

FT-Net Traveler: Fault-Tolerant and Scalable Web Service Broker Architecture

Hillary Caituiro-Monge, Manuel Rodriguez-Martinez

Computing and Information Sciences and Engineering, University of Puerto Rico at Mayagüez
{hcaituiro, manuel}@ece.uprm.edu

Abstract

Web services are becoming the ultimate response for organizations and companies offering their core business services over the Internet. In fact, those business services are being offered through Web services which act like APIs providing the logic to handle a business service. Although, some Web services do not require other Web services to accomplish their tasks, the most interesting ones do the opposite. In that sense, Web services are orchestrated to create new complex business services that can reach Web services of several companies. For the purpose to handle it we proposed Net Traveler that is an autonomic Framework for collaboration, orchestration and choreography of Web services. Some design requirements are critical in the composition of Web services. Reliability and scalability among others must be done in order to offer a minimum of quality of service for composite Web services. In this paper we present the architecture of FT-Net Traveler which provides reliability through fault tolerance and scalability through load balancing to composite Web services without requiring modifications on them.

Keywords

Web services, Fault tolerance, Scalability.

1. Introduction

Web services are broadly being used among organizations and companies around the world to expose their core services through the Internet. It is the ultimate response for the problem of integration [1] and collaboration [2] of heterogeneous information systems and data sources that in fact allow them to work together in a consistent manner. Furthermore, it is being used at a higher level, thus enabling enterprises to pipeline their business processes among them [3].

Nowadays, organizations in general are putting some part of their business process in the Internet through Web services. It makes Web services the most suitable information technology to support business-to-business (B2B) interactions [4] [3] [5]. Figure 1 depicts a basic interaction diagram between a client and Web services. This client makes a request to a Web services and it contact other Web services to complete the request.

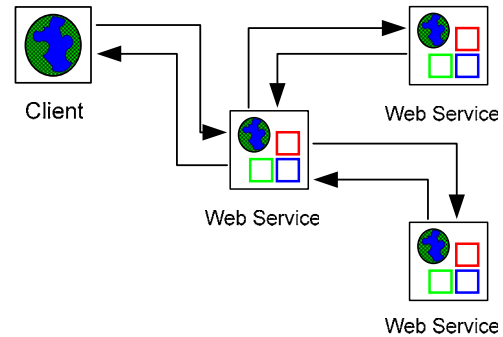


Figure 1 Basic interaction between a client and Web services

The growth and importance that Web services are gain, lead its use in a spectrum of applications with different design requirements. The main goals that designers pursue in the design of Web services are transparency, scalability, reliability and performance. Of these, the most challenging ones are scalability and reliability. Fault tolerance is one means of achieving reliability.

Fault tolerance is the ability of a system to continue operating as expected, despite internal or external failures. Some failures can be detected, such as hardware or software crashes and unavailable communication channels. Other kind of failures cannot be detected, such as logic design failures. When failures are detectable, the fault tolerance of a composite of Web services can be improved through

redundancy, i.e., replication of its hardware or software components.

Web services are reactive since they initiate a response as a reaction to an external stimulus such as a SOAP message. A Web service is asynchronous if any other Web service whose stimuli cause a reaction in it does not have to wait for such reaction to continue its execution; otherwise, a Web service is said to be synchronous. A Web service is non-deterministic if, given one type of stimulus, there is more than one possible type of reaction; otherwise, the Web services is said to be deterministic.

There is a challenge to achieve fault tolerance in non-deterministic Web services. In non-deterministic Web services; the output could be different even if the same sequence of stimuli is input with the same initial state. Since the Web services is asynchronous, timing assumptions are not valid.

Existing fault-tolerance techniques have been implemented using components such as failure detectors; state transfer protocols; and duplicates detection/suppression mechanisms. Failure detectors use timing assumptions that are valid only with synchronous or semi synchronous systems, semi synchronous systems are asynchronous systems with timing assumptions. State transfer protocols are valid only with deterministic systems; also they are very intrusive, changing definitely the system behavior. Duplicate detection and suppression mechanisms use encoding techniques based on automated sequenced code assignment, which depends on determinism and synchronism.

This research is about reliability through fault tolerance and scalability through load balancing in the composition of Web services without requiring modifications on the individual Web services.

1.1. Contributions

The main contribution in this work is the fault tolerant and scalable architecture for the main component of Net Traveler. By means of it, also is achieved the fault tolerance for composite Web services.

1.2. Roadmap

In the second section of this paper we present an overview of the Net Traveler Architecture to explain their main components and some terminology used on it. In the third section we introduce our approach to address the reliability and scalability in Net Traveler. Finally, we have the related work, conclusions and future work.

2. Net Traveler Architecture Overview

Web Services are being deployed everywhere and offering a wide spectrum of services. Also, organizations are publishing those Web Services as an open infrastructure. For example, Amazon had published ECS (Amazon E-Commerce Service) enabling developers interact with it through standard protocols. The building blocks of our Framework are Web services. By composing them, can be built complex service-oriented applications (SOA).

2.1. WS-Net Traveler architecture

Net Traveler's architecture to support autonomic business processes is a non-centralized one. In this section we show Figure 2 shaping the architecture and also we describe each component.

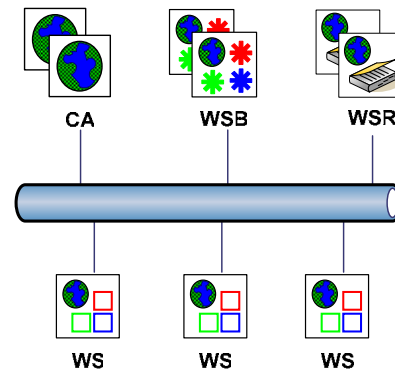


Figure 2 WS-Net Traveler Architecture

2.1.1. Web Service (WS). In our architectural view, Figure 2, WS represents a Web service. It can be described as a software component made accessible over the Internet. A WS is intended to be machine-understandable and to achieve this purpose it uses URI / UDDI / WSDL / SOAP and other specifications. By means of these standards WSs can be adequately identified, discovered and defined, also they are able to interchange data.

2.1.2. Web Service Broker (WSB). WSBs, in general, are responsible for the completion of a cross Web Services Business Processes. Using P2P techniques they deal with discovery and negotiation. They can be plugged with software components to expand their functionality. These extra capabilities include autonomic management, fault tolerance, transaction management and load balancing. WSBs are deployed by organizations owning WSs or also by organizations which their business is to offer Web

Services brokering services. WSBs can be grouped in permanent and ad-hoc federations. [2] They have decentralized coordination.

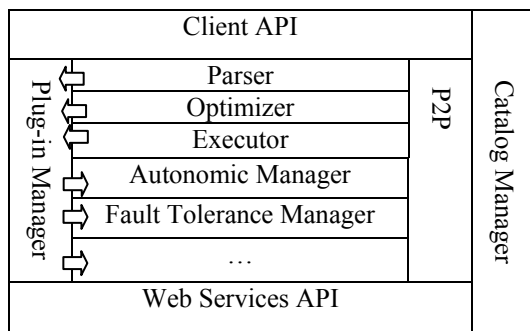


Figure 3 Web Service Broker Architecture

WSB architecture (see Figure 3) includes several core components which are in charge of basic functions and pluggable components which can be installed, upgraded and removed. The first component is the *Client API* which is in charge of the interaction with clients that can be WSB, WS or other applications. The second component is the Parser, which is in charge of interpreting and organizing the request in a execution plan. In third place is the Optimizer that optimizes the plan in accordance with their local information and the information of the neighbors WSBs. If there is something that needs to be executed whiting the context of the QSB the Executor component does it.

A QSB interacts with the Catalog Manager to find other QSB. And, the Catalog Manager, if necessary, triggers discovery mechanisms or it gets Web services definitions from the Web service registration server or another component holding WS descriptions. The P2P communication component is in charge of P2P interactions among WSBs. Finally the Plug-in Manager is in charge of install, update and remove new component in the WSB. Some important pluggable pre-installed components are the Autonomic Manager, the Fault Tolerance Manager and the Transactional Manager.

2.1.3. Web Service Registration (WSR). Additionally, this architecture presents a component in which WSs can be registered. It is an alternative way to make WSs visible to others for use. Also, for collecting WSs can be used P2P techniques for discovery.

2.1.4. Web Service Client Access (CA). CAs, in this architecture, represent front-end web -based or non web-based GUIs which capture users' requests. They

are optional, because WSs and WSBs do not rely on them to perform their activities.

2.2. WS-Net Traveler global architecture

Figure 4 depicts the AWS-Net Traveler Global Architecture. This shows how the components of the AWS-Net Traveler Architecture are deployed whiting a wide-area environment like the Internet. The big circle is a Local Group (LG) where components are grouped usually by functionality. A LG can be composed by several WSs, WSBs, CA, WSRs. Also, a LG includes the orchestration of these components as defined in Caituiro et al. work [2].

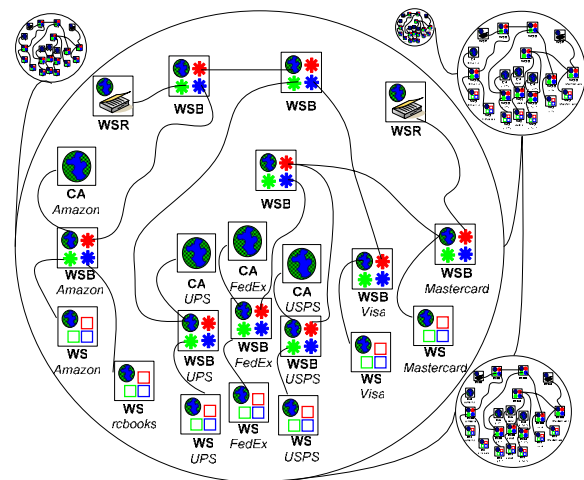


Figure 4 AWS-Net Traveler Global Architecture

Figure 4 also shows small circles which are other Local Groups (LG). Different LG can be linked by mean of the cross relationships among their members.

A *federation of Web Services (FeWS)* is a collection of WSs grouped with some purpose. This usually is for functionality or to solve users' requests. For example a federation is created to allow potential vacationers to plan a vacation or regular student to matriculate in a new season. For a vacation planning the federation members are the airline company Web Service, the rent-a-car company Web Service, and the hotel company Web Service.

A *family of Web Services (FaWS)* is a collection of Web Services having an equivalent functionality but they are not necessarily identical. For example Expedia Travel and Travelocity offer equivalent travel services.

A *choreograph* is the execution plan, which is created for each user request. A difference of an

orchestration that is fixed, a *choreograph* changes for every user request. [2]

A Web Service is *equivalent* to another if it is capable de provide an equivalent service. An equivalent service is one that for example if we want to travel from San Juan, PR to Lima, Peru allows us to do that with one Web service and their equivalent.

3. Fault-tolerant and scalable Web service broker architecture

One approach for achieving scalability and fault tolerance in Net Traveler is distribution and replication. Scalability can be achieved by distributing Web service brokers load over several replicated WSBs instances. Fault tolerance can be achieved by means of WSBs replication. The architecture of this approach is presented in Figure 5.

In the distribution (horizontal) dimension, N WSBs instances are distributed, forming a distribution group, while in the replication (vertical) dimension, each distributed WSB has M replicas, where all replicas form a replication group. Let Δ be a distribution group $\{\delta_1, \delta_2, \dots, \delta_N\}$, where each δ_i is a WSB instance. Each δ_i is replicated by a replication group $\Gamma_i \{\tau_{i1}, \tau_{i2}, \dots, \tau_{iM}\}$ where each τ_{ij} is a replica of the i_{th} WSB instance. A workload from a given domain is partitioned into N disjoint subsets, and distributed over N WSB instances of Δ . Such a distribution should be done for workload distribution. Load distribution is affected by many factors, such as priority, interdependence, timing and performance.

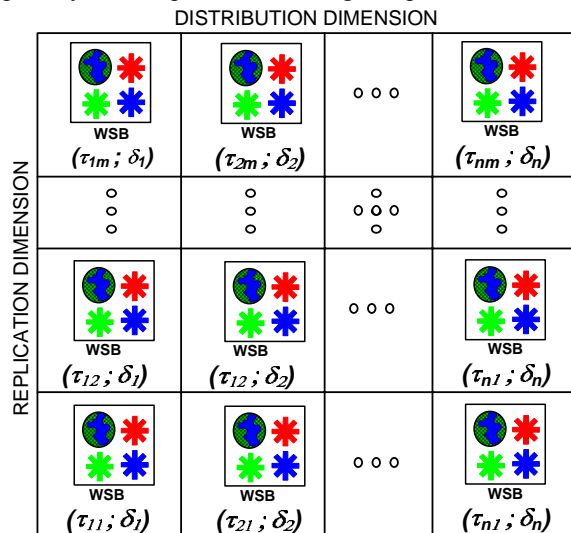


Figure 5 Fault-Tolerant and Scalable Web Service Broker Architecture

In the case of replication (vertical) dimension, we have the WSB replication. For WSB replication, we have two alternatives: passive or active replication. In passive replication, only one replica is active at the same time. When it crashes, another inactive replica is activated. Before activation, it gets the state from the crashed replica log file. One disadvantage of this approach is that the operation for recovering the state is very expensive in time. In active replication, all replicas are running and are active at the same time. All replicas reply to a given request, but only one reply is returned to the client and the others are stopped. Upon a crash, there is no recovery overhead, because it is not necessary switch to a new replica. The disadvantage is the number of computational resources involved; however, cost reduction in time is more important than cost reduction in resources.

4. Related work

The OMG (Object Management Group) [7], which standardized CORBA, has recently published specifications for Fault-tolerant CORBA. The framework is based replications of objects, fault detection, and recovery. This standard has been adopted in our work.

Birman et al. [6] in the paper “Adding High Availability and Autonomic Behavior to Web Services” present an approach extending the Web Services Architecture with additional components for Fault Detection, Enhanced Communication, and High Assurance. Although, service consumers must remain unmodified for the basic form of fault tolerance, services have to be modified with additional components. Furthermore, for an advanced form of fault tolerance both service consumers and services have to be modified in order to implement the features introduced in this solution. Net Traveler, our approach, includes a Web service broker, a middleware component among web services, does not require service consumers or services to be modified. Indeed extending Web service stack architecture is an optimal solution, currently, there are a large amount of published Web services which are already deployed. We intend to include those installed Web services as part of our solution without requiring any modification on them. Additionally, we believe that separating Web services which represent some service from Web services which represent management functions is a desirable outcome in an service oriented architecture with loosely coupled components.

Issarny et al. [8] propose a solution based on forward error recovery. This solution is oriented

towards providing dependability of composite Web services. As desirable this approach has no impact on the autonomy of the individual Web services. A difference our proposed architecture provides support for both reliability and scalability at the same time.

5. Conclusion and Future Work

In this paper we presented a reliable and scalable architecture for composite Web services by means of enabling fault tolerance and load balancing in Net Traveler. Currently, we are implementing a prototype and a simulation to validate our architecture.

6. References

- [1] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In Proc. of IPSJ Conference , Tokyo, Japan, 1994.
- [2] H. Caituiro-Monge, M. Rodriguez-Martinez, "Net Traveler: A Framework for Autonomic Web Services Collaboration, Orchestration and Choreography in E-Government Information Systems", ICWS, IEEE, San Diego, California, 2004, pp. 2-10.
- [3] B. Medjahed and B. Benatallah and A. Bouguettaya and A. H. H. Ngu and A. K. Elmagarmid, "Business-to-business interactions: issues and enabling technologies", The VLDB Journal, Springer-Verlag New York, Inc., 2003, pp. 59-85.
- [4] K. Hogg and P. Chilcott and M. Nolan and B. Srinivasan, "An evaluation of Web services in the design of a B2B application", CRPIT '26: Proceedings of the 27th conference on Australasian computer science, Australian Computer Society, Inc., Dunedin, New Zealand , 2004, pp. 331-340.
- [5] H. Kreger, "Fulfilling the Web services promise", Communications ACM, AC M Press, volume: 46, number: 6, 2003, pp. 29-ff.
- [6] K. Birman, R. v. Renesse, W. Vogels, "Adding High Availability and Autonomic Behavior to Web Services", ICSE '04: Proceedings of the 26th International Conference on Software Engineering, IEEE Computer Society, 2004, pp. 17-26.
- [7] Object Management Group, "Common Object Request Broker Architecture: Core Specification", published by OMG, December 2002.
- [8] V. Issarny, F. Tartanoglu, A. Romanovsky, N. Levy, "Coordinated forward error recovery for composite Web services", 22nd International Symposium on Reliable Distributed Systems, IEEE, 2003, pp. 167- 176.