Web Services

Richard Sinnott

http://csperkins.org/teaching/2004-2005/gc5/



Overview

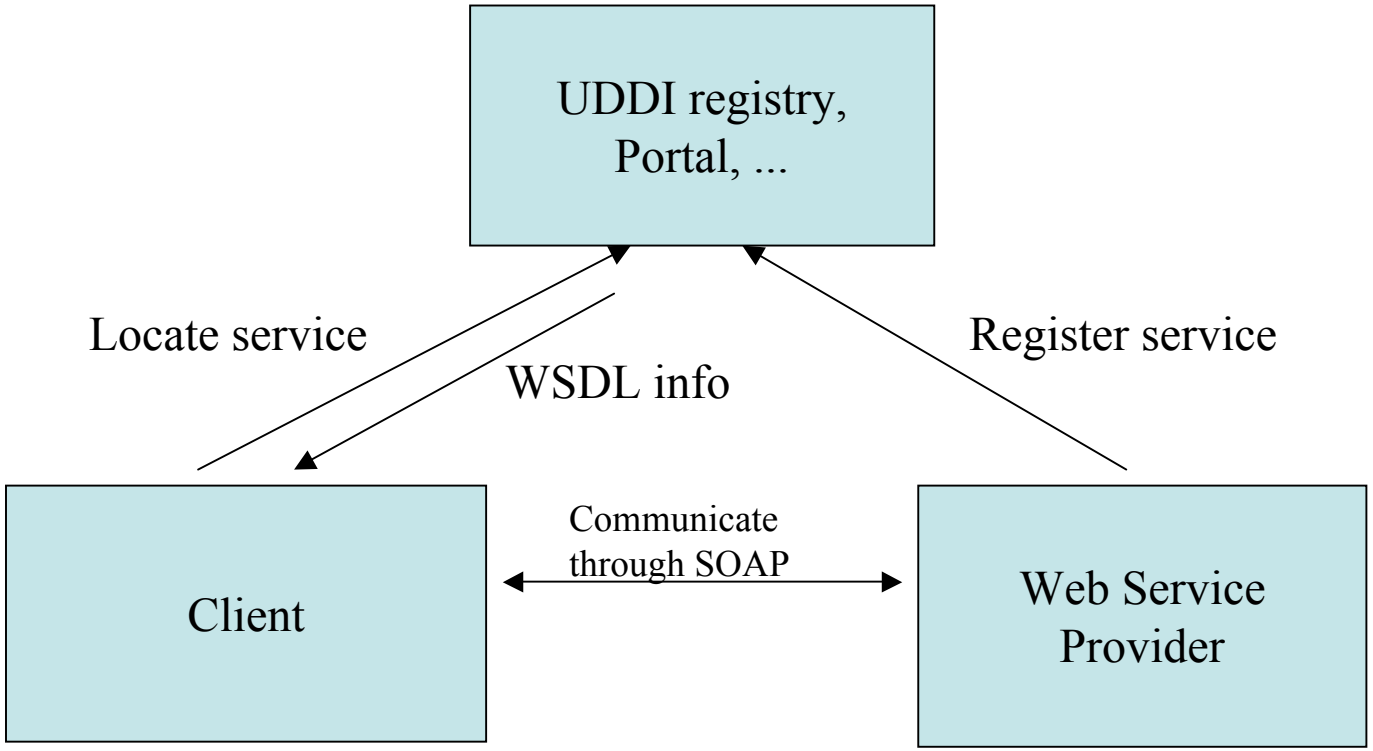
- Web Services Overview
- Technologies associated with web services
  - XML
  - XML schema and namespaces
  - SOAP
  - WSDL

    - Too much material for one lecture
      - may not get through everything…
      - Trying to bring class up to speed on course pre-requisites
      - Any questions ask in labs, next lecture, …

- Technologies for Building Grids (AG - virtual lecture tomorrow)
  - Ant, JAX-RPC, Web Servers, Apache Tomcat servlet containers, deployment of applications into containers, …

Copyright © 2004 University of Glasgow

Web Services



WSDL info = how to interact, what operations, what XML grammar flavour, …

Service Oriented Architecture is common approach

Copyright © 2004 University of Glasgow

Web Services

- Not a new idea
  - Publish, find, bind

- Core languages and technologies involved
  - XML
    - Simple Object Access Protocol (SOAP)
    - Simple API for XML (SAX),
    - Document Object Model (DOM)
    - Web Services Description Language (WSDL)
    - Universal Description, Discovery and Integration of Web Services (UDDI)
    - …

Copyright © 2004 University of Glasgow

XML is…

- …foundation for future Grid technologies
  - … an e**X**tensible **M**arkup **L**anguage
  - ... HTML on steroids
  - ... a semi-structured data model
  - ... an exchange syntax
  - …the ASCII of the Web
  - ... a meta-language that allows for evolutionary systems
    - XML constant but new markups, schemas …

Copyright © 2004 University of Glasgow

XML can be used to…

- … Exchange Data
  - With XML, data can be exchanged between incompatible systems
    - Converting the data to XML can greatly reduce this complexity and create data that can be read by many different types of applications
- … Share Data
  - With XML have a software and hardware-independent way of sharing data
- … Store Data
  - XML can also be used to store data in files or in databases and applications developed to store/retrieve information from the store, display the data etc
    - XML databases now exist (Xindice, …)
    - XML based query languages (XPath, XLink, XQuery, …)

Copyright © 2004 University of Glasgow

XML can be used to…

- …describe meta-data
  - information about the structure and meaning of data
  - can be used to perform more intelligent web searches for goods or information
    - Search for all genomic data produced within the last 3 years associated with the common house fly
      - (Rather than return every web site that mentions flies, genomes, …!)

- …support operation invocations, e.g. RPC
  - Information in XML documents parsed and used to invoke services
    - Simple Object Access Protocol – SOAP
      - A lightweight protocol for exchange of information in a decentralised, distributed environment
      - Web-Sites expose interfaces for interrogation

Copyright © 2004 University of Glasgow

XML History

- SGML (Standard Generalized Markup Language)
  - ISO Standard, 1986, for data storage & exchange
  - Meta-language for defining languages (through Document Type Definitions and Schemas)
  - A famous SGML language: HTML!!
  - Separation of content and display
  - SGML standard is 600 pages long

- XML
  - W3C - http://www.w3.org/XML/ recommendation in 1998
  - XML specification is 26 pages long
  - … but designed for extensibility and broad applicability!

Copyright © 2004 University of Glasgow

XML and HTML

- The main difference between XML and HTML
  - XML was designed to describe data and to focus on what data is
  - HTML was designed to display data and to focus on how data looks
  - HTML is about displaying information
  - XML is about describing information
  - HTML has predefined tags (<p>, <h1>, …)
  - XML developers define own tags

Copyright © 2004 University of Glasgow

XML Basics

- XML is a cross-platform, software and hardware independent tool for transmitting information
  - XML was designed to describe data
  - XML uses a Document Type Definition (DTD) to describe format and content of legal XML documents
    - … now deprecated
  - or an XML Schema to describe the structure of legal XML documents
    - Based upon XML syntax
    - Supports data types
    - Extensibility
    - …

Copyright © 2004 University of Glasgow

XML Basics ...Ctd

- XML was created to structure, store and to send information
- The following example is an XML memo to Rich from John

```
<memo>
    <to>Rich</to>
    <from>John</from>
    <heading>Reminder</heading>
    <body>Stop making me use GT3!</body>
</memo>
```

  - The memo has sender/receiver information, a header and a message body
  - But this XML document does not DO anything
    - It is just pure information wrapped in XML tags
    - Someone must write software to send, receive or display it

Copyright © 2004 University of Glasgow

XML Basics …ctd

- **An example XML document**

```
<?xml version="1.0" encoding="ISO-8859-1"?>   …XML declaration (version, character encoding used)
<memo>                                        …root element
    <to>Rich</to>                             …child element
        <from>John</from>                     …child element
        <heading>Reminder</heading>           …child element
    <body> Stop making me use GT3!</body>     …child element
</memo>                                       …end of root element
```

- **All XML elements must have a closing tag**
  - (Except for opening declaration)
    - In HTML some elements do not have to have a closing tag
      - e.g. <p>This is a paragraph <p>This is another paragraph
- **Unlike HTML, XML tags are case sensitive**
  - i.e. tag <Memo> is different from tag <memo>
    - <Memo>This is incorrect</memo>
    - <memo>This is correct</memo>

Copyright © 2004 University of Glasgow

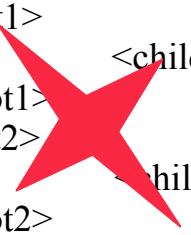
XML Basics …ctd

- **All XML elements must be properly nested**
  - **Improper nesting of tags is required in XML**
    - In HTML some elements can be improperly nested within each other:
      - This text is bold and italic
    - In XML all elements must be properly nested within each other:
      - This text is bold and italic

- **All XML documents must have a single root element**
  - **All XML documents must contain a single tag pair to define a root element**
    - All other elements must be within this root element
    - All elements can have sub elements (child elements)
    - Sub elements must be correctly nested within their parent element:

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

```
<root1>
  <child1>…</child1>
</root1>
<root2>
  <child2>…</child2>
</root2>
```



Copyright © 2004 University of Glasgow

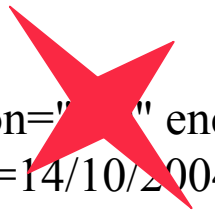
XML Basics …ctd

- Elements defined with **element** element… eh??? ;o)
  - User defines names, but
  - …can't start with number/punctuation character, no spaces, no >, <, :
- Can be empty
  - <name/> = = <name> </name>
- Attributes name/value pairs used once per element to provide extra information
  - <car price="expensive" price="cheap">myCar</car> …error
- Attributes vs elements
  - <car><colour>red</colour></car> *vs* <car colour="red">
    - General rule is elements provide logical structuring of information and attributes characteristics of the information

Copyright © 2004 University of Glasgow

XML Basics …ctd

- XML elements can have attributes in name/value pairs just like in HTML
  - In XML the attribute value must always be quoted
    - it is illegal to omit quotation marks around attribute values



```
<?xml version=" " encoding=“UTF-8"?>
<memo date=14/10/2004>
    <to>Rich</to>
    <from>John</from>
    …
</memo>
```

```
<?xml version="1.0" encoding=“UTF-8"?>
<memo date="14/10/2004">
    <to>Rich</to>
    <from>John</from>
    …
</memo>
```

Copyright © 2004 University of Glasgow

XML Basics …ctd

- **Comments in XML**
  - The syntax for writing comments in XML is similar to that of HTML

```
<!-- This is a comment -->
```

- **Whitespace**

```
<?xml version="1.0" encoding="“UTF-8"?>
<memo  date="14/10/2004">
   <to>Rich       </to>

   <from>John</from>
    <body> Stop making me use
                                                    GT3!</body>
</memo>
```

    - …XML processors must pass all non-markup characters to application
      - also have xml:space = preserve/default if whitespace significant

- **There is nothing special about XML**
  - It is just plain text with the addition of some XML tags enclosed in angle brackets
  - But… XML combined with software to process tags, contents, leads to very open “internet scale” systems

Copyright © 2004 University of Glasgow

XML Basics …ctd

- **How can XML be used?**
  - **Need to produce vocabulary for meaningful exchanges, for tools to process**
    - Tools available to parse XML and ensure well-formed (syntactically correct)
      - must begin with the XML declaration
      - must have one unique root element
      - all start tags must match end-tags
      - XML tags are case sensitive
      - all elements must be closed
      - all elements must be properly nested
      - all attribute values must be quoted
      - XML entities must be used for special characters (&gt = >, &lt = <, …)
    - But flexibility comes through defining XML that can be **validated**
      - Legal grammars/format of content for exchanges
    - Previously done by XML Document Type Definitions
      - Now XML Schemas accepted as best way to achieve this
  - XML Schemas define the legal building blocks of an XML document

Copyright © 2004 University of Glasgow

XML Schema Basics

- **Even if XML documents are well-formed they can still contain errors**
  - With XML Schemas many errors can be caught
  - Need to understand what XML tags/values mean…
- **An XML Schema:**
  - defines elements that can appear in a document
  - defines attributes that can appear in a document
  - defines which elements are child elements
  - defines the order of child elements
  - defines the number of child elements
  - defines whether an element is empty or can include text
  - defines data types for elements and attributes
  - defines default and fixed values for elements and attributes

Copyright © 2004 University of Glasgow

XML Schema Basics

- **One of the greatest strengths of XML Schemas (over DTDs) is the support for data types**
  - With the support for data types it is easier to:
    - describe permissible document content
      - <xsd:element name="age" type="xsd:integer"/>
      - <age>true</age> …error
    - validate the correctness of data
      - 2004-10-9 = 10th September 2004 or 9th October 2004
      - <date type="date">2004-10-9</date>
      - Schema defines format YYYY-MM-DD
    - define restrictions on data
      - <xsd:element name="name" type="xsd:string" **fixed**="Rich" />
        <name /> …ok, i.e. can be null
        <name>Rich</name> …ok
        <name>John</name> …error
    - define data patterns/formats
      - <xsd:element name="name" type="xsd:string" **default**="Rich" />
        <name>Rich</name> = = <name />
    - convert data between different data types

Copyright © 2004 University of Glasgow

XML Schema Basics …ctd

- **XML Schemas are extensible**
  - With an extensible Schema definition you can:
    - Reuse your Schema in other Schemas
      - can use import/include elements
    - Create your own data types derived from standard types
    - Reference multiple schemas from the same document
- Key to this are Namespaces

```
<car>                          <customer>
    <name>Volvo</name>             <name>Rich</name>
</car>                         </customer>
```

  - Now consider this XML document

```
<customer>
    <name>Rich</name>          …name element child of customer element
    <order>
        <car>
            <name>Volvo</name>  …name element child of car element
        </car>
    </order>
</customer> …..parsers cannot tell difference between name elements unless instructed
```

Copyright © 2004 University of Glasgow

XML Schema Basics …ctd

- XML Namespaces allow to overcome this restriction

```
<customer>
   <cust:name xmlns:cust="cust-ns-uri">Rich</cust:name>
    <order>
          <car>
              <car:name xmlns:car ="car-ns-uri">Volvo</car:name>
          </car>
    </order>
</customer>
```

- Note
  - URI = uniform resource identifier (formatted string) has two flavours
    - URL (uniform resource locator), e.g. "http://www.nesc.ac.uk/projects"
      - includes information on where to find resource (IP address/port number/path…)
    - URN (uniform resource name), e.g. urn:uuid:12345
      - Location independent (name space identifier, name specific string)

- Namespaces define scope of names/variables
  - e.g. JAVA packages, C++ #include…

Copyright © 2004 University of Glasgow

XML Schema Basics …ctd

- Can declare a name space in two ways
  - Default declaration

```
<customer  xmlns="cust-ns-uri">
   <name>Rich</name>
   <order>
         <car>Volvo</car>
   </order>
</customer>
```

    - All child elements of customer use "cust-ns-uri" namespace
    - Normally used only for local schema usage

  - Explicit declaration

```
<co: customer xmlns:co ="cust-ns-uri">
   <co:name>Rich</co:name>
   <co:order>
         <co:car>
            <co:name>Volvo</co:name>
         </co:car>
   </co:order>
</co:customer>…co = shorthand notation for "cust-ns-uri"
```

    - Normally used when schema for external usage

Copyright © 2004 University of Glasgow

XML Schema Basics …ctd

- True power of namespaces when using elements from different namespaces

```
<cust: customer xmlns:cust =“cust-ns-uri”
                        xmlns:ord = “ord-ns-uri” >
   <cust:name>Rich</cust:name>
    <ord:order>
          <ord:car>
              <ord::name>Volvo</ord:name>
           <ord:car>
    </ord:order>
</cust:customer>
```

  - Different sets of rules applied for customer names, order information

Copyright © 2004 University of Glasgow

XML Schema Basics …ctd

- Common namespace prefixes

xsi	http://www.w3c.org/2000/10/XMLSchema-instance *namespace governing XMLSchema instances*
xsd	http://www.w3c.org/2000/10/XMLSchema *namespace of schema governing XMLSchema (.xsd) files*
tns	by convention this refers to “this” document *refers to the current XML document*
wsdl	http://schemas.xmlsoap.org/wsdl/ *WSDL namespace*
soap	http://schema.xmlsoap.org/wsdl/soap/ *WSDL SOAP binding namespace*

Copyright © 2004 University of Glasgow

XML Schema Basics ...ctd

- A simple schema

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
    targetNamespace="http://www.nesc.ac.uk/nescSchema" xmlns="http://www.nesc.ac.uk/ns" >

<xsd:element name="memo">
        <xsd:complexType>     … as contains other elements
            <xsd:sequence>
                <xsd:element name="to" type="xsd:string"/>          ...simple type
                <xsd:element name="from" type="xsd:string"/>       ...simple type
                <xsd:element name="heading" type="xsd:string"/> ...simple type
                <xsd:element name="body" type="xsd:string"/>       ...simple type
            </xsd:sequence>                                              (as no other elements)
        </xsd:complexType>
</xsd:element>
</xsd:schema>
```

- Target namespace (intended namespace) against which XML document should be validated

Copyright © 2004 University of Glasgow

XML Schema Basics …ctd

- Common primitive data types used in schemas
  - xsd:string
  - xsd:decimal
  - xsd:integer
  - xsd:boolean
  - xsd:date
  - xsd:time
  - …
- More complex structures supported also,
  - e.g. element groups for logically associated elements
    - xsd:sequence (ordering elements important),
    - xsd:choice (some elements),
    - xsd:all (ordering unimportant)
  - Can have logical groupings of attributes also

Copyright © 2004 University of Glasgow

More Information on XML

- This is a very short summary of XML, XML Schema, XML namespaces
- Lots more information available
  - http://www.w3schools.com/xml/xml_whatis.asp
  - http://www.xmlfiles.com/xml/
  - Are good beginner guides

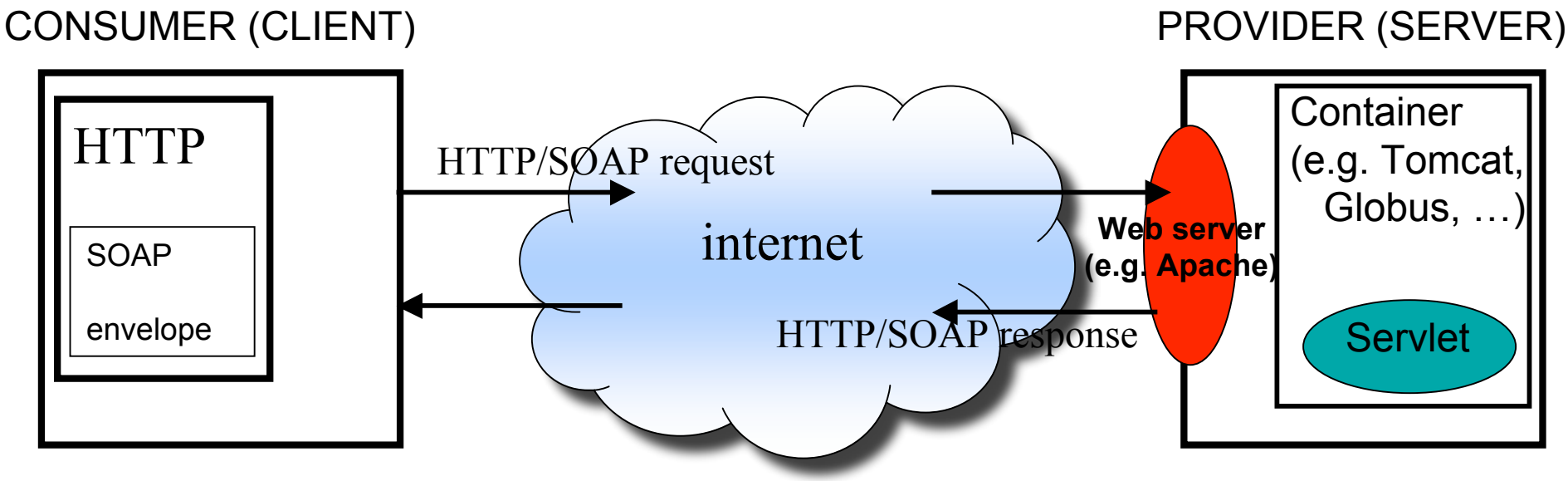
Copyright © 2004 University of Glasgow

SOAP Overview

- SOAP provides a standard ‘envelope’ within which a message to a web service 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

Copyright © 2004 University of Glasgow

Basic Web Services Architecture

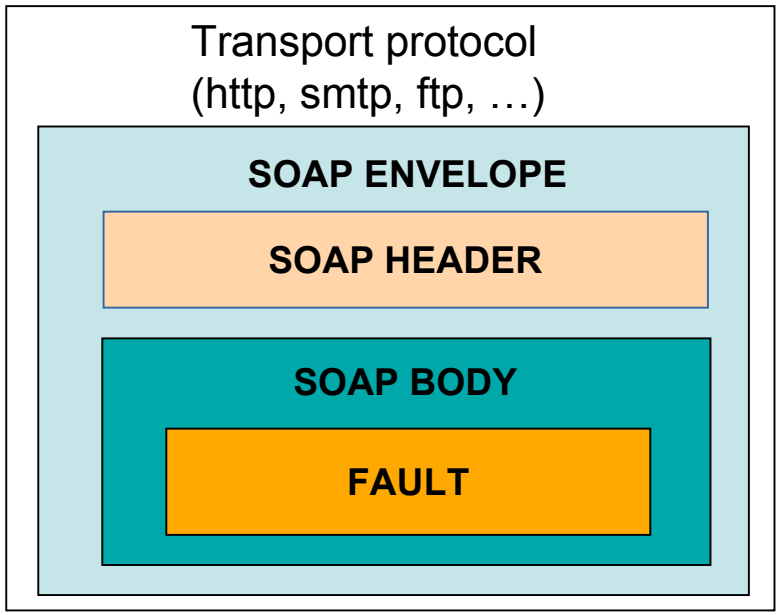


- Servlet is server side application running in container that processes SOAP information
- Web server extracts information from http/soap envelope and directs to appropriate service being hosted in container

Copyright © 2004 University of Glasgow

SOAP Message Structure

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




```
<?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>
        …typically used for authentication, transaction support, payment information, or other capabilities
        …mustUnderstand attribute can be used to ensure that header is processed
        …actor attribute used for indicating recipient of message (not needed if only two parties involved)
  </SOAP-ENV:Header>
  <SOAP_ENV:Body>
        …mandatory child of envelope element used for carrying payload of SOAP message
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Copyright © 2004 University of Glasgow

SOAP Over HTTP

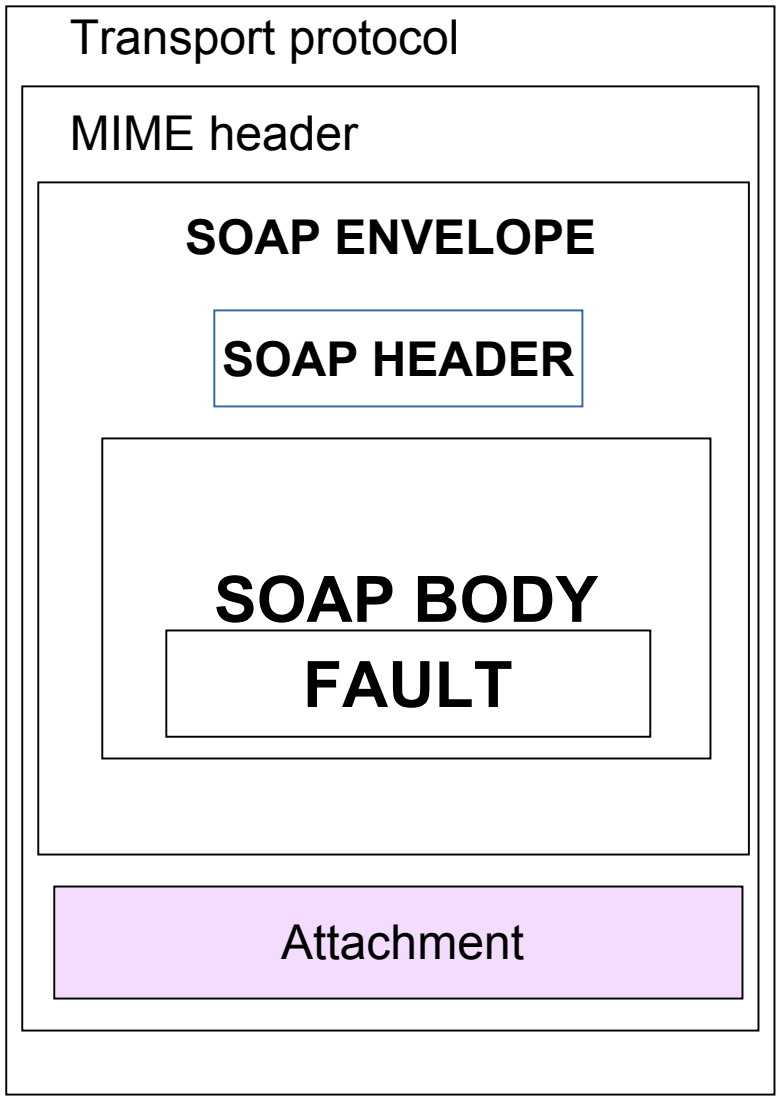
- HTTP supports two request methods
  - GET
    - send parameters in URL
    - typically used to request web pages from web server
  - POST
    - Send data to server in payload that comes after http header
    - POST payloads can be indefinite length
    - SOAP requests sent via HTTP POST
  - Consider HTTP POST request with simple SOAP message

```
POST /someOpLocation HTTP/1.1          …someOpLocation = SOAP endpoint
HOST: 130.29.25.3
Content Type: text/xml                 …normally text/plain for web pages
Content Length: 500                    …total length of message (XML BLOAT)
SOAPMethodName: urn:www.nesc.ac.uk:someOp#op   …op is method to be called of someOp class
<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>
        <ns:op xmlns:ns="someOp"><name>op-inputs</name></ns:op>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Copyright © 2004 University of Glasgow

SOAP Attachments

- 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.*



Copyright © 2004 University of Glasgow

Attaching Files to SOAP Messages

- To add a file to a SOAP message a tag can be 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>
```

Copyright © 2004 University of Glasgow

SOAP Fault Handling

```
<?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>
```

Copyright © 2004 University of Glasgow

SOAP Implementations

- There are several implementations of the SOAP Specification
  - Microsoft with Visual Basic, C#, .net languages/environments
  - Apache Axis with Java
  - SOAP::Lite with Perl
  - Systinet WASP with C++, Java
  - GLUE with Java

- More information on SOAP available at
  - http://www.w3schools.com/soap/default.asp
  - http://www.topxml.com/soap/default.asp

Copyright © 2004 University of Glasgow

WSDL Overview

- Web Service Description Language (WSDL) describes a service's exposed interface
  - It is what a client sees of a web (Grid) service
    - Contact between client and service

- WSDL describes what a web service can do, where it can be found, how to invoke it

  - Consider WSDL as XML grammar that describes web services (and in general any network service) as collections of communications endpoints that are able to exchange information with one another

Copyright © 2004 University of Glasgow

WSDL Document Overview

- WSDL documents use the following elements
  - Definitions
    - Associates web services with their namespaces
  - Types
    - Containers for data type definitions (schemas)
  - Message
    - Abstract typed definition of data contained in message
  - Operations
    - Abstract descriptions of actions that web service supports
  - PortType
    - Sets of operations supported by one or more endpoints
  - Binding
    - Specification of protocol and data format for particular portType
  - Port
    - an endpoint defined in terms of binding and network address, e.g. URL
  - Service
    - Collection of related endpoints

Copyright © 2004 University of Glasgow

The <types>

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

```
<types>
 <schema targetNamespace="http://myExample.com/myService.xsd"
xmlns="http://www.w3.org/2000/10/XMLSchema">
  <element name="myRequest">
     . . .
  </element>
 </schema>
</types>
```

Copyright © 2004 University of Glasgow

The <message>

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

```
<message name="getMyInput">
   <part name="body" element="xsd1:MyRequest"/>
</message>
<message name="getMyOutput">
   <part name="body" element="xsd1:MyParameter"/>
</message>
```

- All the parts are namespace qualified

Copyright © 2004 University of Glasgow

Types of <operation>

- There are four distinct types of operation
- Synchronous
  - **Request-response**
    - service receives a message and sends a reply
  - **Solicit-response**
    - service sends a message and receives a reply message
- Asynchronous
  - **One-way**
    - service receives a message
  - **Notification**
    - service sends a message
- Presence and order of input/output elements in WSDL defines the type of operation.
    - Request-response `<input><output>`
    - Solicit-response `<output><input>`
    - One-way `<input>` only
    - Notification `<output>` only

Copyright © 2004 University of Glasgow

The <portType> element

- 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

Copyright © 2004 University of Glasgow

The <binding> element

- This element is used to define the mechanism that the client will actually use to interact with the web service, e.g. SOAP
- The binding element defines the protocol specific information for the portTypes previously defined

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

  - Indicates using SOAP binding extensions to map the operations
  - alternative to "rpc" is "document".
  - GET/POST use <http:binding…>
  - MIME use <mime:binding…..>

Copyright © 2004 University of Glasgow

The <service> element

- 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="MyService">
    <port name="MyServicePort" binding="tns:MyServiceBinding">
        <soap:address location="http://MyExample.com/MyService" />
    </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

Copyright © 2004 University of Glasgow

Conclusions

- Knowledge of XML, XML Schema, XML namespaces, SOAP, WSDL fundamental to web (Grid) services

- Grid services (based on OGSI) extend web services with further capabilities
  - Persistence/transience
  - Statefulness
  - inheritance of portTypes
  - service data
  - …
  - Explore this with GT3 in lab session on Monday

- Next (virtual) lecture focuses upon technologies for building Grid services

Copyright © 2004 University of Glasgow