Lecture 6

XML-RPC
Announcements

• Lab 1 due back today

• Lab 2 (XML-RPC) out today, due back October 7th
(Review) Remote Procedure Calls

- A procedure-call like request-reply abstraction built on top of lower-level networking protocols

**Diagram:**
- Client program
  - Stubs (proxies)
  - rpcgen
  - RPC Interface
  - rpcgen
  - nameserver
  - Transport: UDP, TCP
- Server program
  - Stubs (proxies)
  - rpcgen
  - nameserver
  - RPC Interface
  - Transport: UDP, TCP

**Legend:**
- Client
- Server
- Standard wire-format (e.g., XDR, NDR)
XML-RPC

- A procedure-call like request-reply abstraction built on top of lower-level networking protocols

Client program

RPC Interface

Stubs (proxies)

rpcgen

Server program

RPC Runtime

Standard wire-format (e.g., XDR, NDR)

HTTP

XML

Transport: UDP, TCP

Client

Server

URI-based naming

nameserver

Transport: UDP, TCP

HTTP
XML

- XML is a standard for describing \textit{structured} documents
  - Uses \textit{tags} to define structure: \texttt{<tag> … </tag>} demarcates an \textit{element}
    - Tags have no predefined semantics …
    - … except when document refers to a specific namespace
  - Elements can have \textit{attributes}, which are encoded as name-value pairs

- A well-formed XML document corresponds to an element \textit{tree}

```xml
<?xml version="1.0"?>
<methodCall>
    <methodName>SumAndDifference</methodName>
    <params>
        <param><value><i4>40</i4></value></param>
        <param><value><i4>10</i4></value></param>
    </params>
</methodCall>
```
HTTP

HTTP: Hyper Text Transfer Protocol

- An application-layer protocol connecting web clients and servers
  - HTTP is used to transfer resources between the two
    - Example of a resource: file, dynamically-generated output, ...
  - Resources referred to by URIs (Universal Resource Identifiers)
    protocol://server-name:port-number/resource-path
    - Resource paths use hierarchical naming (similar to files in a filesystem)

- HTTP transactions
  Initial-line – Header – Body (optional)
  - Many types
    - Client issues a GET, Server responds with header and body (resource)
    - Client issues a HEAD, Server responds with header
    - Client issues a POST (with optional parameters in body),
      Server processes this request (e.g., using a CGI script) and returns output
    - ...

9/22/2003
XML-RPC “Wire Format”

- Scalar values
  - Represented by a `<value><type> ... </type></value>` block

<table>
<thead>
<tr>
<th>type tag</th>
<th>.NET Type</th>
<th>Description</th>
<th>Example of value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;i4&gt; or &lt;int&gt;</td>
<td>System.Int32</td>
<td>4-byte signed integer</td>
<td>-12</td>
</tr>
<tr>
<td>&lt;boolean&gt;</td>
<td>System.Boolean</td>
<td>0 (false) or 1 (true)</td>
<td>1</td>
</tr>
<tr>
<td>&lt;string&gt;</td>
<td>System.String</td>
<td>string</td>
<td>Hello world</td>
</tr>
<tr>
<td>&lt;double&gt;</td>
<td>System.Double</td>
<td>double-precision signed floating point number</td>
<td>-12.215</td>
</tr>
<tr>
<td>&lt;dateTime.iso8601&gt;</td>
<td>System.DateTime</td>
<td>date/time</td>
<td>19980717T14:08:55</td>
</tr>
<tr>
<td>&lt;base64&gt;</td>
<td>System.Byte[]</td>
<td>Base-64 encoded binary</td>
<td>eW91IGNhbid0I</td>
</tr>
</tbody>
</table>
XML-RPC “Wire Format” (cont’d)

• Structures
  – Represented as a set of `<member>`s
  – Each member contains a `<name>` and a `<value>`

  <struct>
    <member>
      <name>lowerBound</name>
      <value><i4>18</i4></value>
    </member>
    <member>
      <name>upperBound</name>
      <value><i4>139</i4></value>
    </member>
  </struct>

• Arrays
  – A single `<data>` element, which contains any number of `<value>` elements

  <array>
    <data>
      <value><i4>12</i4></value>
      <value><i4>18</i4></value>
      <value><string>Egypt</string></value>
      <value><boolean>0</boolean></value>
      <value><i4>-31</i4></value>
    </data>
  </array>
XML-RPC Request

- **HTTP POST message**
  - **URI** interpreted in an implementation-specific fashion
  - **Method name** passed to the server program

```xml
POST /VSDev/Public/vijayk/XMLRPC/sumAndDiff.rem HTTP/1.1
Content-Type: text/xml
User-Agent: XML-RPC.NET
Content-Length: 278
Expect: 100-continue
Connection: Keep-Alive
Host: localhost:8080

<?xml version="1.0"?>
<methodCall>
  <methodName>SumAndDifference</methodName>
  <params>
    <param><value><i4>40</i4></value></param>
    <param><value><i4>10</i4></value></param>
  </params>
</methodCall>
```
XML-RPC Response

- HTTP Response
  - Lower-level error returned as an HTTP error code
  - Application-level errors returned as a `<fault>` element (next slide)

HTTP/1.1 200 OK
Date: Mon, 22 Sep 2003 21:52:34 GMT
Server: Microsoft-IIS/6.0
Content-Type: text/xml
Content-Length: 467

```xml
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param><value><struct>
      <member><name>sum</name><value><i4>50</i4></value></member>
      <member><name>diff</name><value><i4>30</i4></value></member>
    </struct></value></param>
  </params>
</methodResponse>
```
XML-RPC Fault Handling

- Another kind of a MethodResponse

```xml
<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value><struct>
      <member>
        <name>faultCode</name>
        <value><i4>500</i4></value>
      </member>
      <member>
        <name>faultString</name>
        <value><string>Arg `a' out of range</string></value>
      </member>
    </struct></value>
  </fault>
</methodResponse>
```
XML-RPC: Discussion

- Very simple specification
  - Large number of implementations for every conceivable language
  - No surprises during integration
- XML wire-format
  - Human readable
  - Somewhat verbose
    - Binary encoding possible, but additional layer of specification
- HTTP transport
  - Widespread use
  - Firewalls already permit HTTP traffic, so no reconfiguration required

- Main drawback
  - Leaves a lot unspecified
    - Mapping of URL to server handler
    - State management at the server, …
XML-RPC Example

[Code walkthrough of the SumAndDiff example from Lab2]