Agenda

- Last time
  - Midterm
- Today
  - Web Services
  - Assignment #3 out (due in 1 week)
- Next time (Tues Mar 27)
  - Finish naming: DNS
  - P2P
  - Midterm, PA#2 back
- Make-up class next week!
  - Note: we probably need one more, for Thurs Apr 26

Schedule

<table>
<thead>
<tr>
<th></th>
<th>Sun</th>
<th>Tues</th>
<th>Thurs</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midterm</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2P</td>
<td></td>
<td></td>
<td>27</td>
<td>22</td>
</tr>
<tr>
<td>DNS</td>
<td></td>
<td></td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Coordination/agreement</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transactions</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transactions</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Student presentations</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA#5 due</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The problem (arguably)

- COM/DCOM, CORBA, RMI do not easily play with each other
- None of these “play well” over HTTP
- Can we come up with a set of standards and best practices by which to perform “distributed computing over the Internet”?
  - Proposed solution: Web services

Today’s keywords/Themes

- Religion
- Philosophy
- Marketing
- FUD
- Composability
- Reuse
- Extensibility

"Things should be made as simple as possible, but no simpler." – Albert Einstein

What is a Web service?

- Definition from W3C (“Web Services Architecture Requirements”, W3C Working Draft 29 April 2002)
  - A Web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols

- Features/Goals:
  - Semantically encapsulate discrete functionality
  - Loosely coupled, reusable components
  - Programmatically accessible

- What is:
  - W3C?
  - HTTP?
  - URI?
  - XML?

Internet Technologies

Web Standards

- Internet Engineering Task Force (IETF)
  - http://www.ietf.org/
  - Founded 1986
  - Request For Comments (RFC) at http://www.ietf.org/rfc.html
- World Wide Web Consortium (W3C)
  - http://www.w3.org
  - Founded 1994 by Tim Berners-Lee
  - Publishes technical reports and recommendations
- Many more: OASIS, ECMA, etc.
**W3C**

- World Wide Web Consortium
- Founded by Tim Berners-Lee in 1994; at MIT in collaboration with CERN
- **Goals**
  - Universal Access
  - Semantic Web
  - Web of Trust
- **Roles**
  - Vision future of WWW
  - Design
  - Standardization
- **Principles**
  - Interoperability
  - Evolution
  - Decentralization
- **Sample standards:** HTML, HTTP, SOAP, URI/URL, XML encryption, XML Signature, etc.

**Evolution of WWW**

**Process**

- **WD:** A document that W3C has published for review by the community
- **CR:** A document that satisfies the Working Group’s technical requirements
- **PR:** A mature technical report
- **REC:** Specification that is endorsed by W3C members and the Director

**Pros/Cons of Standards**

- Who benefits from standards?
- Why would a company want to make a standard?
- Why would a company NOT want to make a standard?
- Many companies involved in Web services
  - Sun, Microsoft, IBM, etc.

**HTTP**

- **HTTP:** request/response over TCP
- Client: typically a browser; typical servers (typically on port 80): IIS, Apache
- **HTTP is “stateless”** (after the server replies, it closes the connection/socket)
  - Stateful: “give me the next one”
  - Stateless: “give me item #3”
  - Stateful vs. Stateless: implications on crashing, load-sharing/mobility, server “complexity”, message size
- **Typical request:**
  - “GET /path/to/file/index.html HTTP/1.0”
  - “GET” = “give me this Resource” (URL)
  - Other verbs: POST (sent to server for server to “process”) and HEAD (just return the headers of GET)

**HTTP Request**

<table>
<thead>
<tr>
<th>Method</th>
<th>File</th>
<th>HTTP version</th>
<th>Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/default.asp</td>
<td>HTTP/1.0</td>
<td>Accept: image/gif, image/x-bitmap, image/jpeg, <em>/</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Accept-Language: en</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>User-Agent: Mozilla/1.22 (compatible: MSIE 2.0; Windows 95)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Connection: Keep-Alive</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If-Modified-Since: Sun, 17-Ago-96 04:32:56 GMT</td>
</tr>
</tbody>
</table>

Data – none for GET
HTTP Response

HTTP/1.0 200 OK
Date: Sun, 21 Apr 1996 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/5.0
Connection: keep-alive
Content-Type: text/html
Last-Modified: Thu, 18 Apr 1996 17:39:05 GMT
Content-Length: 2543

<HTML> Some data... blah, blah, blah </HTML>
Basic XML

```xml
<?xml version="1.0" standalone="yes"?>
<conversation>
  <greeting>Hello, world!</greeting>
  <response>Hello to you too!</response>
</conversation>
```

XML with XML Schema

- Checking the data content of elements as well as the markup itself
- XML Schema written in XML itself

```xml
<xsd:simpleType name="myInteger">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="10000"/>
    <xsd:maxInclusive value="99999"/>
  </xsd:restriction>
</xsd:simpleType>
```

XML Namespaces

- A collection of element and attribute names identified by a Uniform Resource Identifier reference

```xml
<Department>
  <Name>DVS1</Name>
  <addr:Address xmlns:addr="http://www.tu-darmstadt.de/ito/addresses">
    <addr:Street>Wilhelminenstr. 7</addr:Street>
    <addr:City>Darmstadt</addr:City>
    <addr:State>Hessen</addr:State>
    <addr:Country>Germany</addr:Country>
  <serv:Server xmlns:serv="http://www.tu-darmstadt.de/ito/servers">
    <serv:Name>OurWebServer</serv:Name>
    <serv:Address>123.45.67.8</serv:Address>
</Department>
```

HTML vs. XML

- HTML tags: presentation, generic document structure
- XML tags

```xml
<h1>Bibliography</h1>
<p><i>Foundations of DBs</i>, Abiteboul, Hull, Vianu
  <br>Addison-Wesley, 1995</p>
<p><i>Logics for DBs and ISs</i>, Chomicki, Saake, eds.
  <br>Keuser, 1998</p>
<bibliography>
  <book><title>Foundations of DBs</title>
    <author>Abiteboul</author>
    <author>Hull</author>
    <publisher>Addison-Wesley</publisher>
  </book>
  ...<book><book><editor>Chomicki</editor><book>
  ...<book>
</bibliography>
```

XML as a Self-Describing Data Exchange Format

- Can be easily "understood" by our friend
- Can be parsed easily
- Contains its own structure (sparse tree) in the data
  - Requires the application programmer to rediscover schema and content semantics (to which extent??)
- May include an explicit schema description (e.g., DTD, XML schema)
  - Meta-language: definition of a language w.r.t. which it is valid
- Allows separation of marked-up content from presentation (=style sheets)
- Many tools (and many more to come) = (re)use code: parsers, validators, query languages, storage, ...
- Standards (good for interoperability, integration, etc.):
  - Generic standards (XML, DTDs, XML Schema, XPath, ...)
  - Community/industry standards (=specific markup languages)

Simple Object Access Protocol (SOAP)

- For information exchange in a distributed environment
- Message format based on XML
- Can be combined with various transport protocols
- Originally developed by Microsoft
- SOAP Version 1.2
- Don’t get confused:
  - XML-RPC:
    - Subset of SOAP
    - Remote procedure calling using HTTP as the transport and XML as the encoding
  - JAX-RPC
  - Java API for XML-based remote procedure calls (can use SOAP)
**Composition of a SOAP Envelope**

```xml
<env: Envelope xmlns:env="http://www.w3.org/2001/09/soap-envelope/"
  env:encodingStyle="http://www.w3.org/2001/09/soap-encoding/">
  <env:Header>
    <t:Transaction xmlns:t="some-URI">
      env:mustUnderstand="1"
    </t:Transaction>
  </env:Header>
  <env:Body>
    <m:GetLastTradePrice xmlns:m="some-URI">
      <symbol>DEF</symbol>
    </m:GetLastTradePrice>
  </env:Body>
</env:Envelope>
```

**A SOAP Envelope**

**SOAP Request and Response**

**Request (partial):**
```xml
<env:Body>
  <m:GetLastTradePrice xmlns:m="some-URI">
    <symbol>DEF</symbol>
  </m:GetLastTradePrice>
</env:Body>
```

**Response (partial):**
```xml
<env:Body>
  <m:GetLastTradePriceResponse xmlns:m="some-URI">
    <price>22.50</price>
  </m:GetLastTradePriceResponse>
</env:Body>
```

**SOAP Message in an HTTP Request**

POST /StockQuote HTTP/1.1
Host: www.example.org
Content-Type: text/xml; charset="utf-8"
SOAPAction: "http://example.org/2001/06/quotes"

```xml
<env:Envelope xmlns:env="http://www.w3.org/2001/09/soap-envelope/">
  <env:Body>
    <m:GetLastTradePrice xmlns:m="http://example.org/2001/06/quotes">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </env:Body>
</env:Envelope>
```

**SOAP Message in an HTTP Response**

POST /StockQuote HTTP/1.1
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"

```xml
<env:Envelope xmlns:env="http://www.w3.org/2001/09/soap-envelope/">
  <env:Body>
    <m:GetLastTradePriceResponse xmlns:m="http://example.org/2001/06/quotes">
      <Price>34.5</Price>
    </m:GetLastTradePriceResponse>
  </env:Body>
</env:Envelope>
```

**WSDL**

- WSDL is an XML-based language used to define Web Services and describe how to access them.
- WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.
**REST**

- **SOAP Web services**
  - SOAP + WSDL
- **REST Web Services**
  - Roy Fielding
  - “Resource” = anything that has a URI, has one or more “representations”
  - Interfaces are limited to HTTP
    - HTTP GET – get the representation of a resource
    - HTTP DELETE
    - HTTP POST
    - HTTP PUT

---

**Other specs/approaches**

- UDDI
- WS-Security
- BPEL
- WS-Addressing
- WS-AtomicTransaction
- WS-Reliability / WS-ReliableMessaging

---

**Service-Oriented Architectures (SOAs)**

- “Important compliment to O-O”
- “A service is simply a program that one interacts with via message exchanges”
- Microsoft’s “4 tenets” of SOAs:
  1. Boundaries are explicit
  2. Services are autonomous
  3. Services share schema and contract, not class
  4. Service compatibility is determined based on policy

---

**Assignment #3**

- “tutorial”-like, approx. time: 2-3 hours
- Build a Web Service via VS.NET
  - WCF (code name “Indigo”); NOT using IIS
- Consume a Web Service.
- Look at SOAP moving back and forth.
- Investigate interoperability (AXIS, gSOAP)