Performance Evaluation of Web Services as Management Technology

G. Pavlou, P. Flegkas, S. Gouveris
Centre for Communication Systems Research
University of Surrey, UK

http://www.ee.surrey.ac.uk/CCSR/Networks/
Web Services as Distributed Object Technology

- Strong analogies to DOTs
  - WSDL similar to CORBA IDL with service inheritance
  - URI similar to CORBA IOR
  - SOAP similar to CORBA GIOP
  - SOAP over HTTP/TCP/IP similar to CORBA IIOP
  - UDDI similar to CORBA Interface Repository and Naming/Trading services
- Difference: loose message-passing coupling between clients-servers
  - Most implementations though take a static coupling approach through stubs but through proprietary APIs
- “On the wire” interoperability only, no standard APIs
- No sophisticated services yet but work under way for transaction and security, notification also required
• Exactly the same issues as using any other DOT
• The proposal presented in the previous talk could be used, we have used for the performance measurements
• Notification facilities through EFD-like services with filtering on event type very easy to realise
  • Proper notification services with filtering on event content (like in OSI-SM EFDs and CORBA) should eventually appear
• For example, the TCP information on a node becomes a service with an advertised URI
  • Methods as described in the previous talk are modelled through operations with messages
• We experimented with three WS implementations
  • Systinet WASP, Apache Axis and gSOAP (for small devices)
• Writing WSDL specs is a pain but all platforms provide converters from Java/C++ object specs
• Apache Axis is not user-friendly, supporting only a low-level SOAP API
• WASP and gSOAP support a CORBA-like stub-based framework and usability is similar to CORBA
  • The APIs are syntactically different but the abstractions are similar, so it is relatively easy to deal with both
  • But, of course, there is no code portability
Evaluation

- We implemented the TCP protocol and connections in CORBA and Web Services and compared the performance of SNMP, CORBA and WS versions

- Hardwired values for TCP counters and connections (40 connections) in order to only assess the infrastructure overhead and achieve repeatability
  - We had to modify a SNMP agent implementation for this

- We used two modelling approaches for TCP connections:
  - Through `getConnNo()`, `getConnInfo()` methods
  - Through separate interfaces and a `getConnRefs()` / `getConnURIs()` method of the TCP interface/endpoint
Environment

- WASP Web Services platform
- Orbacus CORBA platform
- NET-SNMP SNMP platform
- Both C++ and Java implementations, apart from the NET-SNMP agent which was implemented in C
  - Wanted to also see Java to C++ implementation differences
- GNU C/C++ 2.95, Java 2 SE JDK 1.3.1 versions on Linux RedHat 7.3
- Two Celeron 1GHz Linux PCs with 256 Mb RAM connected through a dedicated 100 Mb/s Ethernet
We measured response times for the following:
  - A method returning a single TCP counter - 1Attr/1Method
  - A method returning all the TCP counters – Nattr/1Method
  - Retrieving the whole table in two ways:
    - Through 1 IDL/WSDL method and SNMP GetBulk – NMOs/1Method
    - Through N methods for separate TCP connection interfaces / endpoints in IDL/WSDL and SNMP GetNext – 1MO/1Method

We also measured traffic incurred

And we finally measured the memory footprint for the managed system side for the C/C++ case
WS/SOAP vs SNMP & CORBA: Java Response Times

<table>
<thead>
<tr>
<th></th>
<th>1attr/1method</th>
<th>Nattrs/1method</th>
<th>NMOs/1method</th>
<th>1MO/1method</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP</td>
<td>1</td>
<td>2</td>
<td>8</td>
<td>45</td>
</tr>
<tr>
<td>CORBA</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>97</td>
</tr>
<tr>
<td>WS/SOAP</td>
<td>5</td>
<td>9</td>
<td>40</td>
<td>250</td>
</tr>
</tbody>
</table>
WS/SOAP vs SNMP & CORBA: C++ Response Times

![Graph showing response times for SNMP, CORBA, and WS/SOAP for different methods.]

<table>
<thead>
<tr>
<th>Response Time (msec)</th>
<th>1attr/1method</th>
<th>Nattrs/1method</th>
<th>NMOs/1method</th>
<th>1MO/1method</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP</td>
<td>0.8</td>
<td>1</td>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>CORBA</td>
<td>1.5</td>
<td>1.7</td>
<td>2.5</td>
<td>49</td>
</tr>
<tr>
<td>WS/SOAP</td>
<td>2.5</td>
<td>3.7</td>
<td>25</td>
<td>100</td>
</tr>
</tbody>
</table>
WS/soap vs SNMP & CORBA: Traffic Incurred

<table>
<thead>
<tr>
<th>Traffic (bytes)</th>
<th>1attr/1method</th>
<th>Nattrs/1method</th>
<th>NMOs/1method</th>
<th>1MO/1method</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP</td>
<td>138</td>
<td>413</td>
<td>8160</td>
<td>15917</td>
</tr>
<tr>
<td>CORBA</td>
<td>280</td>
<td>316</td>
<td>2252</td>
<td>24157</td>
</tr>
<tr>
<td>WS</td>
<td>1390</td>
<td>2052</td>
<td>18266</td>
<td>81350</td>
</tr>
</tbody>
</table>
WS/SOAP vs SNMP & CORBA: C++ Memory Footprint

Note: the SNMP managed system memory footprint is for the whole MIB-II while the CORBA / WS ones are only for the TCP MIB-II part
Summary

- Java times roughly twice those of C++
- CORBA is very efficient for retrieving many attributes, the WS time increases because of XML encodings
- CORBA is also very efficient for retrieving a whole table through 1 method in comparison to both SNMP and WS – WS is by far the slowest (by 1 order of magnitude)
- The 1 object per TCP connection in both CORBA/WS results in much slower response times and prohibitive amount of traffic (includes IOR/URI retrieval first)
- CORBA traffic the smallest, WS traffic high but acceptable for simple methods and table retrieval
- Memory footprint highest for CORBA, smallest for SNMP
- In summary, WS performance is not prohibitive