Semantic Integration of Web services and Peer-to-Peer Networks to Achieve Fault-tolerance

Jorge Cardoso

Abstract— **One fundamental property that critical Web services need to provide is a high level of availability. Along with the development of Web services, 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 of the Whisper architecture. This architecture semantically integrates Web services with a peer-to-peer infrastructure to increase service availability. Whisper achieves transparent fault-tolerance by automatically forwarding Web service requests to semantically equivalent peers that are dynamically located, selected, and invoked.**

Index Terms— **Semantic Web, Web Services, Peer-to-Peer, Fault-tolerance, Bully algorithm**

I. Introduction

THE vision of Service-Oriented Architectures (SOA) promises a new paradigm providing an extremely flexible approach for building complex information systems. Service-oriented architectures can rely on Web services to allow a more efficient integration of applications and improve the accessibility of business processes for customers and partners.

Current Web service specifications [1] do not provide support to handle service failures and prevent service downtime. It is therefore indispensable to start developing solutions to increase the fault tolerance of SOA based on Web services. The purpose of our work is to provide a transparent approach to enable a significant increase in the availability of Web services.

In this paper we describe the design and implementation of our fault-tolerant architecture called Whisper. We use emerging technologies, such as the semantic Web, Web services, and peer-to-peer (P2P) networks, for building the next-generation of service oriented systems. Currently, these three technologies constitute the most promising solutions for distributed computing and their importance has been recognized in several major conferences (e.g. ISWC 2005, WWW 2005, ICWS 2005, and P2P 2005). The specific contributions of this paper are: (i) the design of a fault-tolerant architecture which relies on the semantic integration of semantic Web services and a peer-to-peer infrastructure, (ii) the semantic specification of Web service, peer interfaces and peer advertisements.

Manuscript received January 10, 2006.

J. Cardoso is with the Department of Mathematics and Engineering, University of Madeira, 9050-390 Funchal, Portugal (phone: 291-705-156; fax: 291-705-199; e-mail: jcardoso@uma.pt).

II. Web services, JXTA architecture, and fault tolerance

Web service computing is still in an evolving state and much research needs to be done to overcome complex issues such as fault-tolerance and availability. Whisper architecture addresses precisely these limitations. It consists on the use of a P2P infrastructure that implements fault-tolerant mechanisms to insure a high degree of availability of peers that are responsible for executing Web services invoked by clients.

A. Do Web services support fault-tolerance?

Web services do not support or make available any support for fault tolerance; only mechanisms for error handling are provided. Web services are composed of a messaging layer and a service description layer that have been standardized to ensure interoperability with the Simple Object Access Protocol (SOAP) and the Web Services Description Language (WSDL). Both layers only provide mechanisms for error handling. At the messaging layer, SOAP provides a <soap:fault> tag to inform a client about errors encountered while processing an invocation message. Similarly, the WSDL description layer provides the <wsdl:fault> tag which specifies the abstract message format for any error that may be output as a result of a remote operation invocation. The mechanisms provided by SOAP and WSDL help handling errors raised by applications, but no mechanism exists for handling failures and system errors [2].

B. Implementing fault-tolerance using a JXTA

In our implementation we have selected the JXTA [3] infrastructure to deploy a fault-tolerant peer-to-peer back-end architecture due to: (a) the dynamic nature of the networks that can be created, (b) the existence of the peer group concept, and (c) its level of decentralization.

(a) Dynamic networks. JXTA networks are inherently dynamic. By using a number of protocols, 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 execution of Web service requests.

(b) Peer groups. Peers groups are important for Whisper architecture since they allow the implementation of the concept of semantically equivalent peers. Peers that belong to a given semantic group implement the same functionality, but possibly in a different way.

(c) Decentralization. Web Services are based on a centralized model and primarily focused on standardizing messaging formats and communication protocols. JXTA 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.

C. *Fault-tolerance and redundancy*

Redundancy has long been used as a means of increasing the availability of distributed systems. 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. Using this approach a Web service invocation can be forwarded to peers located in different computers and networks.

III. ARCHITECTURE

To facilitate the understanding of Whisper architecture we describe a running scenario which is illustrated in Figure 1. The application shown has two Web services available to clients: 'Register Student' and 'Student Information'. The 'Register Student' service accepts a data structure describing a student (i.e., its name, address, degree, etc.), connects to a relational database, stores student's data, and returns the ID of the newly created student record. The service 'Student Information' 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 these two Web services is not provided with the Web services themselves, but it is provided by a JXTA network of peers. Each peer belongs to a semantic peer group. The peers of the same semantic peer group implement the same service functionality, but possibly in a different way. When a Web service is invoked by a client, Whisper dynamically tries to find a semantic peer group that will be able to process the invoked Web service.

The mechanics behind Whisper architecture are carried out in the following sequence. When a business partner (client) requires a certain functionality to be satisfied, it invokes or initiates an interaction with a Web service (1) by creating a SOAP request message (our prototype was implemented using Axis SOAP 1.1 for Java (http://ws.apache.org/axis/)). The Web service receives the request and forwards it to the Semantic Web Service proxy (SWS-proxy) (2). The proxy contacts the JXTA infrastructure (3) and using the Semantic Discovery Service (4) locates a semantic group of peers (5) that can satisfy the client's request. Once a suitable semantic group of peers is found (6), the group is queried to find a peer (7) that will process the client's request. Since peers implement the Bully algorithm [4] (we call the peers that implement the Bully algorithm b-peers), 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 (8). The proxy translates the data received to a suitable format and sends the results to the semantic Web service (9) that will in turn send the results back to the client (10) that initially issued the request. In the following sections we describe the most relevant components of our architecture.

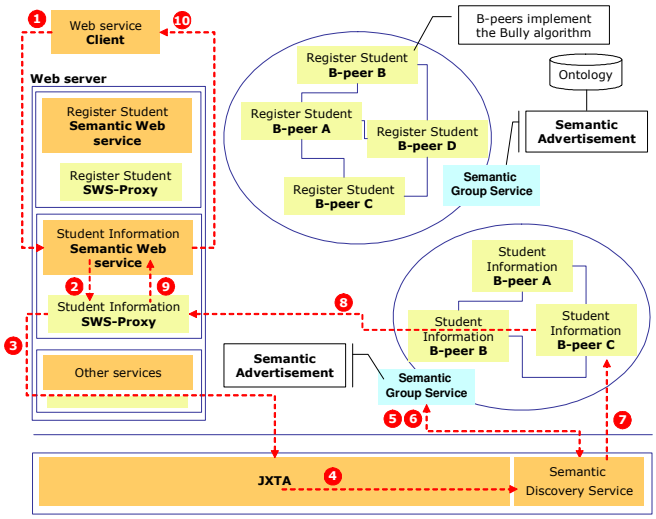

Fig. 1. Whisper architecture for Fault-tolerant Web Services

A. *Semantic Web services*

Traditional Web services are described using the Web Services Description Language (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 [5].

Several researchers have pointed out that Web services should be semantically enabled [6-8] to develop distributed applications over the Web due to its heterogeneity, autonomy, and distribution. Semantics articulate a well-defined set of common data elements or vocabulary allowing a rich description of Web services which can be used by computers for an automatic or semi-automatic processing and

management of distributed applications. In Whisper, Web service are semantically annotated following the WSDL-S specification [9]. 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.

1) Data Semantics

Web services and JXTA peer services 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. For example, a Web service may contain an input structure called 'client' which includes the 'name', 'address', 'city', 'country', and 'telephone' of a client, while a JXTA peer service may have an input structure called 'customer' and subdivides it into 'first name', 'last name', 'address', and 'tel'. In such a scenario, how can the data input of the Web service be transferred to the input of the peer service? While the two structures do not match syntactically, they match semantically. To allow the integration of Web services and JXTA peer services 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 [6, 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) Functional Semantics

As seen previously, services are described using input and output data, but also operations (i.e., methods or functions). 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. For example, two services (Web services or JXTA peer services) can have an operation with the same signature even if they perform entirely different functions. As a step towards representing the functionality of services, in Whisper, Web services and JXTA peer services are annotated with functional semantics. We achieve functional integration by having a Functional Ontology in which each concept/class represents a well-defined functionality. The Functional Ontology was created using Protégé Ontology Editor and it was modeled using the OWL (Web Ontology Language) specification.

The reader is refereed to [6] for a comprehensive description on the use of semantics to integrate information systems at the data, functional, and operational levels.

3) Using WSDL-S

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. We use WSDL-S [9, 10] to semantically describe Web services. WSDL-S establishes mapping between WSDL descriptions and ontological concepts.

To create, represent, and manipulate WSDL-S documents, WSDL4J (http://sourceforge.net/projects/wsdl4j/) can be used. WSDL4J provides JAVA API's for WSDL parsing and generation. WSDL4J supports extensibility elements providing an easy mechanism to add new extensions. This allows WSDL to represent a specific technology under various elements defined by WSDL. Given a WSDL-S specification, Whisper generates a java file with specific methods to access the semantics of the inputs, outputs, and actions (i.e., operations). These methods will be used by SWS-proxies to retrieve the semantic information of Web services.

B. SWS-Proxies

Our semantic Web services are JXTA-enabled. They do not contain an implementation coded by programmers; instead they contain a module, called SWS-proxy (Semantic Web Service-proxy), which is automatically generated by the proxy generator (see Fig. 1). When a Web service is invoked by a client the request is forwarded to the SWS-proxy.

SWS-proxies are modules that provide the communication between Web services and JXTA semantic peers. While proxies are widely used in other contexts (Web servers proxies, firewall proxies, etc.), our approach, however, uses proxies to discover semantic peer groups advertisements and enable the translation of Web service invocations to JXTA peers invocations, since the two technologies use incompatible communications protocols. When advertisements that have the same semantic functionality (see section III.A.2) 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 III.A.1) 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.)

C. Semantic Advertisements

In Whisper, peer groups semantically advertise to other peers the services they provide to the network. This is an important feature, because it creates a dynamic environment where the network and services available can be discovered and used as they are created.

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. Semantic advertisement includes information to allow the semantic integration at the data and functional levels of Web services and JXTA peer groups. This information includes the action, input, and output.

In order to have the semantic peer group advertisement recognized upon discovery, it needs to be registered through the 'AdvertisementFactory'. Once our new type of advertisement is registered, we need to specify the semantics that will describe the peers that belong to a specific group. Finally, the advertisement published can later be searched by WS-proxies. A semantic advertisement is created and semantic information is added to the advertisement to specify the ontology used to describe the ontological concepts, the action that the peers of the group will have, and the inputs and outputs of the peers.

D. B-peers

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. B-peers exist independently and communicate with other b-peers asynchronously.

The Bully algorithm insures that there is exactly one coordinator in a group of semantically equivalent b-peers. If a b-peer that is not the coordinator crashes, the semantic group continues to answer to requests. It can happen that the b-peer coordinator goes down leaving the semantic group without a leader to manage the group. In such a case, the algorithm starts an election process that lets a group appoint a new coordinator. An election is started by sending an election message to all b-peers in the semantic group. The algorithm guarantees that the b-peer with the greater peer-id will be the new coordinator. An election is held whenever a b-peer restarts or joins the semantic group or when the current coordinator crashes. Therefore, a semantic group of b-peers is able to continue operating even if one or more b-peers crash. While we have implemented the Bully algorithm to achieve a higher level of availability, other algorithms can also be implemented, without any impact on Whisper architecture, to protect Web services against other types of failures. For example, the N-version model [11] can be used to protect Web services against Byzantine faults.

IV. Related work

Dialani et al. [2] describes an architecture to deploy fault-tolerant Web services. In their approach, Web services extend a checkpoint and rollback interface. The checkpoint interface allows Web services to store their last correct state, while the rollback interface allows, in case of failure, to restore the service to its last correct state. Looker et al. [12] propose WS-FTM (Web Service-Fault Tolerance Mechanism), an implementation for Web services of the N-version model [11] for fault tolerance. WS-FTM achieves transparent usage of replicated Web services by use of a modified stub. The stub is created using tools included in WS-FTM. Whisper differs from previous work 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.

V. Conclusions

In this paper, we have presented 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. Since Web services and WSDL do not provide any mechanism to increase their availability, we have use JXTA to deploy a fault-tolerant peer-to-peer back-end architecture. In Whisper, Web services are semantically enabled and implement a semantic proxy that dispatches Web service requests to b-peers (these peers implement the Bully algorithm). B-peers are then responsible for answering to requests. The integration and interoperation of Web services and JXTA peer-to-peer networks is a difficult task since due to the heterogeneity of the two technologies there is a disagreement about the meaning, interpretation, or intended use of the same or related data and functions. To facility this integration and interoperation we rely on the technological foundations of the semantic Web.

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

[4]. Garcia-Molina, H., *Elections in a Distributed Computing System.* IEEE Transactions on Computers, 1982. **31**(1): p. 48-59.

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

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

[7]. Martin, D., et al. *Bringing Semantics to Web Services: The OWL-S Approach*. in *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*. 2004. Diego, California, USA: Springer-Verlag.

[8]. Fensel, D., C. Bussler, and A. Maedche. *Semantic Web Enabled Web Services*. in *First International Semantic Web Conference*. 2002. Sardinia, Italy: Springer-Verlag.

[9]. Rajasekaran, P., et al., eds. *Enhancing Web Services Description and Discovery to Facilitate Composition*. International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), ed. A.S. Jorge Cardoso. Vol. LNCS 3387. 2004, Springer-Verlag Heidelberg: California, USA. 147.

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

[11]. Pease, M., R. Shostak, and L. Lamport, *Reaching Agreement in the Presence of Faults.* Association of Computing Machinery, 1980. **27**(2): p. 228-234.

[12]. 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.