IRIS Web Services Workshop II

Session 4
Thursday PM, September 22

Dennis M. Sosnoski

Outline

- Web service attachments
  - Why and how
  - The many "standards"
  - Attachments in Axis
- Web service performance
  - Comparing frameworks
  - Implications

SOAP Attachments

- SOAP Body content issues:
  - Character set restricted in XML
  - XML must be well formed
  - Binary data must be encoded with heavy overhead
- Attachments let you avoid these issues:
  - ***S***OAP ***w***ith ***A***ttachments (SwA, MIME-based)
  - ***D***irect ***I***nternet ***M***essage ***E***xchange (DIME)
  - ***M***essage ***T***ransmission ***O***ptimization ***M***echanism (MTOM)

SwA

- Early proposal from Microsoft for standard
  - Based on existing MIME standard
  - Attachments follow actual SOAP Envelope
  - Use boundaries string, content type, for each block
    - But data transmitted as raw bytes
- Basis for WS-I Attachment Profile 1.0
- But not supported by Microsoft
  - Usefulness thereby highly limited

DIME

- More recent proposal from Microsoft for standard
  - Uses binary header with data length
  - Attachments again follow SOAP envelope
- No standard support for Java
  - Individual implementations may include (e.g., Axis)
- Microsoft supports in advanced services pack

MTOM

- Protocol layering is such a 20th Century idea
  - Make everything part of the XML Infoset
  - Let the code figure out how best to serialize
  - Advantage is that application can ignore attachments
  - Disadvantage is that application has no control over attachments
- Good idea or bad, it's the coming thing

WS-I Basic Profile 1.1

- Basic Profile 1.0a addressed SOAP/WSDL
- Basic Profile 1.1 addresses attachments:
  - Referenced Attachments Profile 1.0 uses SwA
  - Defines a special type for attachment references:

```
<xsd:schema
  targetNamespace="http://ws-i.org/profiles/basic/1.1/xsd"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:simpleType name="swaRef">
    <xsd:restriction base="xsd:anyURI"/>
  </xsd:simpleType>
</xsd:schema>
```

  - But won't be supported by Microsoft, so largely irrelevant

SAAJ

- ***S***OAP with ***A***ttachments ***A***PI for ***J***ava
  - Originally part of JAXM, then split off
  - Designed for SwA style of attachments
  - Builds on JavaMail MIME handling
  - Current version includes WS-I BP 1.1 and WS-I AP 1.0 support
- JAX-RPC uses directly, or behind the scenes
- Axis adds DIME support option
  - But need *activation.jar* and *mail.jar* for this!

Using attachments in Axis

- Attachments have to be defined in WSDL

```
<wsdl:definitions xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" ...>
 ...
 <wsdl:message name="binaryResponse">
  <wsdl:part type="xsd:hexBinary" name="serializedResponse"/>
 </wsdl:message>
 ...
 <wsdl:binding name="seismicSoapBinding" type="impl:Seismic">
  ...
  <wsdl:operation name="findQuakesAttachment">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="findQuakesAttachmentRequest">
    <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="findQuakesAttachmentResponse">
    <mime:multipartRelated>
     <mime:part>
      <wsdlsoap:body use="literal"/>
     </mime:part>
     <mime:part>
      <mime:content part="serializedResponse" type="application/octet-stream"/>
     </mime:part>
    </mime:multipartRelated>
   </wsdl:output>
  </wsdl:operation>
```


Code generation

- Code generation based on mime-type
  - Various image types treated as Java images
  - Raw bytes treated as `org.apache.axis.attachments.OctetStream`

```
public OctetStream findQuakesAttachment(Calendar mndate, Calendar mxdate,
    Float mnlat, Float mxlat, Float mnlng, Float mxlng) { ... }
```

- Conversion to and from attachment form done behind the scenes

Using flexible attachments

- Attachments can also use Axis hooks directly
  - On server, use org.apache.axis.`MessageContext`

```
MessageContext ctx = MessageContext.getCurrentContext();
Message rsp = ctx.getResponseMessage();
byte[] byts = ...;
AttachmentPart attch =
    (AttachmentPart)rsp.createAttachmentPart();
attch.setContent(new ByteArrayInputStream(byts),
    "application/octet-stream");
rsp.addAttachmentPart(attch);
```

  - On client, use Axis stub implementation class

```
Object[] attchs =
    ((org.apache.axis.client.Stub)stub).getAttachments();
AttachmentPart part = (AttachmentPart)attchs[0];
DataHandler dh = part.getDataHandler();
InputStream is = dh.getInputStream();
```


Converting data

- Converting Java objects to binary form
  - Server uses `java.io.ObjectOutputStream` to serialize:

```
QuakeSet[] set = ...
ByteArrayOutputStream bos = new ByteArrayOutputStream();
ObjectOutputStream oos = new ObjectOutputStream(bos);
oos.writeObject(sets);
byte[] byts = bos.toByteArray();
```

  - Client uses `java.io.ObjectInputStream` to restore:

```
byte[] byts = ...
ByteArrayInputStream bis = new ByteArrayInputStream(byts);
ObjectInputStream ois = new ObjectInputStream(bis);
QuakeSet[] sets = (QuakeSet[])ois.readObject();
```

Attachments configuration

- Copy *mail.jar* and *activation.jar* to:
  - The *lib* directory of your Axis installation
  - The */WEB-INF/lib* directory of your Axis servlet installation under Tomcat (*webapps/axis/WEB-INF/lib*)

Debugging service

- With Tomcat stopped, edit the file webapps/axis/WEB-INF/server-config.wsdd and add the line (under globalConfiguration):

```
<parameter name="axis.development.system" value="true"/>
```


Assignment 1

- Modify Seismic service code from yesterday
  - Add attachment method `findQuakesAttachment`
    - Same parameters as `findQuakes`, but returning Java-serialized data in attachment
      - Modify WSDL to add method returning attachment
      - Generate code, modify implementation classes to match (using code samples from slides)
  - Try full range query (passing nulls) both ways, check if one is faster

Attachments summary

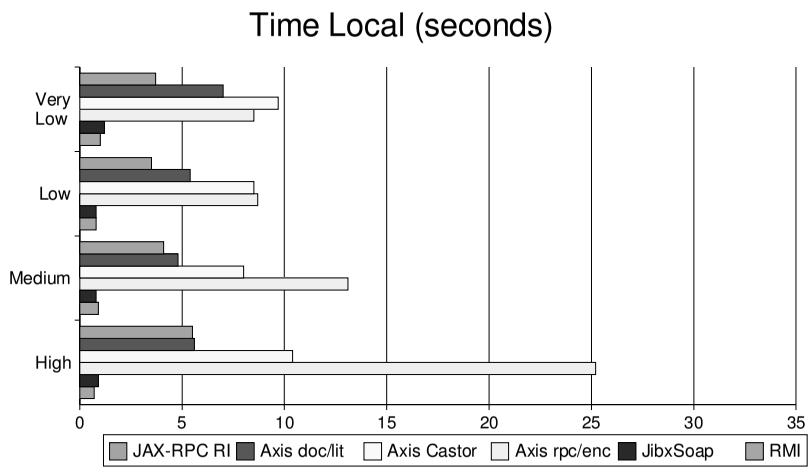
- No one standard
  - Not all SOAP implementations support **any** form
  - Some support SwA, some DIME, some both
  - Standard SAAJ support is SwA only
- Not yet usable for general-purpose interfaces
- Use to meet requirements when you can
  - But must be able to restrict client pool for now

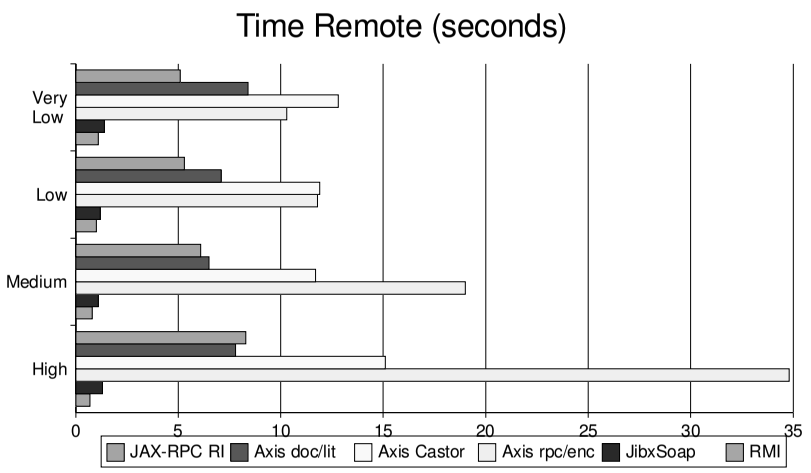
Web services performance

- Earthquake information web service
  - Query by date, location, magnitude, etc.
  - Returns results sorted by area, regions as needed
- Variations tested:
  - JAX-RPC RI – doc/lit WSDL to Java code
  - Axis implementation of JAX-RPC – Java to rpc/enc WSDL to Java, doc/lit WSDL to Java, and Castor data binding
  - JibxSoap – direct data binding
  - Java RMI for "native" timing comparison

Performance test

- Use pseudo-random sequence for queries
- Tune request ranges for different densities:
  - Very low – 400 queries with just 88 matching quakes
  - Low – 100 queries with 853 matching quakes
  - Medium – 20 queries with 3052 matching quakes
  - High – 10 queries with 6155 matching quakes
- Verify number of quakes returned for each, etc.
- Local to one system vs. over local network





Why the differences?

- Axis rpc/enc much bulkier than doc/lit format
  - Uses only elements, not attributes
  - Includes xsi:types by default, repeated namespaces
- JAX-RPC RI better tuned than Axis
- JiBX data binding much faster than others
  - JibxSoap almost as fast as Java RMI
  - On a par with "Fast Web Services" from Sun
- Commercial products probably in between
  - Glue and WASP declined benchmark permission

Slower services

- Web services generally not the fastest choice
  - RMI faster for Java-Java applications
  - CORBA for cross-language (major) applications
  - Custom protocols for any application
- But offer advantages
  - Easy to view / capture message exchange
  - Cross-language support better than CORBA
  - Ties in to XML tooling and mindset

Service performance

- Three major factors in performance:
  - Transport inefficiencies (HTTP rather than socket)
  - XML inefficiencies (more bulk, conversion and parsing overhead)
  - Framework inefficiences (poor performance designs, SOAP overhead, etc.)
- Can address all these issues in your usage

Transport issues

- HTTP more overhead than TCP/IP socket
  - HTTP builds on top of socket connection
  - Connection establishment cost relatively high, but HTTP does not maintain for long
- Effect is per-call overhead

Transport alternatives

- Can use bare socket transport
  - But very few frameworks support this now
  - Looses advantages of HTTP common transport
    - Firewalls need to be reconfigured
    - Security weakened (at least in perception)

Transport implications

- Call granularity issue, just as with EJBs
- Design interface to reduce number of calls
  - Structure operations to accomplish as much as reasonable
  - Use multiple submission for repeated operations
  - Work through use cases to see number of calls required

XML issues

- XML much bulkier than binary data
- XML parsing fairly expensive
  - More so than regular text processing, because of XML requirements
- Conversions to and from binary fairly expensive

XML alternatives

- Define binary representation for XML?
  - Reduce message size, parsing/converting costs
  - Allow interchange, just as with text XML
- Good benefits, but no consensus yet

XML implications

- Overhead roughly proportional to size
  - Reducing amount of data shipped can help
    - But often contrary to proper interface granularity
    - Make portions optional as a way to provide trade-offs?
  - Structuring XML can help
    - doc/lit generally more concise than rpc/enc
    - Can design doc/lit XML structure to be even better
- Attachments often best solution

Framework issues

- Many frameworks not performance designed
  - Internal XML conversions to/from DOM
    - Costly in both time and memory usage
    - Sometimes necessary (WS-Security), but often not
  - Inefficient data binding implementations
    - Reflection-based frameworks tend to be slow
    - XMLBeans uses internal parse event store

Framework alternatives 1

- Use lighter-weight protocol (e.g., REST)
  - Simpler protocols generally add less overhead
  - But loses SOAP advantages in enterprise services
    - WS-Security, WS-Addressing, etc.
    - Performing necessary functions for mission-critical applications
- Some frameworks avoid the overhead
  - JibxSoap, perhaps Axis2, commercial choices?

Framework alternatives 2

- Skip the framework and go commando:
  - Interface directly to HTTP
  - Process both SOAP envelope and message in application code
- Potentially best performance for heavy usage applications
- Drawbacks are need to understand SOAP in detail, added coding

Performance summary

- Web service performance a real concern
- Can be addressed by both architecture and design choices
- Future may provide even better approaches