WSMB: a middleware for enhanced Web services interoperability

Trung Nguyen Kien, Abdelkarim Erradi and Piyush Maheshwari

School of Computer Science and Engineering
The University of New South Wales, Australia
{trungnk, aerradi, piyush}@cse.unsw.edu.au

Abstract. Service-Oriented Architecture (SOA) using Web services is considered as the state-of-the-art approach to support interoperability between distributed systems and therefore facilitates complex interactions between heterogeneous and autonomous systems both within the enterprise and for cross-organizational collaboration. Message-based interactions are seen as the core building block in this new document-centric computing paradigm. Building SOA-based applications is a complex undertaking, design patterns and frameworks have an important role to play to ease the process and to provide high interoperability. This paper reviews interoperability challenges and possible solutions to address them then it reports our experiments with Web Services Message Bus (WSMB), a lightweight service-oriented integration framework for dependable Web services interactions using broker pattern. The main goal that guided the framework's deign is to maximize interoperability through a scalable mediation service to cope with the inherit heterogeneity of systems involved. We discuss the system architecture, some implementation issues and report our experiences in building WSMB as well as some initial performance results.

1. Introduction

The increasing move towards extensive automation of business processes coupled with the need for agile inter-enterprise cooperation over the Web, has fuelled interest in Service Oriented Architecture (SOA) as business processes are likely to involve high number of participants with diverse and incompatible technologies. SOA is an approach for designing, architecting and delivering interoperability-centric and loosely coupled integration where software capabilities are seen as services delivered and consumed on demand [1]. Message-based interactions using an orchestrated sequence of message exchanges (rather than invoking methods on interfaces) are seen as the core building block for appli-

cations using this emerging computing paradigm as they provide the glue that ties services together [1].

Web services offer a strong foundation for software interoperability through the core open standards of XML, SOAP, WSDL, and UDDI. Moving Web services beyond simple point-to-point applications to Internet-scale application-to-application interoperation, requires a paradigm shift from RPC-based/object-based architecture to a loosely-coupled, message-focused and service-oriented architecture. This paper contributes to an improved understanding of Web services interoperability challenges. It presents WSMB, as a light-weight integration framework that mediates Web services interactions to enhance interoperability and reliability. WSMB adds value and directly addresses some of the most critical issues surrounding the enterprise-grade deployment of Web services.

The rest of this paper is organized as follows. Section 2 discusses background material about Web services interoperability challenges and solutions while Section 3 discusses related work. Section 4 describes the architecture and features of WSMB. Section 5 presents initial evaluation results and discusses interoperability and performance aspects of WSMB. Section 6 finally concludes the paper and provides some thoughts on future work.

2. Background and problem area

Interoperability between systems is becoming more challenging in the Web era because the increased level of connectivity, the information space is large and dynamic, the data formats and semantics are becoming more diverse, systems are autonomous and there is a need for integration to be simple, fast, secure and adaptable to changes.

Interactions between heterogenous and autonomous applications require interoperability at three layers: communication, content and business process layer [4]. Interacting applications need to agree on their joint business process (e.g., documents format, contracts), the content of the exchanged documents as well as the communication protocol to exchange messages.

The ***communication layer (Transport)*** provides protocols for exchanging messages among applications (e.g., HTTP, JMS...). The objective of interoperability at this layer is to achieve seamless interactions over various communication protocols (e.g., RMI, JMS, DCOM…). Gateways could be used to mediate between heterogeneous protocols.

The ***content layer (Data)*** provides languages and models to describe and organize information. The objective of interoperability at this layer is to resolve semantic and structural heterogeneity issues in order to achieve a seamless integration of data formats, data models, and languages, through reconciliation/transformation among disparate representations, vocabularies, and semantics.

The ***business process layer (Services composition)*** is concerned with semantic consistency in business processes such as ordering, billing, or shipping, by explicitly defining

the implications of processing of messages and what responses are expected, etc. The objective of interoperability at this layer is to allow autonomous and heterogeneous partners to advertise their terms and capabilities, and engage in peer-to-peer interactions.

The objective of achieving software interoperability at different layers has been there for years and still remained a main research focus. Before Web services came into picture, there have already been number of prior technologies like EDI (Electronic Data Interchange), CORBA (Common Request Broker Architecture), DCOM (Distributed Component Object Model) and Java RMI (Remote Method Invocation) with limited success. Neither of the existing technologies could gain broad industry supports and acceptance due to their complexities (e.g., CORBA, DCOM), cost factors (EDI), or platform/language specifics (e.g., RMI). Nevertheless, these attempts have yielded tightly coupled systems that are not suitable for B2B integration [3]. Web services promises to overcome these limitations by promoting standards-based interoperability and loose coupling.

Web services are software components that expose some computational or data offering capabilities in a standard format and accessible using standard Internet protocols [1]. Web services interactions are based on exchanging standards-based XML documents (i.e., message-based interoperability). Interoperability is one of the main features that Web services promise. Along with its open standards, Web services permit seamless communication between applications in a platform and language neutral way.

Simple Object Access Protocol (SOAP) is one of the most important standards in building a backbone infrastructure for Web services communication [9]. SOAP deals with objects serialized to plain text and not with remote object references, distributed garbage collection has no meaning and is not covered. For the same reason, SOAP clients do not hold any stateful references to remote objects. Due to these characteristics, using SOAP, disparate components implemented in different programming languages and different technologies can talk to each other in a distributed, decentralized environment, as long as component frameworks can issue and process XML messages.

In a fully autonomous collaboration, a Web service can be viewed as a black box that is able to exchange messages conforming to well-defined schemas. This gives interacting partners more local control over implementation and operation of services, and flexibility to change their processes without affecting each other.

However, the capabilities of current Web services frameworks are limited and do not address the potential mismatches that hinders Web services interoperability, particularly data mismatch (data structure and semantic), protocols mismatch (the interacting applications might use different protocols) and semantic mismatch (the interacting parties might interpret the same information in different way). We propose a mediation layer named Web Services Message Bus (WSMB) that bridges differences in communication protocols, data formats and in future work will extend the framework to support the reconciliation of incompatibilities in business protocols and processes.

3. Related work

Several ongoing efforts recognize the need to extend the current technological infrastructure to foster wider adoption of Web services and SOA. In our early work [5], we tried to explore enhancing Web services by introducing Message Oriented Middleware (MOM) in the architecture. We came to the conclusion that it is unlikely that Web services can simply adapt existing MOM technologies to achieve higher QoS properties, given that they try to address interoperability on a scale at which conventional technologies have failed. Hence there is a clear need for new wide-area messaging infrastructure to enhance service delivery in web-centric environment. WSMB is a first step towards achieving this. Its design was initially inspired from ideas introduced by the Web services Invocation Framework (WSIF) [6] but it adds more advanced QoS features as well as support for message-oriented interactions instead of RPC-based invocations promoted by WSIF. Web services are also an active area of standardization; our approach builds upon the building blocks of these standards and seeks to refine them by providing early implementations. The efforts around Enterprise Service Bus (ESB) [8] are closely related to our work but their focus is on Enterprise Application Integration (EAI). In our approach we contribute to the realization of ESB vision while putting more focus on decentralized architecture, and broadening ESB application space as well as validating its feasibility and benefits.

Our future work will leverage the outcomes of Harmonized Messaging Technology (HMT) project [9] which is working on defining a formal and generic specification language to express and capture rules and constraints that govern messaging-based interactions.

WSMB aims to extend and leverage the principles of messaging systems as well as the emerging Web services standards to support more interoperable and manageable Web services interactions using a decentralized architecture. The next stage of our research is to complete the development of the framework and to develop a number of services to fully test the features of WSMB.

4. WSMB – A highly interoperable middleware for Web services

WSMB is an important architectural component for SOA-based applications. It is a service intermediary that enhances the delivery of Web Services by providing run-time support for reliable messaging, routing, monitoring and managing Web Services. It can be used internally and in B2B scenarios. WSMB aims to facilitate large scale deployment of Web services in a secure, reliable and consistent manner by offering the following features:

- Multi-protocol services to bridge interfaces and protocol differences. This is motivated by the difficulty to dictate specific protocols and policies to business partners. Therefore, WSMB infrastructure acts as a service façade exposing services via number of protocols and policies that can be declaratively defined and enforced. For example, for secure access, a Web service client can choose between HTTS and WS-Security over http.
- Protect applications from emerging and rapidly evolving Web services standards: WSMB provides an extensible infrastructure to abstract away and implement standards compliance without application changes.
- Protect servers from overload by queuing or redirecting messages when the average response time goes beyond a certain threshold that is configured on a specific Web service using WSMB management console. Interested parties can also subscribe to violation alert notifications by e-mail or SOAP action. For example, a trigger could be configured for an order processing service to raise an alert when more than 10 orders are being processed per second or if order processing takes more than 5 seconds. WSMB could then be configured to respond to the trigger by queuing orders or by processing high value orders first. This allows WSMB to perform basic load balancing and fail-over.

This paper concentrates on the interoperability features of WSMB.

4.1 Architecture overview

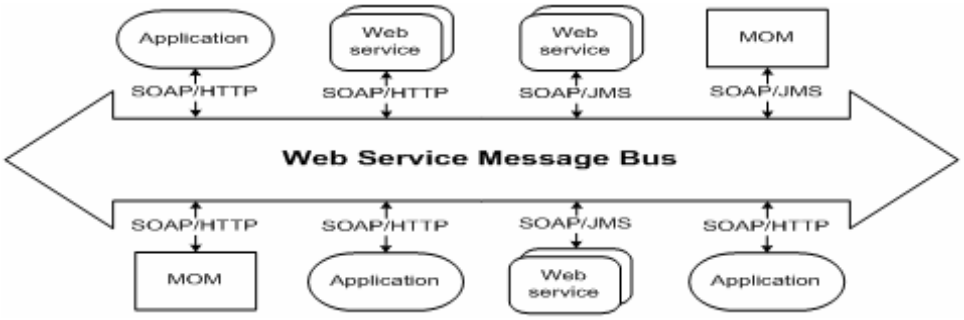

Figure 1. WSMB architecture overview

Figure 1 illustrates WSMB architecture. Though WSMB is implemented by using the basic Message Oriented Middleware (MOM) concepts [13], it serves as a bus, analogous to computer hardware bus system, rather than a normal queuing system. This bus architecture emphasizes on the integration capability among connected applications (e.g., Service clients, Web service endpoints, MOM application, etc).

The core idea of WSMB is to act as a bus which conveys SOAP messages from one end to another regardless the transport protocols (e.g., HTTP, JMS) being used in either ends.

4.2 WSMB communication layer interoperability

HTTP is considered as the standard transport protocol for exchange SOAP messages. Consequently, Most of the current Web services frameworks like Axis are almost exclusively geared to synchronous invocation over HTTP which might not be suitable for applications with high reliability requirements.

As a middleware bridging between clients and Web service providers, WSMB provides a configurable framework for reliable Web services interactions. WSMB provides various channels to access the registered Web services (each service is bound to one or more channels). The incoming message is assessed on arrival through the channel to determine the destination service. Filters bound to the destination service –if any- intercept and manipulate both request and response messages (e.g., transform old-format messages into new formats). The message is then passed through a reliability layer where it is checked -for expiration, duplication, and ordering- and then it get queued for processing. WSMB then dispatches the message to the destination Web service and the response is passed back to the requester via the same path. The diagram in Figure 2 shows WSMB communication layer interoperability.

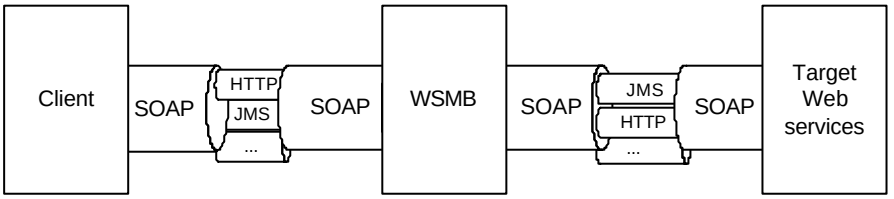

Figure 2. Supporting multiple transport protocols enhances interoperability

WSMB handles interception of messages and protocol bridging to allow interactions over multiple communication protocols (e.g., HTTP, JMS, MSMQ…). The choice of the appropriate protocol to use can be decided based on the required Quality of Service (QoS) and available infrastructure. WSMB allows the Web service client to choose the desired transport without directly interfacing any deployed middleware that transports SOAP. From the schema of the endpoint URL of the destination Web service (e.g., jms:) WSMB transparently uses the appropriate communication channel. Our goal here is to allow a client to transparently send a message to queue as it would to an http-based endpoint. WSMB demonstrates the ability to effectively switch between transports protocols by

simply altering the target endpoint from a HTTP URL, to a listener waiting on JMS or WebsphereMQ queue in synchronous and asynchronous fashion.

Multi-protocol support helps to preserve interoperability while leveraging existing infrastructure. For instance, a purchase order encapsulated in a SOAP message could arrive using HTTP then WSMB could place the SOAP message in a message queue to be to be pickup, validated, prioritized and processed by an order processing workflow system. The requesting process will eventually receive a SOAP message confirming or rejecting the order through WSMB mediation. This allows shielding external users and partners from the internal implementation details while keeping data losses to a minimum.

4.2 WSMB content layer interoperability

WSMB infrastructure is based on interception (Figure 3). At start-up it loads the configuration and starts a pool of listeners to begin receiving messages. The listeners read messages from different channels and then call a series of configurable handlers to manipulate both request and response messages (e.g., transform old-format messages into new formats) as instructed by the configuration settings. Messages travels along a configurable pipeline made of a series of operations for processing and relaying messages. WSMB support two types of operations generic (like validate a message against a schema, transform a message with XSLT, split, route, encrypt/decrypt message…) and specific (like a custom handler to evaluate or enforce a business rule). Operations can be assigned to different nodes in the infrastructure. Each node receives the messages into a private queue, and processes them through a configurable sequence of handlers, after that the message(s) could leave the node to a certain destination or set of destinations.

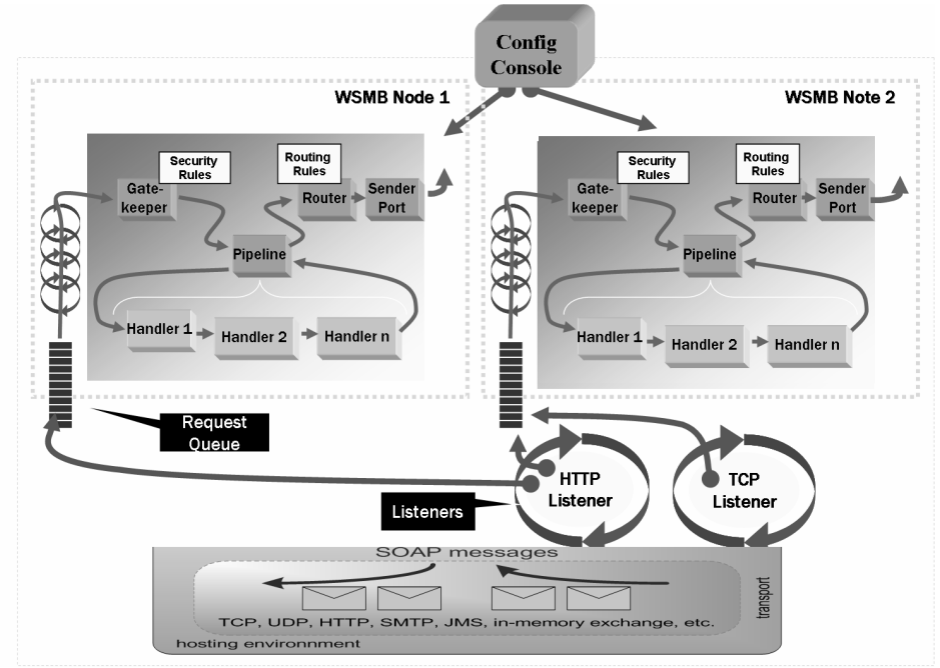

Figure 3. SOAP handler pipeline mechanism enhances content level interoperability

5. Evaluation and discussion

5.1 Performance Evaluation

Our performance tests aimed at comparing the direct invocation using SOAP-over-HTTP with channelling interactions through WSMB.
We run independent tests with increasing number of clients, and fixed the number of requests sent by each client to 50 requests per client. The graph in Figure 5 summarizes some of the results.

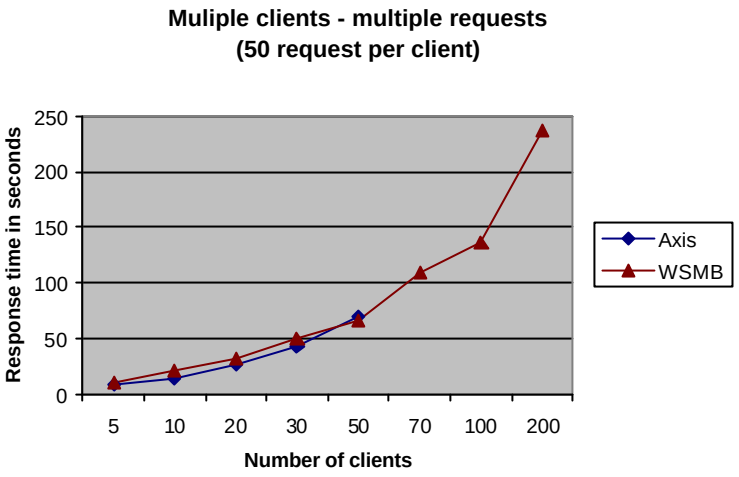

Figure 1. Multiple clients send multiple requests

Looking at the graph, we can observe that direct SOAP-over-HTTP using Axis slightly outperforms channelling SOAP messages through WSMB. However, Axis cannot handle more than 70 clients. Whereas scalability of WSMB still grows up to 200 clients. So messaging through WSMB is more resilient and the response time is more predictable.

5.2 Interoperability Evaluation

SOAP over HTTP is commonly used to invoke Web services. In this evaluation, we discuss an alternative approach to convey SOAP messages to the target Web service (i.e., SOAP-over-JMS).

Scenario 1 – SOAP/JMS: Invoking Web service deployed in Oracle AS J2EE 10g

In this scenario, we deployed a NewsService [14] in OC4J.

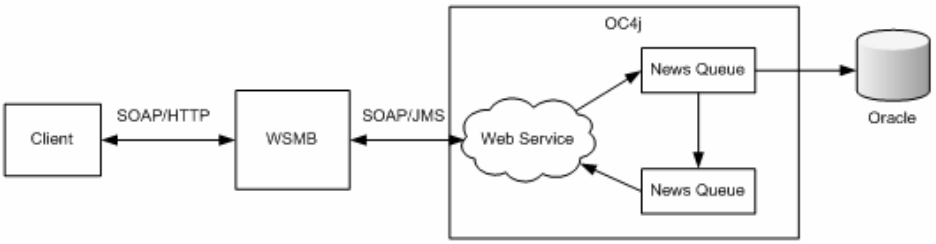

Figure 6: SOAP/JMS with OC4J

The above figure illustrates the work flow of the scenario.

NewsService is a JMS Document Style Web Service which exposes a JMS Queue (News Queue) as a Web Service. This service accepts messages (clients send messages by invoking the 'send()' operation of the Web Service), extracts the news content from the message, persists to the data source and finally publishes to a JMS Topic (News Topic).

The JMS Topic (News Topic) is also exposed as a JMS Document Style Web Service and hence all clients can subscribe to this Topic and receive news items. This is done by invoking the 'receive()' operation of the Web Service.

The above scenario illustrates the interoperability features of WSMQ with a commercial Web services container.

6. Discussion and Conclusion

In this paper we argue that XML-based messaging accompanied by a uniform accessible and manageable communication bus fulfils the interoperability and scalability requirements among services in a generic and dynamic manner.

We presented an initial design and prototype implementation of WSMB, a lightweight messaging framework using broker pattern, it builds on current standards to meet the interoperability and manageability requirements of Web services based applications. WSMB is a first step towards the realization of an interoperable and QoS-aware integration infrastructure to move SOA beyond simple point-to-point service integration by facilitating large-scale implementation of the SOA principles with suitable service levels and manageability. The fundamental principle behind WSMB is abstracting away interoperability features so they can be configured and enforced instead of being hand-coded into the service logic. This results in a greater flexibility and a better maintainability and reuse. While not claiming to be a sliver bullet, WSMB has achieved (in non-invasive way) some success in enhancing interoperability through virtualization of communication protocols and endpoints as well as the provision of a configurable SOAP handlers pipeline to ease the reconciliation of content layer mismatches. However, the framework needs to be improved by adding support for semantic transformation, content-based routing, and advanced reconciliation services using ontological information to allow increased interoperability.

References

1. Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju: *Web Services Concepts, Architectures and Applications*. 2004: Springer.
2. W3C: *Web Services Glossary*. http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/. February 2004
3. IBM Web Services Architecture Team: *Web Services Architecture Overview*. http://www-106.ibm.com/developerworks/library/w-ovr/. September 2000
4. Dabous, F. T., Rabhi, F. A., Ray, P. K. and Benatallah, B., 'Middleware Technologies for B2B Integration', The International Engineering Consortium (IEC) Annual Reviews of Communications, IEC Press, USA, Vol 56,July, 2003.
5. Maheshwari, P., Nguyen, T. and Erradi, A., 'QoS-based Message-Oriented Middleware for Web services', in WISE 2004 Web Services Quality Workshop, Brisbane, Australia, LNCS 3307, pp. 241-251.
6. Mukhi, N.K., Khalaf, R. and Fremantle, P., 'Multiprotocol web services for enterprises and the grid', in Proceedings of the EuroWeb 2002 Conference on the Web and the Grid: From e-science to e-business, Oxford, UK
7. SonicSoftware: *SonicMQ Datasheet*. http://www.sonicsoftware.com/products/docs/sonicmq.pdf. Accessed on October 2004
8. Chappell, D. 2004, Enterprise Service Bus, O'Reilly, ISBN 0-596-00675-6.
9. Sadiq, S. W., Orlowska, M. E., Sadiq, W. and Schulz, K., 'Facilitating Business Process Management with Harmonized Messaging', in 6th International Conference on Enterprise Information Systems (ICEIS 2004), Porto, Portugal
10. Wipro Web Service Team: *Interoperability Issues of Web Services - An Overview*. http://whitepapers.zdnet.co.uk/0,39025945,60045354p-39000542q,00.htm (registration required). March 2003
11. *Microsoft Soap Interop Server*. http://www.mssoapinterop.org/.
12. *SOAPBuilders Interoperability Lab*. http://www.xmethods.net/ilab/.
13. Piyush Maheshwari, Hua Tang, and Roger Liang: *Enhancing Web Services with Message-Oriented Middleware*. in *Proceedings of the IEEE International Conference on Web Services*. July 2004. San Diego.
14. Oracle, JMS Web Service Sample (News Service application). Accessed on October 2004. http://www.oracle.com/technology/sample_code/tech/java/web_services/jmsws/index.html