UniS

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

Web Services for Management

- 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

Usability

- 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 L

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**

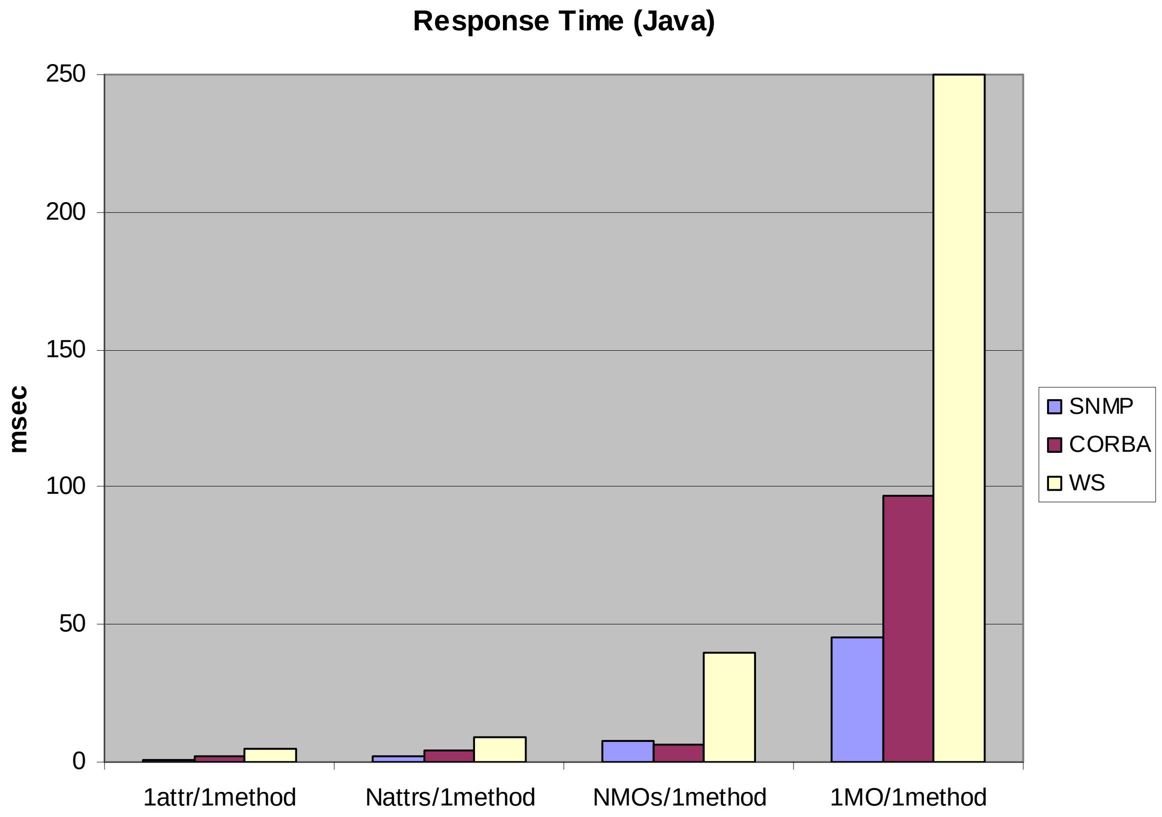
Environment

- **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**

Measurements

- 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

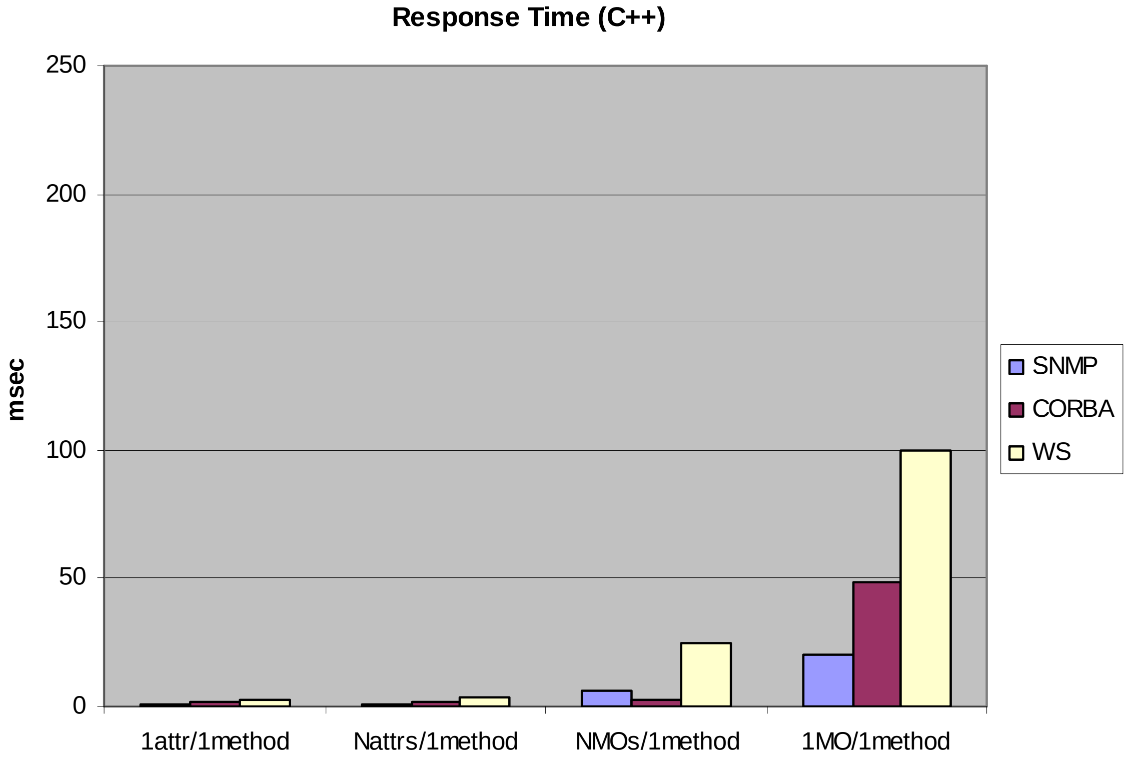
WS/SOAP vs SNMP & CORBA: Java Response Times



	Response Time (msec)			
	1attr/1method	Nattrs/1method	NMOs/1method	1MO/1method
SNMP	1	2	8	45
CORBA	2	4	6	97
WS/SOAP	5	9	40	250

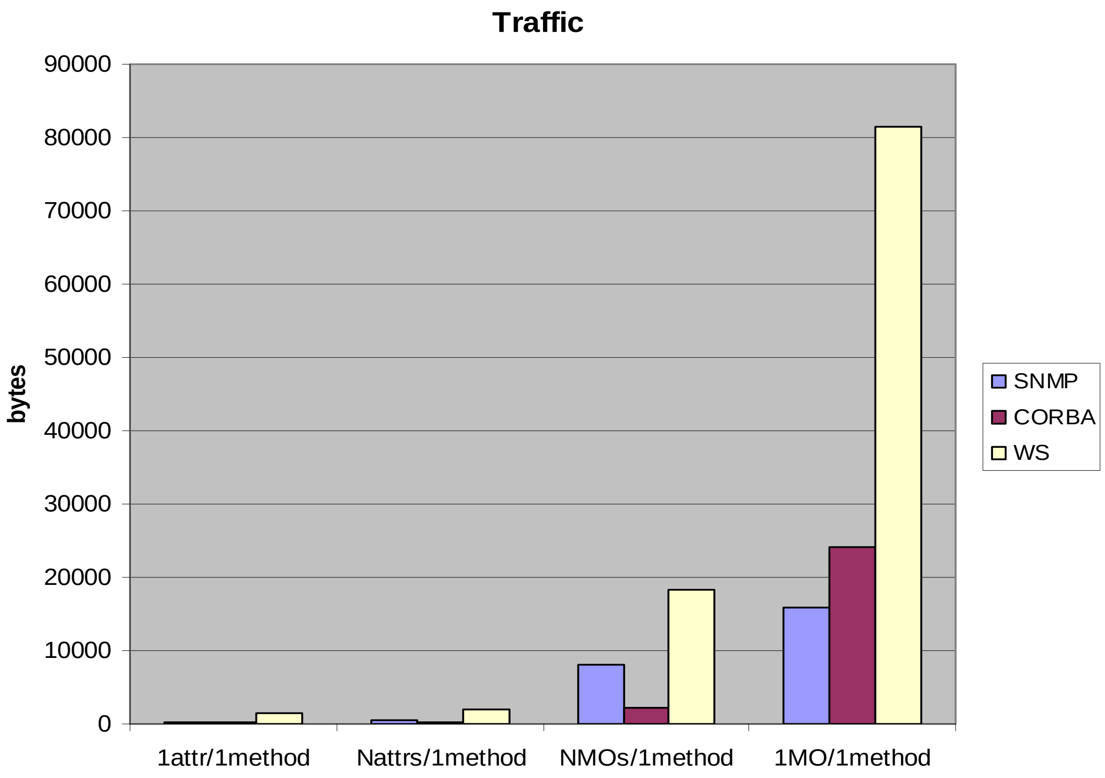
UniS

WS/SOAP vs SNMP & CORBA: C++ Response Times



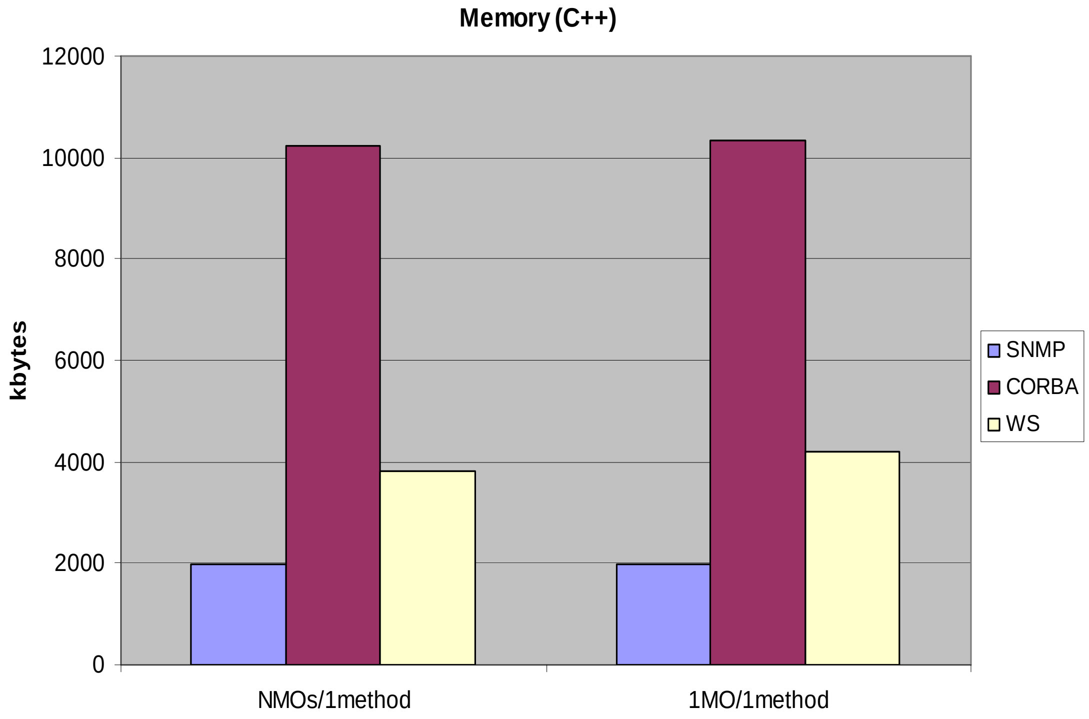
	Response Time (msec)			
	1attr/1method	**Nattrs/1method**	**NMOs/1method**	**1MO/1method**
SNMP	0.8	1	6	20
CORBA	1.5	1.7	2.5	49
WS/SOAP	2.5	3.7	25	100

WS/SOAP vs SNMP & CORBA: Traffic Incurred



	Traffic (bytes)			
	1attr/1method	Nattrs/1method	NMOs/1method	1MO/1method
SNMP	138	413	8160	15917
CORBA	280	316	2252	24157
WS	1390	2052	18266	81350

WS/SOAP vs SNMP & CORBA: C++ Memory Footprint



	Memory (kbytes)	
	NMOs/1method	1MO/1method
SNMP	1981	1981
CORBA	10236	10348
WS	3816	4180

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