Cardoso, J., "Benchmarking a Semantic Web Service Architecture for Fault-tolerant B2B Integration", International Workshop on Dynamic Distributed Systems (IWDDS), In conjunction with the ICDCS 2006, The 26th International Conference on Distributed Computing Systems July 4-7, 2006 - Lisboa, Portugal. IEEE Computer Society. ISBN: 0-7695-2541-5, ISSN: 1545-0678.

Benchmarking a Semantic Web Service Architecture for Fault-tolerant B2B Integration

Jorge Cardoso
Department of Mathematics and Engineering
University of Madeira, 9050-390 Funchal, Portugal
jcardoso@uma.pt

Abstract

With the development and maturity of Service-Oriented Architectures (SOA) to support business-to-business transactions, organizations are implementing Web services to expose their public functionalities associated with internal systems and business processes. In many business processes, Web services need to provide is a high level of availability, since the globalization of the Internet enables business partners to easily switch to other competitors when services are not available. Along with the development of SOA, considerable technological advances are being made to use the semantic Web to achieve the automated processing and integration of data and applications. This paper describes the implementation and benchmarking of an architecture that semantically integrates Web services with a peer-to-peer infrastructure to increase service availability through fault-tolerance.

1. Introduction

Organizations are using the architectural benefits of Service-Oriented Architectures (SOA) to coherently map business processes with enterprise applications. Using Web services it is possible to support the externalization of atomic business capabilities by making business interfaces more transparent. As organizations move to business-to-business (B2B) models, supported by SOA and Web services, they must develop solutions to cope with failures that can cause systems downtime in the supply-chain. The consequences of failures can ripple across multiple organizations and can have significant financial costs. Therefore, providing highly reliable B2B systems is an important goal. Web service computing is still in an evolving state and much research needs to be done to overcome complex issues such as fault-tolerance, availability, and scalability.

Many organizations currently use, or will be soon using, Web services to manage a broad range of distinct distributed applications, such as insurance claim processing, bank loan management, and healthcare processes. Applications can be more oriented to support or enhance existing business processes, to increase competitive advantage, to reduce costs, and also to manage critical infrastructures. In many cases, Web services are of vital significance to the organizations that govern them and the downtime of services can easily incapacitate the completion of running business processes. For example, it is not advisable for an insurance company to delay a customer's insurance claim processing due to a Web service failure. It is also not acceptable to delay a patient's treatment due to a Web service malfunction. High availability, fault tolerance, and scalability are aspects of intra- and inter-organizational Web service-based distributed applications that represent important research areas for SOA.

Current Web service specifications [1] do not provide support to handle service failures and prevent service downtime. The mechanisms provided by SOAP and WSDL help handling errors raised by applications, but no mechanism exists for handling system failures [2]. At the SOAP messaging layer, the <soap:fault> tag is provided to inform a client about errors encountered while processing an invocation message. At the WSDL description layer, the <wsdl:fault> tag provides a way to output the result of a remote operation invocation error.

The purpose of our work is to describe the design, implementation, and benchmarking analysis of a fault-tolerant architecture called Whisper, which provides a transparent approach to enable a significant increase in the availability of Web services whilst at the same time have a minimal impact on B2B distributed

applications' complexity. Whisper system uses emerging technologies, such as the semantic Web, Web services, and peer-to-peer (P2P) networks, for building the next-generation service oriented systems.

The system that we have developed to increase the fault tolerance of Web services differs from previous work [2, 3] since we explore the features and characteristics of peer-to-peer networks to develop a transparent and scalable mechanism to increase the availability of Web services. Another major difference is related to the approach that we have adopted to enable the integration and interoperation of Web services and P2P networks, which uses semantics and ontologies.

2. Web service and P2P semantic Integration

Our approach to Web service fault-tolerance consists of an infrastructure based on a service-oriented architecture, named Whisper, which increases the availability of Web services by using a fault-tolerant mechanism built on peer-to-peer networks and the semantic Web. Whisper architecture integrates semantic Web services and a semantic P2P infrastructure (Figure 1).

Web Services are based on a centralized model and primarily focused on standardizing messaging formats and communication protocols. P2P computing, on the other hand, is based on a decentralized model.

The decentralized model gives a natural approach to develop self-healing and resilience architectures through redundancy. This is precisely how Whisper achieves fault tolerance. We have selected the JXTA [4] infrastructure to implement fault-tolerant mechanisms to insure a high degree of availability of peers that actively communicate with Web services.

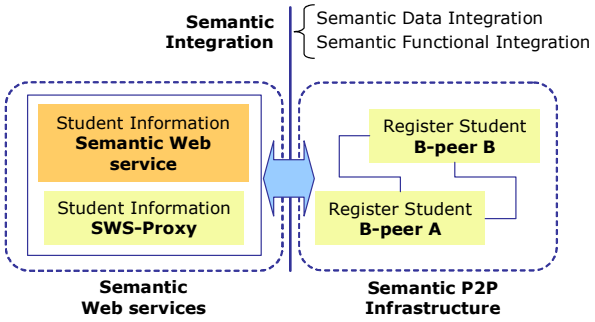

Figure 1. Semantic Integration

2.1. Heterogeneity and Integration Challenges

The problems that might arise when integrating Web services and JXTA infrastructures due to several types of heterogeneity are very similar to the problems known within the distributed database systems community (e. g. [5, 6]). Heterogeneity occurs when there is a disagreement about the meaning, interpretation, or intended use of the same or related data. As with distributed database systems, four types of information heterogeneity [7, 8] may arise in Whisper: system heterogeneity, syntactic heterogeneity, structural or schematic heterogeneity, and semantic heterogeneity.

While Whisper deals with all these types of heterogeneity, it tackles in particular semantic heterogeneity. Approaches to the problems of semantic heterogeneity should equip heterogeneous, autonomous, and distributed software systems with the ability to share and exchange information in a semantically consistent way. The semantic integration of Web services and the P2P infrastructure is achieved using an ontology representation language (OWL) which provides a key element to deal with semantic heterogeneity. Integrating two distinct architectural models requires, among other types of integration, dealing with semantic data integration and semantic functional integration.

2.2. Semantic Data Integration

Web services and JXTA networks use different standardized technology. As a result, incompatibility arises from semantic differences of data schema. In a B2B application, Web services and JXTA peers take a set of data inputs and produce a set of data outputs. Web services and JXTA specifications use only syntactic and structural details of the input/output data. Each data schema is set up with its own structure and vocabulary. To allow the integration of Web services and JXTA peers to exchange data at the semantic level, the semantics of the input/output data have to be taken into account. Hence, we annotated the data of Web and JXTA peer services using ontological concepts [9, 10]. The added semantics can be later used in matching the semantics of the input/output of Web services and JXTA peer services when exchanging data, which was not possible when considering only syntactic information.

2.3. Semantic Functional Integration

Web service and JXTA peer specifications only defines syntactic characteristics. The signature of an operation provides only the syntactic details of the input data, output data, and operation's name. Technological solutions to integrate Web services and JXTA peer networks using operations signatures are not sufficient since services' functionality cannot be precisely expressed. As a step towards representing the functionality of services, in Whisper, Web services and JXTA peers are annotated with functional semantics.

2.4. Other Integration Issues

While our the Whisper system only deals with semantic data integration and functional integration, an other integration issue that can be considered and explored is semantic QoS integration [11].

QoS Semantics. After discovering a JXTA peer whose data and functional semantics match the semantics of the required Web service, the next step is to select the most suitable peer. Each peer can have different quality aspect and hence selection involves locating the peer that provides the best quality criteria match. This demands management of QoS metrics for peers. For organizations, being able to characterize Web services and peers based on QoS has several advantages. It allows organizations to translate their vision into their business processes more efficiently, since services can be designed according to QoS metrics.

3. Semantic Web services and SWS-Proxies

3.1. Semantic Web services

To facilitate the understanding of Whisper architecture we describe a running scenario which is partially illustrated in Figure 2. The application shown has the 'Student Information' Web service available to clients. This service accepts as input a student ID, connects to a relational database, retrieves the information of the student, and returns a structure with the information to the client. The actual implementation of this service is not associated with the Web service itself, but it is supplied by a JXTA network of b-peers (see section 4.2.).

Traditional Web services are described using the WSDL, which provide only syntactical information. However, WSDL poses a problem during the automatic discovery of peer groups to carry out the actual execution of a Web service, since the use of syntactic information alone originates a high recall and low precision during the search [12].

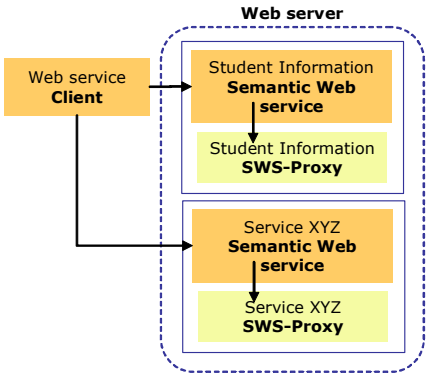

Figure 2. Semantic Web services and SWS-proxies

Whisper supports the notion of semantic Web services. Semantic Web services are the result of the evolution of the syntactic definition of Web services and the semantic Web. With the help of ontologies, the semantics or the meaning of service data and functionality can be explicated. As a result, integration can be accomplished in an automated way and with a superior degree of success.

In Whisper, Web service are semantically annotated following the WSDL-S specification [9, 13]. JXTA peer groups are also semantically annotated. The semantic annotation of Web services and JXTA peer groups allows their semantic integration at the data and functional levels. WSDL-S establishes mapping between WSDL descriptions and ontological concepts. The idea of establishing mappings between service, task, or activity descriptions and ontological concepts was first presented in [10]. The following example illustrates how a WSDL specification, from our initial scenario, is mapped to ontological concepts.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name = "StudentManagement"
...
 xmlns:sm =
"http://.../jcardoso/StudentMng.owl#"
...
<interface name = "StudentManagementUMA" >
<operation name = "StudentInformation" ... >
 <action element = "sm:StudentInformation" />
 <input messageLabel="ID"
                    element="sm:StudentID"/>
 <output messageLabel="student"
                   element="sm:StudentInfo"/>
</operation>
</interface>
</definitions>
```

The WSDL-S specification indicates that the Web service supplies the one operation

'StudentInformation'. This operation uses ontological concepts to annotate the input, output, and action. The ontological concepts are expressed in the ontology http://dme.uma.pt/jcardoso/StudentMng.owl#, which is specified using OWL.

3.2. SWS-Proxies

When a Web service receives a request it forwards it to the Semantic Web Service proxy (SWS-proxy). Proxies contact the JXTA infrastructure and using the Semantic Discovery Service (Figure 3) locates a semantic group of peers that can satisfy the client's request. Once a suitable semantic group of peers is found, the group is queried to find a b-peer that will process the client's request.

```
. . .
public class SWS-proxy {
. . .
 // reference to the semantic Web service
 SemanticWebService sws;
. . .
 // op is an operation
 public SemanticeAdv
        findSemanticPeerGroupAdv(String op) {
. . .
 e = discovery.getLocalAdvertisements(
                       DiscoveryService.ADV,
                       "action",
                       sws.get_Sem_action());
. . .
 while (e.hasMoreElements()) {
  SemanticeAdv sAdv = null;
  sAdv = (SemanticeAdv) e.nextElement();
  if (
   sAdv.getInput().equals(sws.get_Sem_input(op
)) &&
   sAdv.getOutput().equals(sws.get_Sem_ouput(o
p)) {
   // found a semantic peer group
advertisement
   // matching the Web service semantics
   return sAdv;
  }
 }
. . .
```

In the previous example, the SWS-proxy tries to find JXTA semantic group advertisements based on the semantic functionality (action) of its Web services. When advertisements that have the same semantic functionality (see section 2.3) of the semantic Web service request are found, the SWS-proxy checks if the b-peers inside the peer group discovered have also the same data semantics (see section 2.2) of the semantic Web service request. If they do, the advertisement is returned to the SWS-proxy that will connect to a b-peer of the semantic peer group found (this last phase is not shown in this example.)

4. B-Peer Groups, B-Peers, and Semantic Advertisements

Redundancy has long been used as a means of increasing the availability of distributed systems, with key components being replicated to protect against failures. In Whisper, redundancy is achieved using the replication of business process functionalities. Typically, an application's logic and data is distributed on a cluster (group) of computer systems to ensure that it can tolerate any single hardware or software fault within the cluster. The redundancy mechanism of Whisper makes possible to also address scalability requirements through load-sharing, since peer services can be replicated among different computers. We use static redundancy which means that all replicas implementing services are active at the same time. If one replica fails another replica is elected (using the Bully algorithm) and used immediately with little impact on response time.

4.1. B-peer groups

Peers are self-organized into b-peer groups which are logical rather than physical entities (Figure 3). Each b-peer belongs to a semantic b-peer group. The b-peers of the same semantic b-peer group implement the same functionality service, but possibly in a different way. When a Web service is invoked by a client, Whisper dynamically tries to find a semantic b-peer group that will be able to process the requested service.

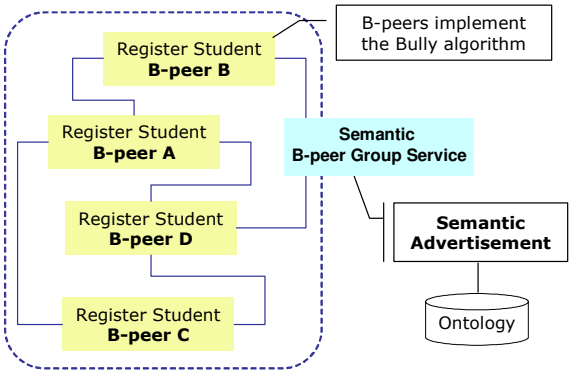

Figure 3. Semantic b-peer groups and b-peers

For example, we may envision the following scenario. In response to a Web service request, a peer accesses student information from an operational database and returns the results to the client. If the operational database is unavailable, a semantically equivalent peer can automatically and transparently

handle the service request by retrieving the same information from a data warehouse.

4.2. B-peers

Once a suitable semantic group of peers is found, the group is queried to find a b-peer that will process the client's request. B-peers are entities on a network implementing one or more JXTA protocols. They implement a specific functionality, such as accessing a database to retrieve students' data, and more importantly they implement the Bully algorithm to provide a fundamental mechanism to enable a good fault-tolerance.

When a b-peer group requests a b-peer to carry out a Web service request, the b-peer found may not be the coordinator. Therefore, additional processing may need to be done to find the current coordinator of the semantic group. When the coordinator is identified, it processes the request and sends the results of the processing to the SWS-proxy. The proxy translates the data received to a suitable format and sends the results to the semantic Web service that will in turn send the results back to the client that initially issued the request.

B-peers exist independently and communicate with other b-peers asynchronously. JXTA provides a framework that allows developers to concentrate on providing high level, business-oriented functionality, rather than implementing the underlying infrastructure. JXTA networks are inherently dynamic. By using a number of protocols, b-peers may join or publish advertisements at different times. For Whisper this characteristic is important since it allows to dynamically increasing the level of availability of a Web service by having a higher number of peers responsible for the processing of service requests.

4.3. Semantic Advertisements

In Whisper, b-peer groups semantically advertise to other b-peers the services they provide to the network. All resources in JXTA networks are represented by a metadata XML document called an advertisement. B-peers publish and discover advertisements representing other resources such as b-peers and b-peer groups.

The default discovery supported by JXTA is inefficient as b-peers retrieved may be inadequate due to low precision (many b-peers you do not want) and low recall (missed the b-peers you really need to consider). The search has to be based, not only on syntactic information, but also on data, and functional semantics. Effectively locating relevant b-peers is required to performing the search operation in a scalable way. To meet this challenge, we use 'extendable advertisements' to create a new type of advertisement that uses semantic information to describe our semantic peer groups. This new type of advertisements is called semantic advertisement.

5. Benchmarking

To validate Whisper we have carried out a benchmark in order to assess the scalability and performance of our architecture under system load. We present the results that have been obtained when implementing Whisper with JXTA infrastructure. We have measured the scalability of the overall system, as well as the latency of JXTA infrastructure and Web services invocation latency. Since our infrastructure implements a distributed algorithm to enable a high degree of availability using message exchange, it is important to analyze the variation of the number of messages exchanged as the number of b-peers increases.

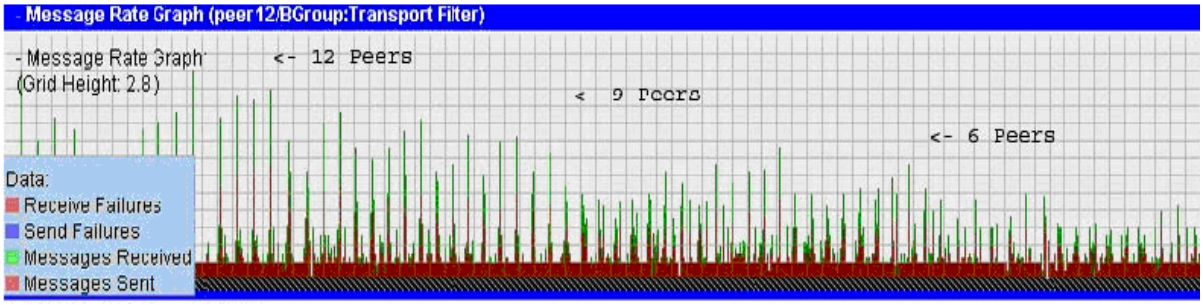

Figure 4. Variation of the number of messages exchanged as the number of B-peers increases

Our distributed architecture consists of 9 identical machines, each equipped with Intel P4 3.0 GHz processors, 512 MB main memory, 40GB 7,800 RPM IDE disks, Microsoft Windows XP home, Java SDK 1.4, and JXTA 2.3.2. The personal computers were connected by a 100Mbit/s Ethernet LAN.

Figure 4 shows the number of messages exchanged between a variable numbers of b-peers. The system performance benchmarking exercise revealed that the proposed solution was able to scale to meet desired throughput and latency requirements. It can be observed that the architecture exhibits a good linear horizontal scalability – adding new b-peers to the configuration results in a predictable linear increase in the number of messages exchanged.

The results obtained are encouraging since JXTA is inherently a heavy architecture and given that it provides an abstract network transport capable of transporting messages between peers, either directly, or via relay peers capable of both enabling multi-hop routing of messages, and traversing firewall or NAT (network address translation) equipment that isolates peers from public networks.

We have also analyzed the Round-Trip Time (RTT) of messages to measure characteristics of the network, such as the bandwidth and latency. RTT is defined as the time interval from the moment at which a request packet is time-stamped by the monitor to the moment at which a reply packet is time-stamped. Our results showed that the average latency is approximately 0.5 milliseconds. Nevertheless, in the worst case the RTT can take several seconds. This low performance is caused by two factors. On the one hand, in case of coordinator failure, the time needed to elect a new coordinator is considerably high. On the other hand, the time to make a new binding between the SWS-proxy and the elected b-peer is also high.

7. Conclusions

Since Web services (WSDL) do not provide any mechanism to increase their availability, we have used a P2P infrastructure (JXTA) to deploy a fault-tolerant peer-to-peer back-end architecture. The integration and interoperation of Web services and JXTA is a difficult task due to the heterogeneity of the two technologies. Our system, Whisper, uses semantic Web technology to integrate centralized and decentralized systems and share and exchange information in a semantically consistent way. To validate Whisper we have carried out a benchmark that has revealed that the system is scalable and achieves a good performance under system load.

8. References

[1].Curbera, F., W. Nagy, and S. Weerawarana. *Web Services: Why and How*. in *Workshop on Object-Oriented Web Services - OOPSLA 2001*. 2001. Tampa, Florida, USA.

[2].V. Dialani, et al. *Transparent fault tolerance for web services based architectures*. in *Eighth International Europar Conference (EUROPAR '02)*. 2002. Padeborn, Germany: Springer-Verlag.

[3].Looker, N. and M. Munro, *WS-FTM: A Fault Tolerance Mechanism for Web Services.* http://www.dur.ac.uk/computer.science/research/technical-reports/2005/A%20Fault%20Tolerance%20Mechanism.pdf. 2002.

[4].Gong, L., *Project JXTA: A Technology Overview* - http://www.jxta.org/docs/TechOverview.pdf. 2001.

[5].Kim, W. and J. Seo, *Classifying schematic and data heterogeinity in multidatabase systems.* IEEE Computer, 1991. **24**(12): p. 12-18.

[6].Kashyap, V. and A. Sheth, *Semantic heterogeneity in global information systems: The role of metadata, context and ontologies*, in *Cooperative Information Systems: Current Trends and Applications*, G. Schlageter, Editor. 1996, Academic Press: London, UK. p. 139-178.

[7].Sheth, A., *Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics*, in *Interoperating Geographic Information Systems*, C.A. Kottman, Editor. 1998, Kluwer, Academic Publishers. p. 5-30.

[8].Ouskel, A.M. and A. Sheth, *Semantic Interoperability in Global Information Systems. A brief Introduction to the Research Area and the Special Section.* SIGMOD Record, 1999. **28**(1): p. 5-12.

[9].Patil, A., et al. *MWSAF - METEOR-S Web Service Annotation Framework*. in *13th Conference on World Wide Web*. 2004. New York City, USA.

[10].Cardoso, J. and A. Sheth, *Semantic e-Workflow Composition.* Journal of Intelligent Information Systems (JIIS). 2003. **21**(3): p. 191-225.

[11].Cardoso, J. and A.P. Sheth, *Introduction to Semantic Web Services and Web Process Composition*, in *Semantic Web Services and Web Process Composition*, A.P. Sheth, Editor. 2005, Springer-Verlag: Heidelberg, Germany. p. 1-13.

[12].Sivashanmugam, K., et al., *Metadata and Semantics for Web Services and Processes*, in *Datenbanken und Informationssysteme (Databases and Information Systems) Festschrift zum 60*, R. Unland, Editor. 2003, Geburtstag von Gunter Schlageter: Hagen, Germany. p. 245-271.

[13].Rajasekaran, P., et al., *Enhancing Web Services Description and Discovery to Facilitate Composition*, in *Semantic Web Services and Web Process Composition*, A. Sheth, Editor. 2004, Springer-Verlag: Heidelberg. p. 55-68.