Web Services 1

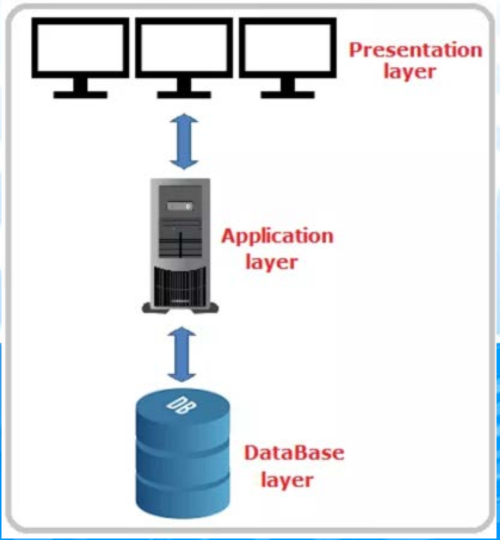
Contents

Before the Web

- Information Systems consist of 3 layers
  - Resource Management (database) Layer
  - Application Logic/Business Layer
  - Presentation (user interface) Layer
- Physical separation of these layers as Tiers

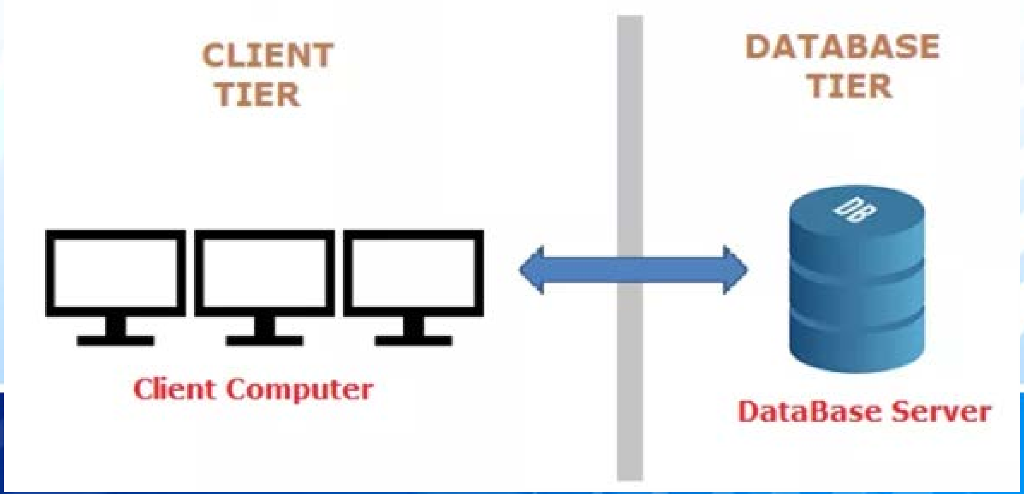
1-Tier Architecture

- 3 layers run on one machine or implemented in one software package
  - Dumb terminals and mainframes
  - Layers tightly connected
  - Difficulty with scalability and portability



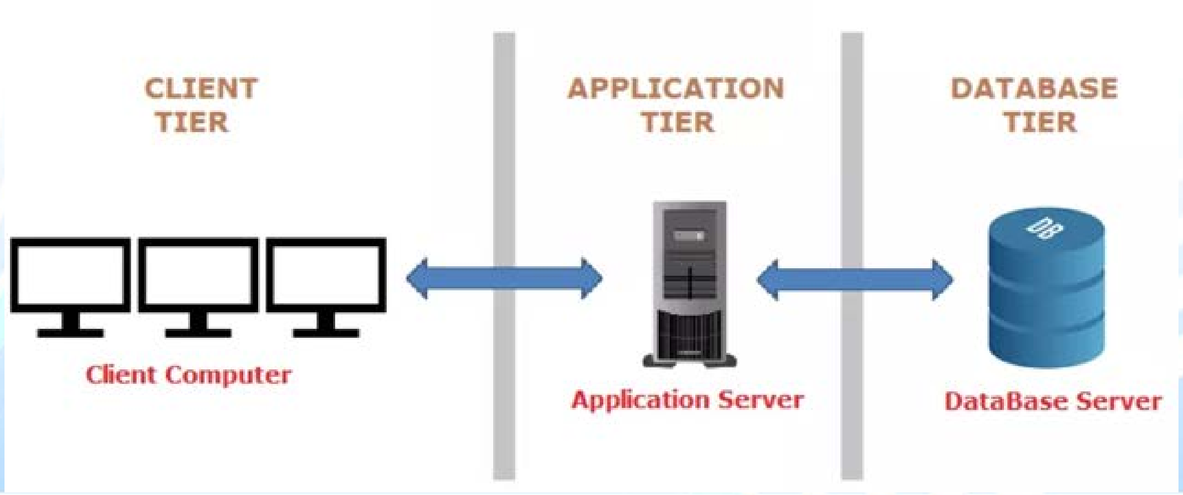
2-Tier Architecture

- It is called Client-Server architecture
- Typically the presentation layer runs on the client machine and the data layer on the server side
- Application layers can run either on client or server



3-Tier Architecture

- Each layer almost independent and running on a separate tier
  - Three Layers sometimes called **front-end** (or GUI), **middleware** and **back-end**

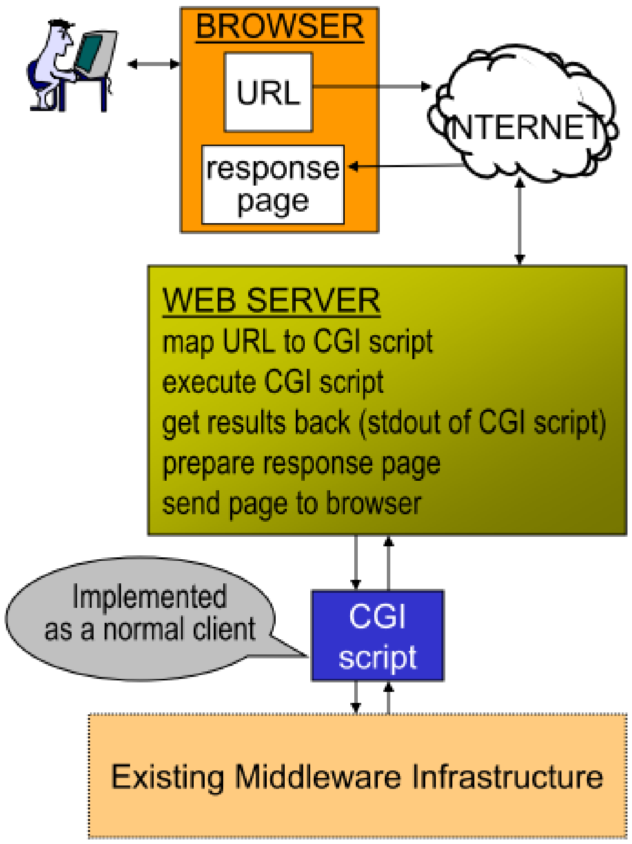


N-Tier Architecture

- AKA **Distributed application**
- It is similar to 3-tier architecture
- However, number of application servers are increased and represented in **individual tiers** in order to distributed the business logic

WWW basics

- The earliest implementations were very simple and built directly upon the existing systems (client/server)
  - The user clicked in a given URL and the server invoked the script corresponding to that URL
  - the CGI script (or program) acted as client in the traditional sense
  - the script executed, produced the results and passed them back to the server (usually as the address of a web page)
  - the server retrieved the page and send it to the browser

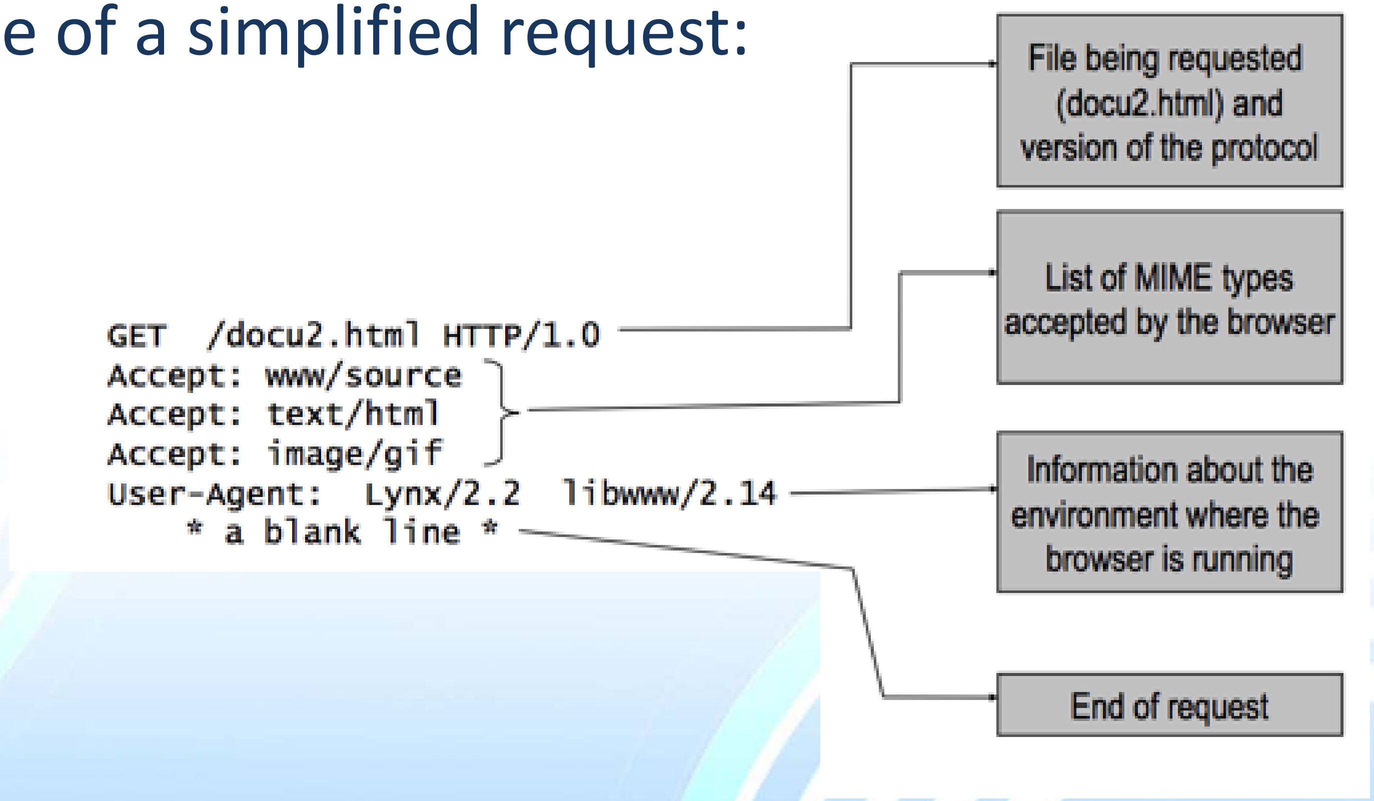


HTTP as a communication protocol

- HTTP was designed for exchanging documents. It is almost like e-mail (in fact, it uses RFC 822 compliant mail headers and MIME types)
- The **MIME type** is the mechanism to tell the client the variety of document transmitted
- Example of a simplified request:

HTTP as a communication protocol (cont.)

- Example of a simplified request:

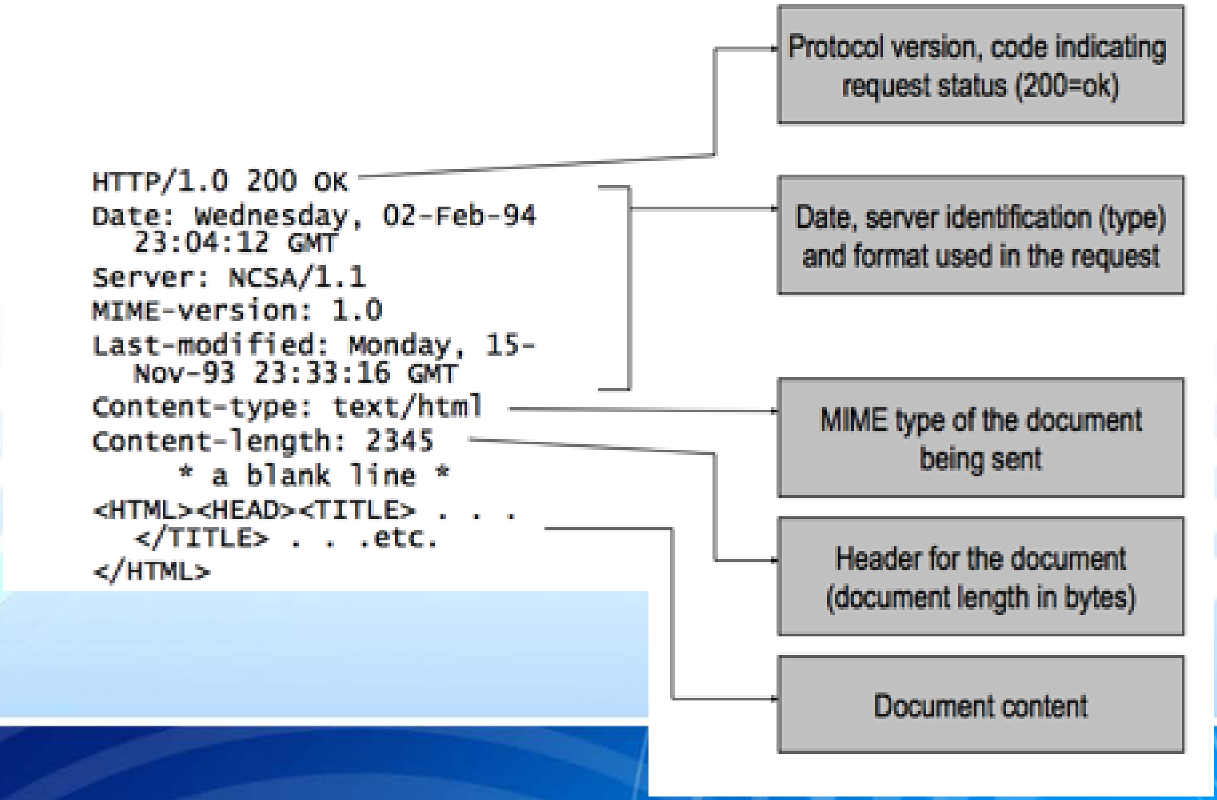


```
GET  /docu2.html HTTP/1.0
Accept: www/source
Accept: text/html
Accept: image/gif
User-Agent:  Lynx/2.2  libwww/2.14
    * a blank line *
```

File being requested (docu2.html) and version of the protocol
List of MIME types accepted by the browser
Information about the environment where the browser is running
End of request

HTTP server side response

- Example of a response from the server



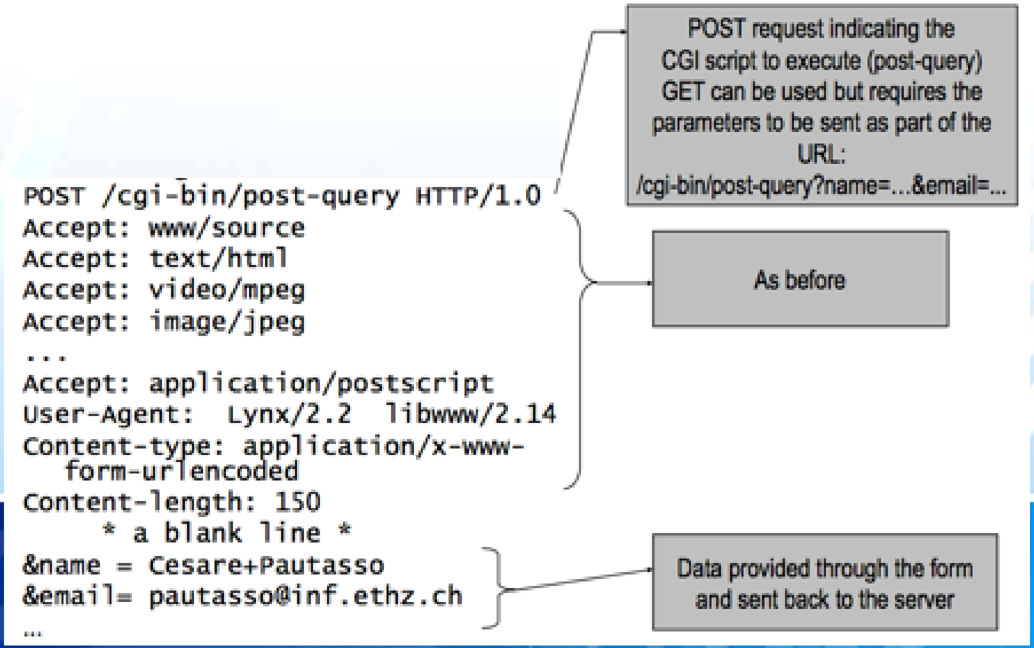
```
HTTP/1.0 200 OK
Date: Wednesday, 02-Feb-94
   23:04:12 GMT
Server: NCSA/1.1
MIME-version: 1.0
Last-modified: Monday, 15-
   Nov-93 23:33:16 GMT
Content-type: text/html
Content-length: 2345
      * a blank line *
<HTML><HEAD><TITLE> . . .
   </TITLE> . . .etc.
</HTML>
```

Protocol version, code indicating request status (200=ok)
Date, server identification (type) and format used in the request
MIME type of the document being sent
Header for the document (document length in bytes)
Document content

fppt.com

Passing Parameter

- The introduction of forms for allowing users to provide information to a web server required to modify HTML (and HTTP) but it provided a more advanced interface than just retrieving files:



```
POST /cgi-bin/post-query HTTP/1.0
Accept: www/source
Accept: text/html
Accept: video/mpeg
Accept: image/jpeg
...
Accept: application/postscript
User-Agent:  Lynx/2.2  libwww/2.14
Content-type: application/x-www-
   form-urlencoded
Content-length: 150
      * a blank line *
&name = Cesare+Pautasso
&email= pautasso@inf.ethz.ch
...
```

POST request indicating the CGI script to execute (post-query) GET can be used but requires the parameters to be sent as part of the URL: /cgi-bin/post-query?name=...&email=...
As before
Data provided through the form and sent back to the server

Contents and presentation

- HTML is a markup language designed to describe how a document should be displayed (the visual format of the document).
- HTML is one of the many markup languages that exist, some of them having being in use before HTML even existed such as SGML (Standard Generalized Markup Language)

HTML and XML

- HTML only provides primitives for formatting a document with a human user in mind
- Using HTML there is no way to indicate what are the contents of a document (its semantics)
- To cope with this requirement, the XML standard was proposed

XML

- XML tags not pre-defined like HTML
- The goal of XML is to provide a **standardized way** to specify **data structures** for **data exchange and storage**
- XML Schemas support data types
- Unlike HTML, XML is not intended for browsers
- XML can be automatically processed by other programs and machines
- XML can be used as the intermediate language for marshalling/serializing arguments when invoking services across the Internet

XML (cont.)

- XML documents form a tree structure that starts at "the root" and branches to "the leaves“
- Example:

XML Declaration – Version and Encoding

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
      <to>Wud</to>
      <from>Bell</from>
      <heading>Reminder</heading>
      <body>Please buy fruits today</body>
</note>
```

Root element

Child element

End Root element

XML Schema and XML Documents

```
<xs:element name="university">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="address" type="xs:string"/>
      <xs:element name="city" type="xs:string"/>
      <xs:element name="country" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<university>
    <name>Mahidol</name>
    <address> 999 Phuttamonthon 4 Road</address>
    <city>Nakhon Pathom</city>
    <country>Thailand</country>
 </university>
```

Web Services

- "A Web service is a software system/application designed to support interoperable machine-to-machine interaction over a network" W3C
- Hosting services on a remote machine
- A standardized way of integrating web-based applications
- The request and the response encoded in a format easy for a program to decode
  - The most common encodings are XML (SOAP or POX) and JSON
- SOAP and RESTful web services (RESTful Web APIs)

Open Standards

- It is built on open standards:
  - Simple Object Access Protocol (SOAP)
  - Web Services Description Language (WSDL)
  - Hypertext Transfer Protocol (HTTP)
    - Supported as a transport protocol – SOAP/HTTP for transporting messages across network applications
  - Representational state transfer (REST)

SOAP Web Service Definitions

World Wide Web consortium (W3C):

“a *software application* identified by *a URI*, whose *interfaces* and *bindings* are capable of being defined, described, and discovered as *XML artifacts*.

“A web service supports direct interactions with other software agents using XML-based messages exchanged via *Internet-based protocols*.”

SOAP Web Service Definitions (cont.)

UDDI(**Universal Description, Discovery, and Integration**) consortium:

“self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces”

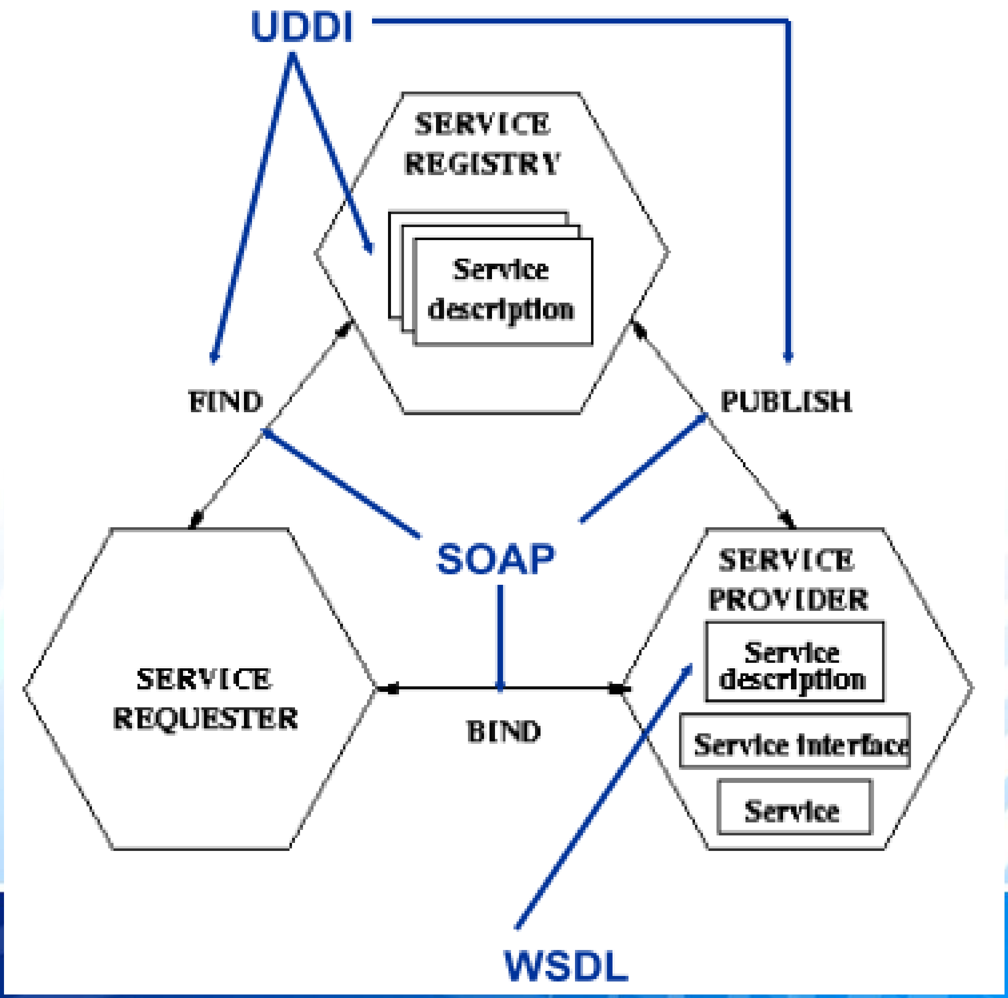
SOAP Web Service Definitions (cont.)

- Web services use the **XML, SOAP, WSDL and UDDI open standards** over an Internet protocol backbone
- **XML** provides an open standard for data exchange, **HTTP** an open transport protocol
- A Web Service
  - has an **interface** describing **a collection of operations**
  - enables **access** to **business logic**, **data** and **processes** or **other services**
  - can be **accessed by humans**, **other applications** or **other WSs**
  - all **communications in XML** so not limited to any operating system or programming language (SOAP)
  - **easy and cheap** to develop with so **many supporting tools**
  - **Motivations:** Enterprise Application Integration (EAI), Supply chain and B2B

UDDI

- UDDI is an XML-based standard for describing, publishing, and finding web services.
  - UDDI stands for **Universal Description, Discovery, and Integration.**
  - UDDI is a specification for a distributed registry of web services.
  - UDDI is a platform-independent, open framework.
  - UDDI can communicate via SOAP, CORBA, Java RMI Protocol.
  - UDDI uses Web Service Definition Language(WSDL) to describe interfaces to web services.
  - UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services.
  - UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet.

Web Service Overview



SOAP

- SOAP Basics
- SOAP Message Format
- SOAP Implementation
- SOAP binding with http

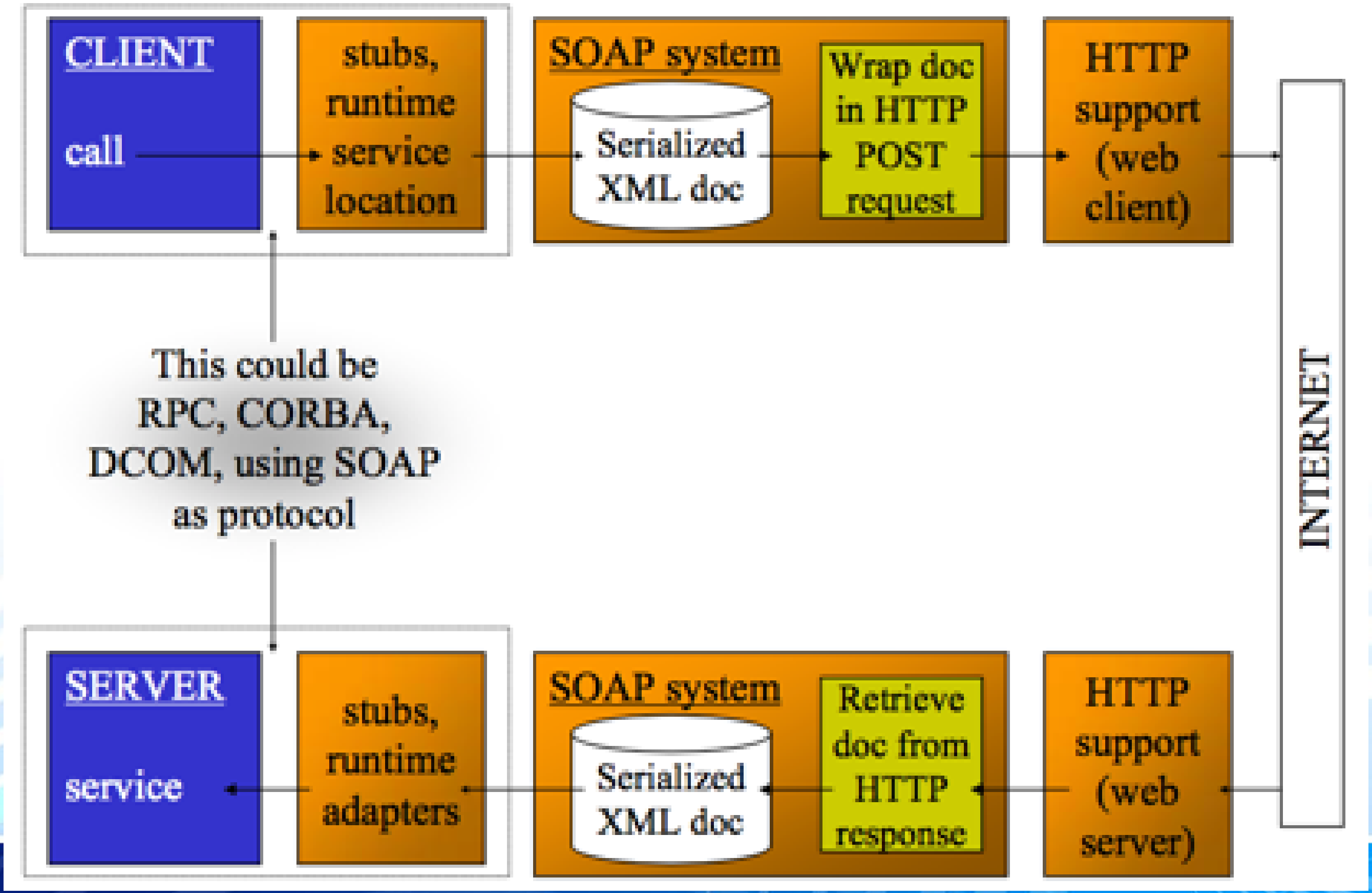
SOAP Basics

- SOAP is an XML-based messaging protocol.
- SOAP is an application communication protocol
- SOAP is designed to communicate via Internet
- It defines a set of rules for structuring messages that can be used for simple one-way messaging but is particularly useful for performing RPC-style (Remote Procedure Call) request-response dialogues.
- It is not tied to any particular transport protocol
- SOAP is platform and language independent
- SOAP allows you to get around firewalls

Why not to use other protocols

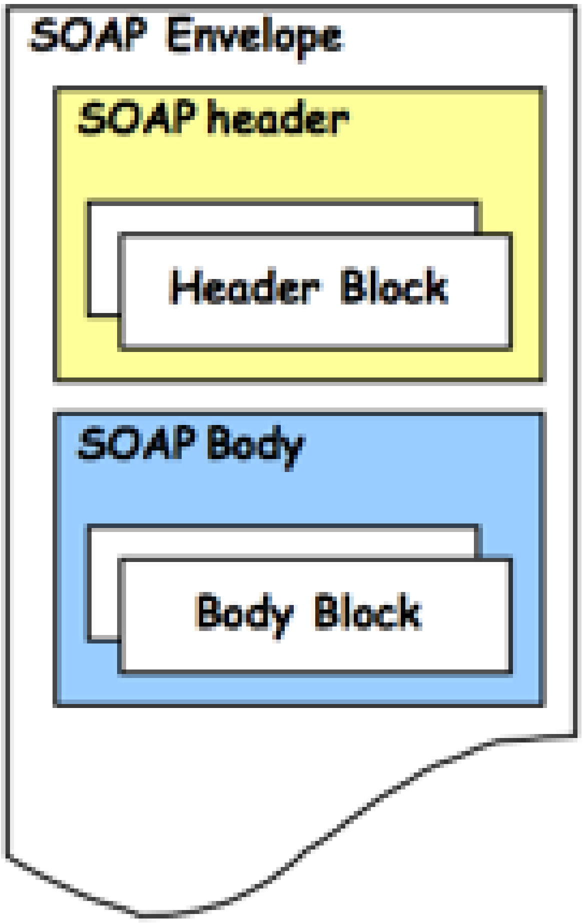
- **across firewalls**, so need to support HTTP or SMTP (so no RPC or RMI)
- sometimes you **don’t want** to have to wait for an ACK
- sometimes you **want an answer back**
- sometimes you want to send **XML documents, not make procedure or method calls** (so no RPC or RMI)
- sometimes you want to **simulate RPC** (Remote Procedure Call (RPC) is a protocol that one program can use to request a service from a program located in another computer on a network without having to understand the network's details.)
- sometimes you want to **use over existing transport** protocols: e.g., SMTP, HTTP, FTP, even TCP

SOAP as RPC mechanism

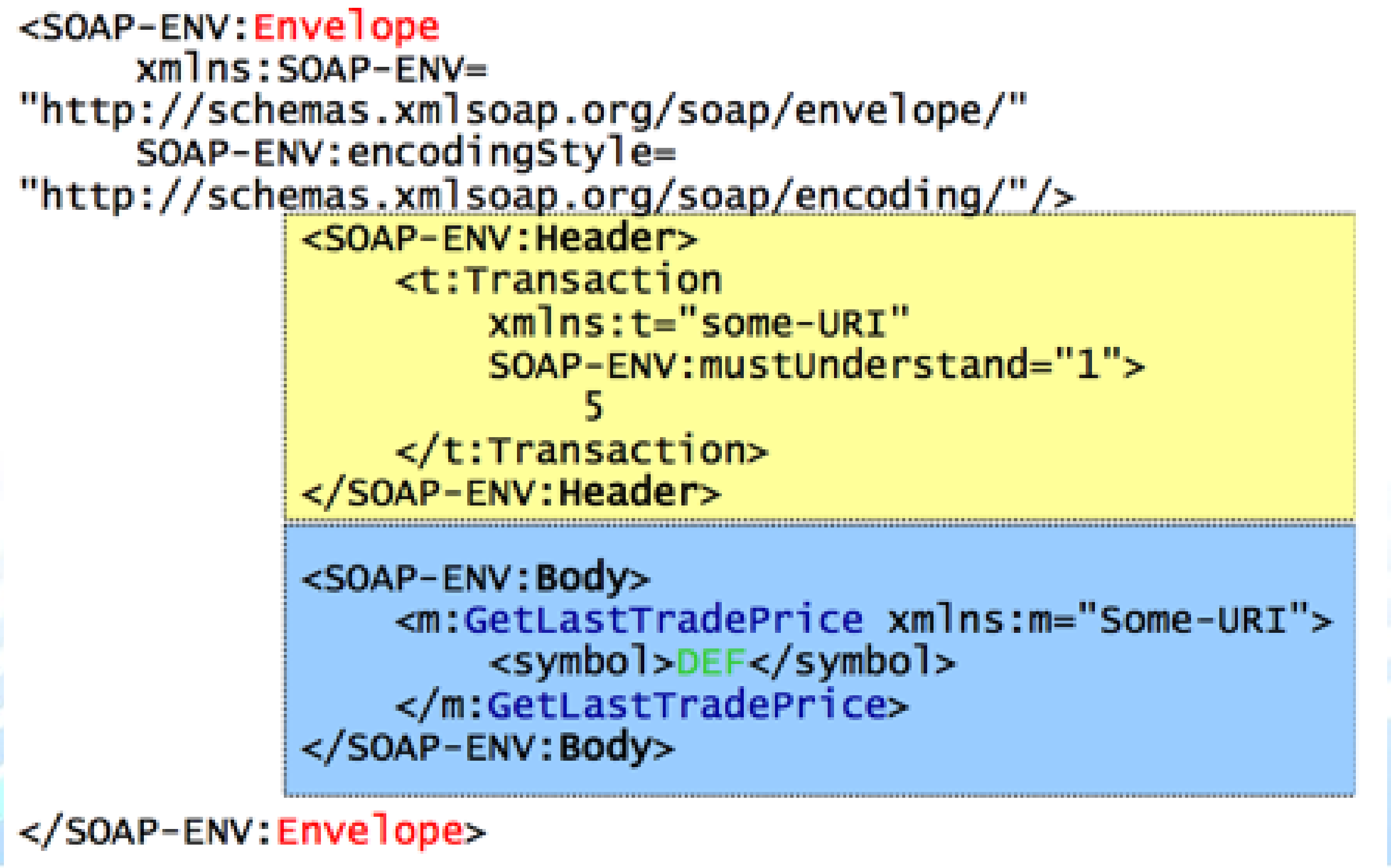

CLIENT
call
stubs, runtime service location
SOAP system
Serialized XML doc
Wrap doc in HTTP POST request
HTTP support (web client)
INTERNET
This could be RPC, CORBA, DCOM, using SOAP as protocol
SERVER
service
stubs, runtime adapters
SOAP system
Serialized XML doc
Retrieve doc from HTTP response
HTTP support (web server)

SOAP Messages

- SOAP is based on message exchanges
- Messages are structured with an envelope where the application encloses the data to be sent
- A message has two main parts:
  - **header**: which can be divided into blocks
  - **body**: which can also be divided into blocks
- SOAP does not say what to do with the header and the body, it only states that the **header is optional** and the **body is mandatory**
- Use of header and body, however, is implicit. The body is for application level data. The header is for infrastructure level data



SOAP example

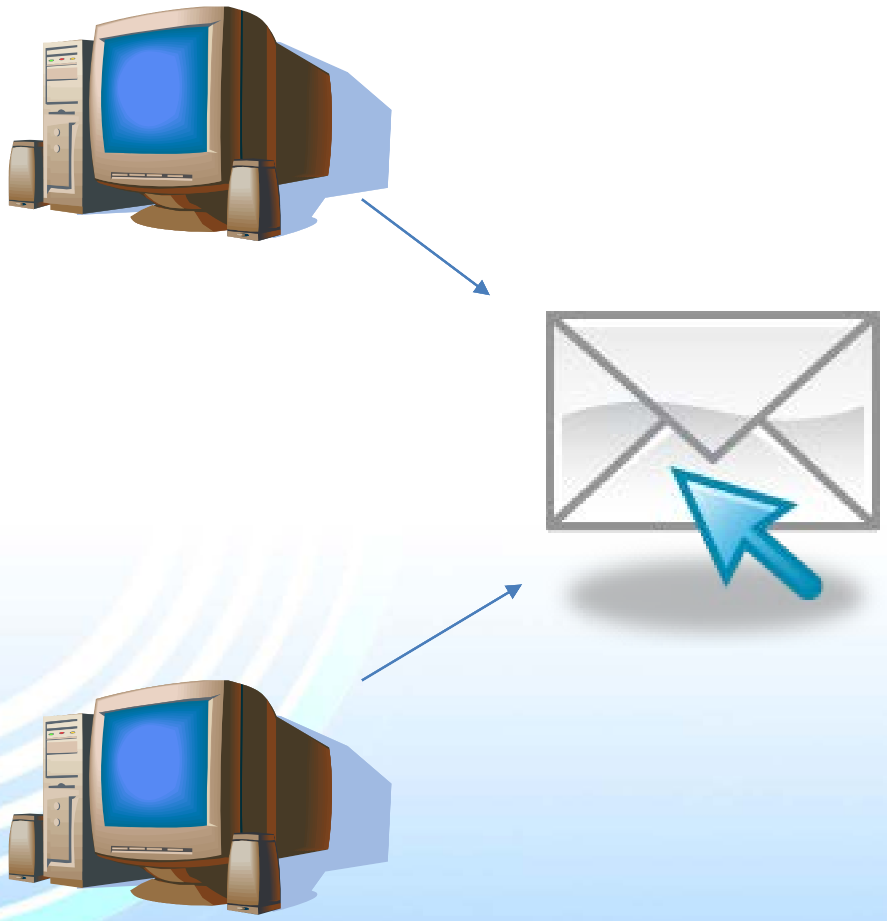


```
<SOAP-ENV:Envelope
     xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/"
     SOAP-ENV:encodingStyle=
"http://schemas.xmlsoap.org/soap/encoding/"/>
            <SOAP-ENV:Header>
                <t:Transaction
                    xmlns:t="some-URI"
                    SOAP-ENV:mustUnderstand="1">
                        5
                </t:Transaction>
            </SOAP-ENV:Header>

            <SOAP-ENV:Body>
                <m:GetLastTradePrice xmlns:m="Some-URI">
                    <symbol>DEF</symbol>
                </m:GetLastTradePrice>
            </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Example of Architecture



```
<soap:Envelope   >
    - namespaces

   <soap:Header>

    - Destination/Roles/Actors
   -  How to get there

   </soap:Header>

   <soap:Body>

     - Data/message/Payload

      <soap:Fault>
      ...
     </ soap:Fault>

</soap:Body>
</soap:Envelope>
```

SOAP Envelope Element

- SOAP message must always have an Envelope element associated with the http://www.w3.org/2001/12/soap-envelope namespace.

SOAP Header Element

- Optional
- SOAP Header element contains application specific information
  - authentication, encryption, buffering/caching, etc

SOAP Body Element

- The required SOAP Body element contains the actual SOAP message intended for the ultimate endpoint of the message
- SOAP defines one element inside the Body element in the default namespace (i.e. Fault)
  - This is the SOAP Fault element, which is used to indicate error messages.

SOAP Body Element (cont.)

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
        encoding">
<soap:Body>
<m:GetPrice xmlns:m="http://www.w3schools.com/prices">
        <m:Item>Apples</m:Item>
</m:GetPrice>
</soap:Body>
</soap:Envelope>
```

The example above requests the price of apples

SOAP Body Element (cont.)

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope”
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body>
<m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
       <m:Price>1.90</m:Price>
    </m:GetPriceResponse>
</soap:Body>
</soap:Envelope>
```

The example is the response from web service.

SOAP Fault Element

- Optional
- Used to hold error/status information for a SOAP message.
- If a Fault element is present, it must appear as a child element of the Body element.
- The SOAP Fault element has the following sub elements:
  - <faultcode> - A code for identifying the fault
  - <faultstring> - A human readable explanation of the fault
  - <faultactor> - Information about who caused the fault to happen
  - <detail> - Holds application specific error information related to the Body element

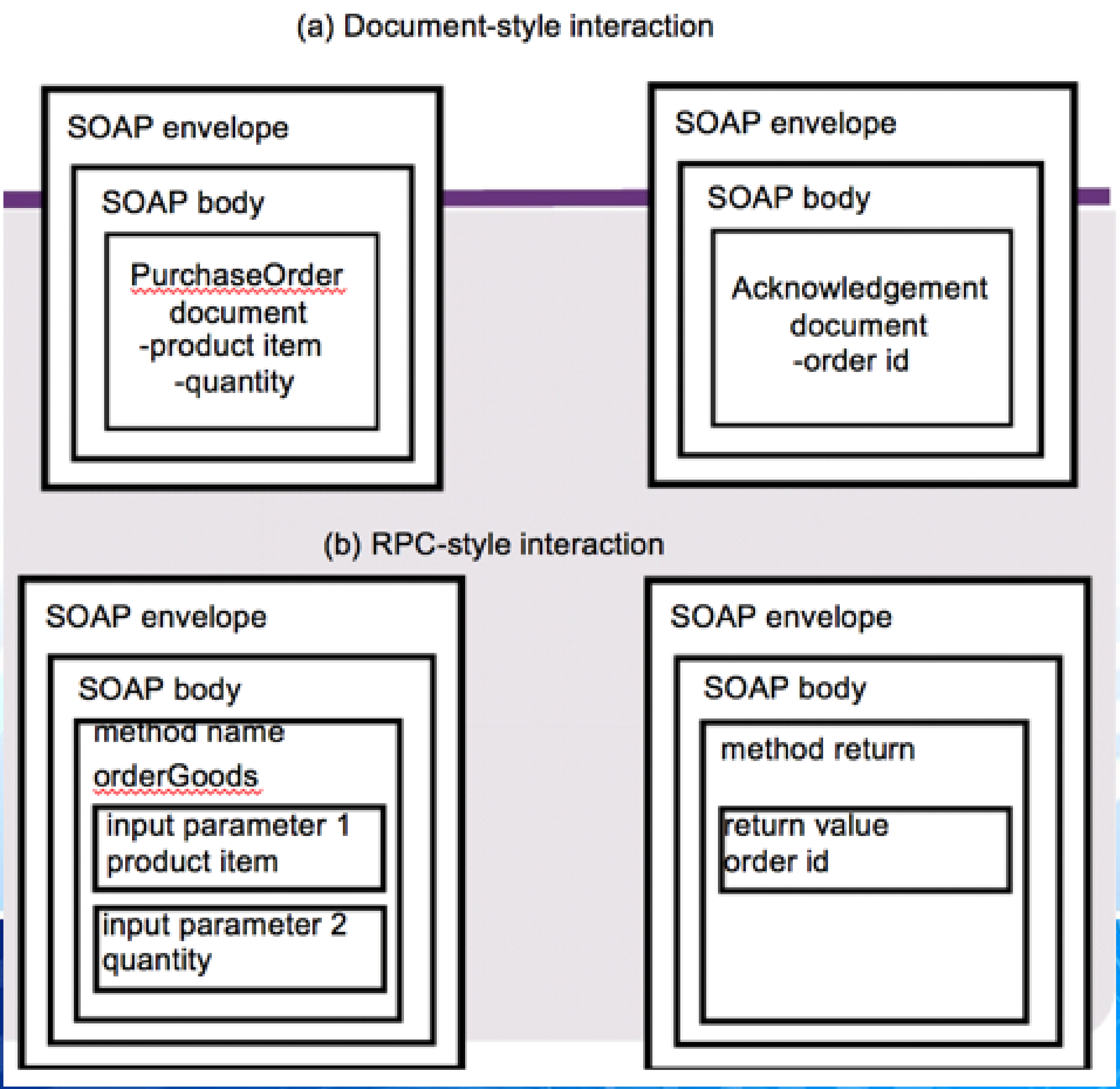
SOAP Fault Element (cont.)

- The **faultcode** values
- **VersionMismatch** - Found an invalid namespace for the SOAP Envelope Element
- **MustUnderstand** - An immediate child element of the Header element, with the mustUnderstand attribute set to "1“ was not understood
- **Client** - The message was incorrectly formed or contained incorrect information
- **Server** - There was a problem with the server so the message could not proceed

SOAP Styles

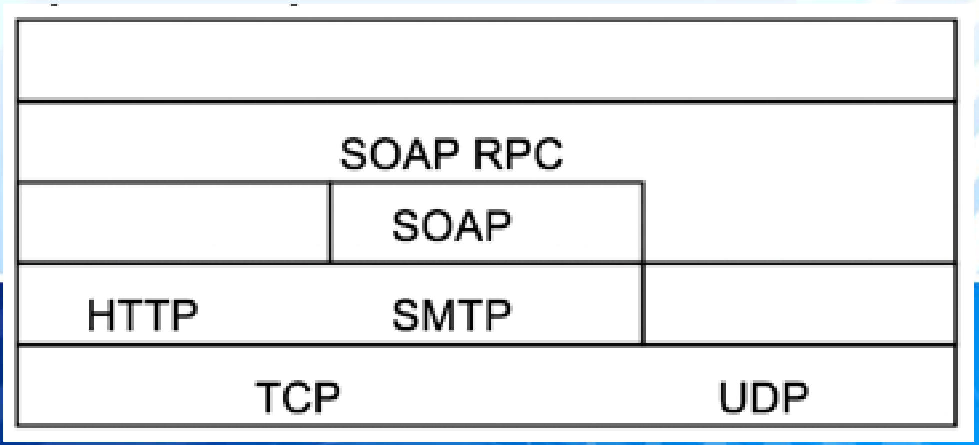
- Two interaction styles: **document and RPC styles**
- **Document style**:
  - the body simply contains an XML document
  - two interacting applications agree on the structure/format of documents exchanged between them
    - E.g. A client ordering goods from one supplier creates a PurchaseOrder document as a SOAP message (i.e. items and their quantities)
    - Supplier sends an Acknowledgement document containing order Id for confirmation
- **RPC style**: (Easy for developer)
  - two sides agreeing on the RPC method signature instead of document structure
  - The request message contains the actual call including the name of the procedure and input parameters
  - The response message contains the results and output parameters
  - XML is used to serialize the parameters

SOAP Styles


(a) Document-style interaction
SOAP envelope
SOAP body
PurchaseOrder document -product item -quantity
SOAP envelope
SOAP body
Acknowledgement document -order id
(b) RPC-style interaction
SOAP envelope
SOAP body
method name
orderGoods
input parameter 1 product item
input parameter 2 quantity
SOAP envelope
SOAP body
method return
return value order id

SOAP protocol binding framework

- SOAP protocol binding framework
- A binding of SOAP to a transport protocol is a description of how a SOAP message is to be sent using that transport protocol
- The SOAP binding framework expresses guidelines for specifying a binding to a particular protocol



HTTP Overview

- HTTP communicates over TCP/IP.
- The client will send the following HTTP message to the server as a request:

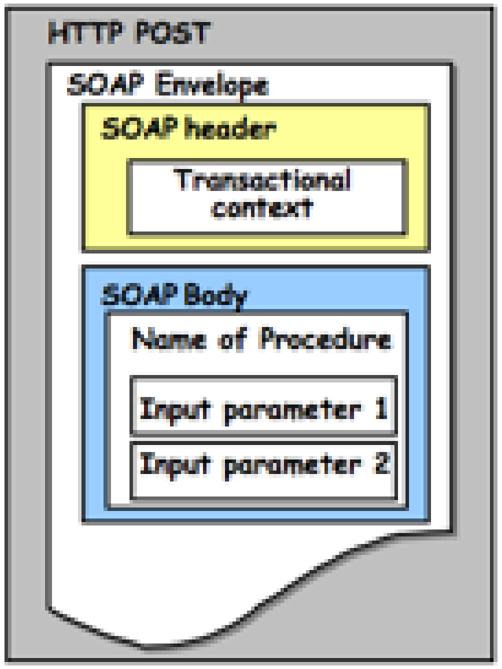
```
POST /item HTTP/1.1
Host: 189.123.345.239
Content-Type: text/plain
Content-Length: 200
```

- Then The server processes the request and sends an HTTP response back to the client.

```
200 OK
Content-Type: text/plain
Content-Length: 200
```

SOAP HTTP Binding

- SOAP messages are enclosed in the payload/body of an HTTP request or response
- A SOAP request could be an HTTP POST or an HTTP GET request
- The HTTP POST request specifies at least two HTTP headers: **Content-Type** and **Content-Length**.



SOAP HTTP Binding (cont.)

- **Content-Type** header for a SOAP request and response defines
  - The MIME type for the message
  - The character encoding (optional) used for the XML body of the request or response.
- Syntax:

```
Content-Type: MIMEType;
charset=character-encoding
```

- Example:

```
POST /item HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
```

SOAP HTTP Binding (cont.)

- **Content-Length** header for a SOAP request and response specifies the number of bytes in the body of the request or response.
- Syntax: `Content-Length: bytes`
- Example:

```
POST /item HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 250
```

SOAP HTTP Example - Request

```
POST /doubleAnInteger HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 200

<SOAP-ENV:Envelope
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
        <SOAP-ENV:Body>
                <ns1:doubleAnInteger
                xmlns:ns1="urn:MySoapServices">
                        <param1 xsi:type="xsd:int">123</param1>
                </ns1:doubleAnInteger>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP HTTP Example - Response

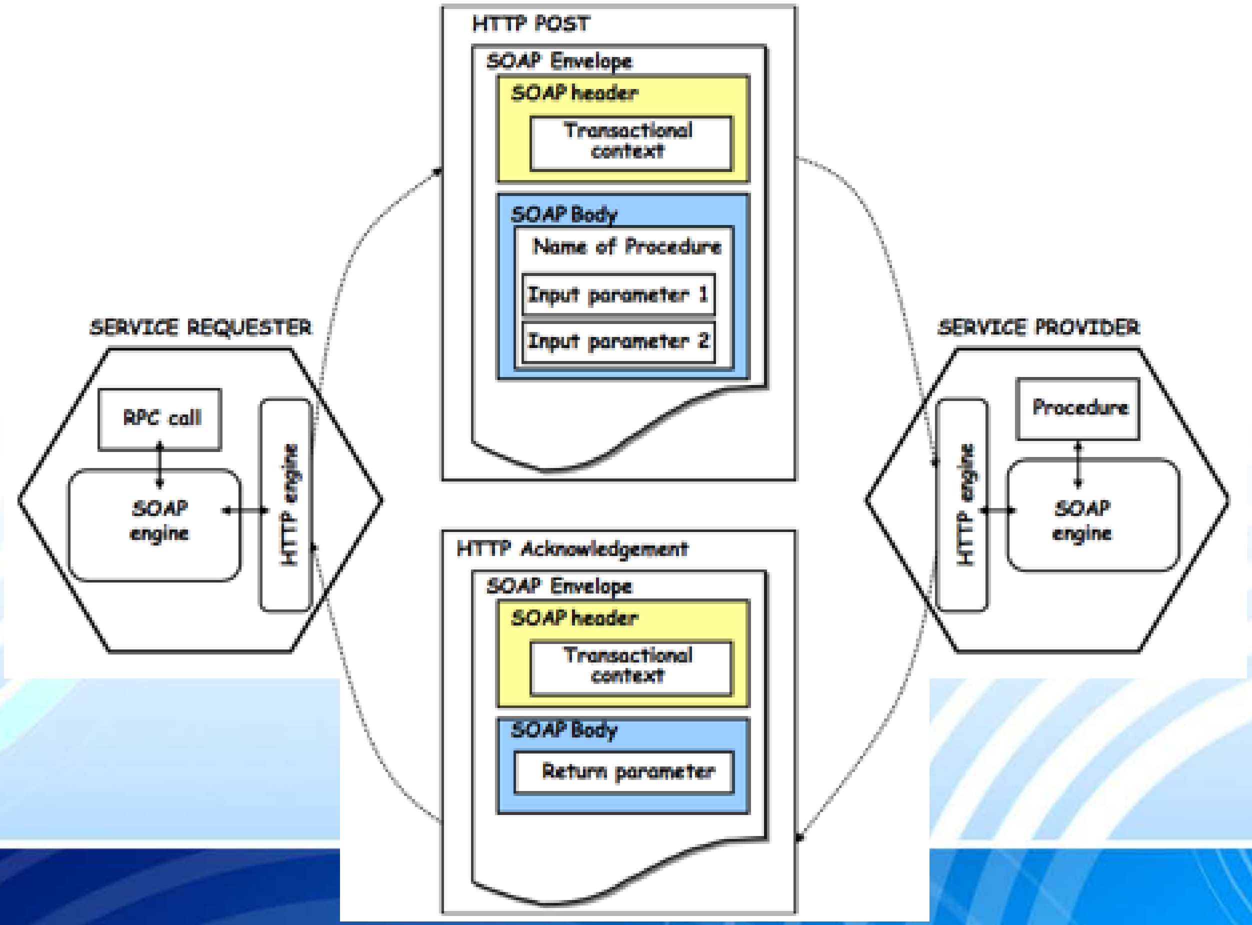
```
POST /doubleAnInteger HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 200

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
       <SOAP-ENV:Body>
              <ns1:doubleAnIntegerResponse
              xmlns:ns1="urn:MySoapServices"
              SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

                     <return xsi:type="xsd:int">246</return>

              </ns1:doubleAnIntegerResponse>
       </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP with HTTP



Reference

- https://www.vs.inf.ethz.ch/edu/WS0405/VS/VS-050124.pdf
- https://www.tutorialspoint.com/uddi/uddi_overview.htm