```

```
<SOAP-ENV:Envelope xmlns:SOAP_ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3c.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3c.org/1999/XMLSchema">
```

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

SOAP RPC

- SOAP RPC messages contain XML that represents a method call or method response

- The SOAP XML will be converted into a method call on the server and the response will be encoded into SOAP XML to be returned to the client

SOAP Faults

- SOAP errors are handled using a specialised envelope known as a Fault Envelope

- A SOAP Fault is a special element which must appear as an immediate child of the body element

- <faultcode> and <faultstring> are required.

A SOAP fault

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP_ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3c.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3c.org/1999/XMLSchema">
    <SOAP_ENV:Body>


            <SOAP-ENV:Fault>
                    <faultcode>SOAP-ENV:Server</faultcode>
                    <faultstring>Test fault</faultstring>
                    <faultactor>/soap/servlet/rpcrouter</faultactor>
                    <detail>

                                   ..
                    </detail>
            </SOAP-ENV:Fault>


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

SOAP Attachment

- Large quantities or binary data may not fit well into a XML SOAP message.

- In which case it can be sent 'out of band' by attaching it to a SOAP message

- *Analogy : email attachments.*

Transport protocol

MIME header

SOAP ENVELOPE

SOAP HEADER

SOAP BODY

FAULT

Attachment

Attaching a file to a SOAP message

- To add a file to a SOAP message a tag is added within the body of the message.
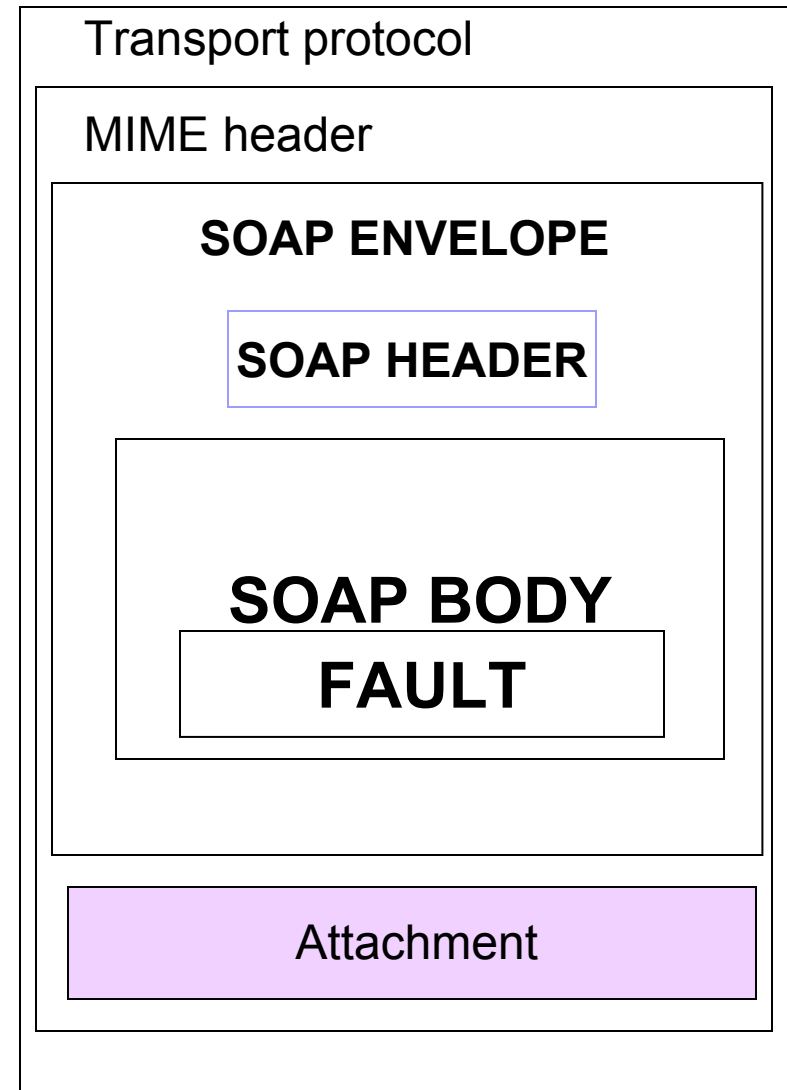
```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
xmlns:SOAP_ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3c.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3c.org/1999/XMLSchema">
        <SOAP_ENV:Body>


        <attachment href="{URL}"/>


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

eGee
Enabling Grids for
E-science in Europe

SOAP Deployment

Deployment:
Making the container aware of a servlet

- The web server has to be aware of the interface and exposed methods of a servlet in order to use it.

- Using Tomcat as an example this can be done in a number of ways.

1. Enter the values manually into the SOAP admin page from a Deployment descriptor.

2. You can use the SOAP manager application from the command line

3. You can manually edit Tomcat's WEB-INFO/web.xml file

4. You can create a WAR file and place it in Tomcat's webapps folder

5. You can use ANT

Using a WAR file

- A WAR file is basically an archive description of a servlet installation

  (*JAR and WAR naming derives from UNIX TAR – java archive, web archive, tape archive*).

- Example: placed in Tomcat's webapps folder it can be interpreted by the container.

Deployment Descriptor

A SOAP manager file

```
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment" id="urn:stock-
    onhand">
    <isd:provider type="java" scope="Application" methods="getQty">
        <isd:java class="StockQuantity"/>
    </isd:provider>
    <isd:faultListener>org.apache.soap.sever.DOMFaultListener</isd:faultListener>
</isd:service>
```

Some containers (Tomcat) provide GUIs for deployment

SOAP Implementations

- There are several implementations of the SOAP Specification
  - Apache Axis
  - GLUE

- Most J2EE application servers contain a SOAP implementation

- .NET has a SOAP implementation

- SOAP is also implemented in PERL.

Enabling Grids for
E-science in Europe

WSDL

The function of WSDL

- WSDL describes a service's exposed interface

- It is what a client sees of your service

- WSDL includes information about
  - The data types it uses
  - Parameters it requires and returns
  - Groupings of functionality
  - The protocol to be used to access the service
  - The location or address of the service

WSDL Structure

- A WSDL document is an XML document

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions>
    <types>
        <!- define the types here using XML Schema →
    </types>
    <message>
        <!- XML messages the web service uses are defined here →
    </message>
    <portType>
        <!- define the input and output parameters here -→
    </portType>
    <binding>
        <!- define the network protocol here →
    </binding>
    <service>
        <!- location of the service →
    </service>
</definitions>
```

<import> element

```
<definitions
    targetNamespace="urn:3950"
    xmlns= "http://schema.xmlsoap.org/wsdl/"
    xmlns:xsd= "http://www.w3c.org/2001/XMLSchema"
    xmlns:soap= "http://schemas.xmlsoap.org/wsdl/soap/"
    xmlnssoapenc= "http://schemas.xmlsoap.org/soap/emcoding/"
    xmlns:tns= "urn:3950">

    <import namespace= "http://nesc.ac.uk" location= "http://nesc.ac.uk/ez.xsd"/>
```

**Acts like C/C++ #include , or Java import.
Incorporates external namespaces**

Namespaces

- WSDL uses a number of different namespaces including

- XML Schema Namespaces
  - http://www.w3.org/2000/10/XMLSchema
  - http://www.w3c.org/2001/XML-Schema-instance

- WSDL Namespaces
  - http://schemas.xmlsoap.org/wsdl/soap/
  - http://schemas.xmlsoap.org/wsdl/

- SOAP Namespaces
  - http://schemas.xmlsoap.org/soap/encoding
  - http://schemas.xmlsoap.org/soap/envelope

The <types>

- The types element contains XML Schemas defining the datatypes that are to be passed to and from the web service

```
<types>
 <schema targetNamespace="http://example.com/stockquote.xsd"
        xmlns="http://www.w3.org/2000/10/XMLSchema">
  <element name="TradePriceRequest">
   <complexType>
     <all><element name="tickerSymbol" type="string"/></all>
   </complexType>
  </element>
  <element name="TradePrice">
    <complexType>
     <all><element name="price" type="float"/></all>
    </complexType>
  </element>
 </schema>
</types>
```

The <message>

- The `<message>` element is used to define the messages that will be exchanged between the client and the service
- These message elements contain `<part>` elements, which will be using types defined in the types element

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

- All the parts are namespace qualified

The <portType>

- The types and messages have been defined, but they have not been defined in terms of where they fit in the functionality of the web service

- This is done within <**portType**> and <**operation**> elements

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

- A portType is analogous to a class
- An operation is analogous to a method in that class

Types of <operation>

- There are four distinct types of operation

- Synchronous

  - **Request-response** - The service receives a message and sends a reply

  - **Solicit-response** - The service sends a message and receives a reply message

- Asynchronous

  - **One-way** - The service receives a message

  - **Notification** - The service sends a message

- All of these can be defined in WSDL

Defining the type of operation

- Presence and order of input/output elements defines the type of operation.

- Request-response `<input><output>`

- Solicit-response `<output><input>`

- One-way `<input>` only

- Notification `<output>` only

The <binding> element

- This element is used to define the mechanism that the client will actually use to interact with the web service

- There are three possibilities

  1. SOAP
  2. HTTP
  3. MIME

- The most common choice is currently SOAP

- The binding element defines the protocol specific information for the portTypes previously defined

The binding tag

```
<binding name="ez3950SOAPBinding" type="tns:ez3950PortTypes">
```

The `<binding>` tag indicates that we will map a `<Port Type>` to a protocol

```
<soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http/">
```

Indicates we will be using the SOAP binding extensions to map the operations.
The alternative to "rpc" is "document".

(to use GET/POST use `<http:binding…>`
 to use MIME use `<mime:binding…..>`)

<binding> Example

- Below is an example of a binding element for SOAP

```
<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>

- The final component of a WSDL file is the `<service>` element

- The `<service>` element defines `<port>` elements that specify where requests should be sent

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

- The `<soap:address>` subelement identifies the URL of the service

- The precise content of `<port>` elements will be dependent upon the mechanism, i.e. SOAP, HTTP or MIME

Objectives

- Architecture

- Standards
  - XML Schema
  - SOAP
  - WSDL
  - UDDI

- Context for Web Services