A QoS-AWARE WEB SERVICE REPLICA SELECTION FRAMEWORK FOR AN EXTRANET

Kambiz Frounchi
Partheeban Chandrasekaran
Jawid Ibrahimi
Department of Systems and Computer Engineering
Carleton University, Canada
email: {kfrounch, pchandr3, jmohamad}@connect.carleton.ca

Shikharesh Majumdar
Chung-Horng Lung
Department of Systems and Computer Engineering
Carleton University, Canada
email: {majumdar, chlung}@sce.carleton.ca

Laura Serghi
Alcatel
Ottawa, Canada
email: laura.serghi@alcatel.com

Abstract

Business partners in an extranet expect web services to be offered in a timely fashion. Replicating web services may reduce the end-to-end web service response time and add fault tolerance to the system. In this paper, we propose a web service replica selection framework that can be deployed in an extranet. The selection procedure is based on dynamic factors such as communication delays between the web service provider and the web service requester and the workload of the web service provider. To analyze the performance of the framework, two web service replica selection policies are simulated: minU and random. Simulation results show that the minU policy has better response time especially under high load conditions.

Keywords: *Extranet, Web Services, Web Service Replica selection, Web Service Replica Selection Broker, QoS.*

1. Introduction

An extranet allows a company to communicate securely with its business partners using the Internet technology. Major organizations are forming extranets with their trading partners to facilitate their transactional business-to-business (B2B) activities and automate their administrative interactions with their suppliers, customers and other businesses [1]. Web services are emerging standards of the service-oriented architecture, providing means to incorporate web applications independent of programming language and execution environment. Web service technology is being highly adopted to provide various B2B services.

Replication of web services in a geographically distributed network as an extranet may decrease web service response time and achieve fault tolerance. A number of research efforts have focused on web service discovery but comparatively little attention has been paid to the devising of an actual framework that could be employed for web service replica selection. A client-side based approach has been investigated in [2]. They propose a framework on the client side to perform the web service replica selection. However this places the overhead of performing the web service replica selection on the web service requester side. In [3], an algorithm is proposed for dynamic replica selection in a CORBA-based middleware deployed in a local area network (LAN). Their work is based on a distributed information repository that stores all the performance parameters such as gateway-to-gateway delay and the workload of the server. Although this may be easy to implement in a LAN, parameters such as gateway-to-gateway delay may be hard to monitor in a wide area network such as an extranet. Moreover, issues associated with updating the information repository such as maintaining the validity of the information is not raised in their work. In this paper, we investigate a broker-based solution for web service replica selection that could be employed in an extranet environment. The broker is a plug-in software module deployed in the gateways of the extranet which selects the appropriate web service replica based on a selection policy. The selection policy is designed to consider factors such as the communication delays in the extranet and the workload of the service providers. Our framework is QoS-aware in the sense that the selection policy attempts to minimize the response time for the web service requests. This paper presents some preliminary research performed in the context of a collaborative project between Alcatel and Carleton University.

The rest of the paper is organized as follows. Section 2 describes the web service replica selection framework. Section 3 introduces the simulation model of the framework. The performance of two selection policies is analyzed in Section 4 and we conclude the paper in Section 5.

2. System Architecture for the framework

In this section we present the traditional web service architecture and an overview of our framework.

2.1. Extension of the Web Service Architecture

The traditional web service architecture consists of three components: a service provider, a service requester and a Universal Description, Discovery and Integration (UDDI)

registry. The service provider publishes its web services to the UDDI registry. The service requester searches the UDDI registry to find a binding for a web service. It then binds to the service provider and the two parties start interacting with each other (see Figure 1).

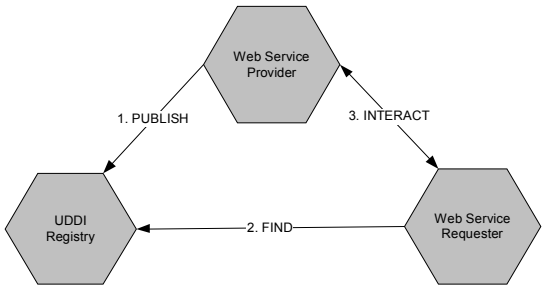

Figure 1: The basic web service architecture [4].

Our framework extends the traditional architecture by introducing a middleware that chooses the appropriate web service based on dynamic factors such as the service provider workload and the communication delays. Although primarily targeted at extranets, the framework could be employed in other distributed environments as well.

In this framework, the web service requester does not directly interact with the UDDI registry to find a web service. It sends a request for a web service to a broker. The broker finds the appropriate web service and chooses a web service replica if the web service is replicated and forwards the request to the web service provider. The selected web service provider executes the web service and sends the response back to the web service requester.

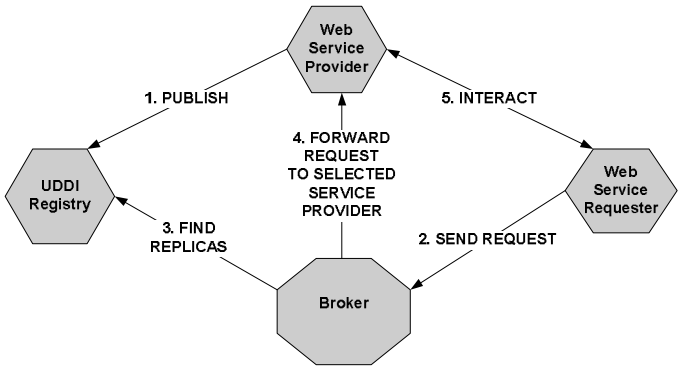

Figure 2: Modification to the current web service architecture

In this paper we focus on a web service replica selection broker. Figure 2 shows how the web service requester could interact with the appropriate web service replica provider using the framework. The broker chooses a replica based on a web service replica selection policy.

2.2. The Replica Selection Framework

There are six components in the context of our proposed framework: the Web Service Requester (WSR), the Web Service Provider (WSP), the UDDI Registry, the Gateway that includes a Broker and a Routing Module. The web service provider typically refers to a server, running in a company's network providing a particular web service replica. A single WSP can host multiple web services. The gateway component provides the functionality of a normal router. The web service replica selection is also done inside the gateway by the broker component. A WSR models the arrival of web service requests and also serves as a sink for the responses. Figure 3 shows the basic flow with reference to how the replica selection is carried out in an extranet. The broker and routing modules are not included in the diagram to avoid cluttering.

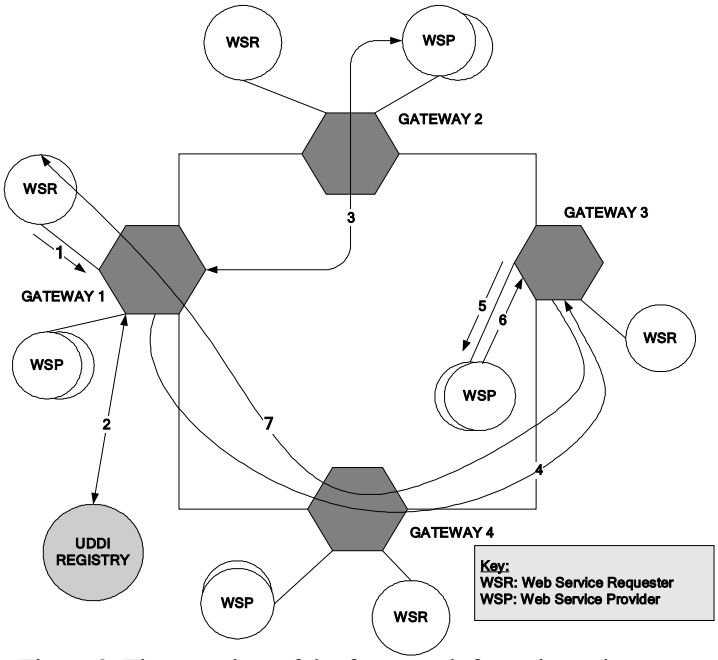

Figure 3: The overview of the framework for web service replica selection in an extranet.

A brief description of the flow of messages among the different system components is provided below.

1. The WSR sends a web service request to the Broker in the gateway.
2. The gateway then obtains all the addresses (Uniform Resource Identifier, URI) of the WSPs offering that specific service from the registry [5].
3. An information policy is used to get the status of the WSPs and the communication delays between the gateways. This step is explained further in Section 2.3. Note that for avoiding cluttering we have shown the polling of only one WSP in Figure 3.
4. The request is then forwarded to the gateway with which the selected WSP is associated.
5. The receiver gateway then forwards the request to the appropriate WSP.

6. The WSP services the request and sends a response back to the gateway it is connected to.
7. The gateway then forwards the response to the WSR.

Note that depending on the route more than one gateway may be involved in delivering a message to a WSR or WSP (see Figure 3).

2.3. Broker

The web service replica selection is performed in the broker component which is a software module residing within a gateway. The internal plug-in policies determine the web service replica selection. Our framework is modeled in such a way that it allows for these policies and criteria to be easily altered or added as required. Figure 4 shows the internals of the broker.

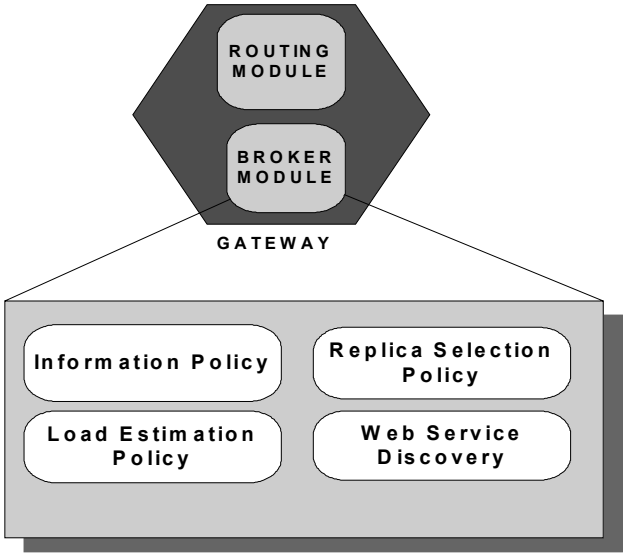

Figure 4: The broker and the routing module inside a gateway

Information policy includes the methods to obtain load information of web service providers and to estimate the communication delay in links between the gateways. Load Estimation policy is used to estimate the current state of the WSP and the links between the gateways [6]. Moreover, additional functionalities such as a cache can be added in the broker to minimize the number of requests to the UDDI registry.

Various replica selection policies may be used in different environments. For example, replica selection may be based on the geographical location of a WSP or it may be based on achieving a particular performance objective. An example is captured in equation (1) where the objective function is to minimize the following utility function.

$$U = (a \times WSPload) + (b \times CommDelay) \qquad (1)$$

a is the weight associated with the load of the WSP (*WSPload*), *b* is the weight associated with the communication delay (*CommDelay* is the communication delay between the gateway and the WSP). The replica that gives rise to the smallest U is selected for satisfying the request of the WSR. Methods for computing *WSPLoad* and *CommDelay* are discussed further in the following section.

3. Simulation

Discrete event simulation was used to model the proposed framework, using C++ as the implementation language. The simulation model consists of the six extranet components in the framework plus an additional link component that models the congestion in the links and the propagation delay between the gateways. In order to study the performance of our web service replica selection framework, a minU selection policy was designed and compared with the random selection policy.

MinU

The features of the plug-in components for the broker used in this selection policy are given below:

Information Policy: The workload of a WSP is calculated using equation (2):

$$WSPload = (LQ \times \hat{s}) \qquad (2)$$

where LQ is the queue length of the WSP and $\hat{s}$ is the average service time for a web service request. These values are obtained by polling the WSPs. The communication delays to the WSPs are gathered as round trip times. Each time, the broker receives these parameters, it uses them to calculate the utility function (see equation 1) for a particular iteration.

Load Estimation policy: To reduce time wasted in waiting for the workload information and the communication delay, we used an exponential average of the utility function. Equation (3) shows how to predict the utility function of the nth iteration based on the two previously computed utility functions:

$$EU(n) = p \times U(n-1) + (1-p) \times U(n-2) \qquad (3)$$

where *EU(n)* is the estimate for the utility function for the nth iteration. *U(n-i)* (i=1,2) is the utility function computed using equation (1) in the $(n-i)^{th}$ iteration and p ($p \leq 1$) reflects the weight given to the most recently computed utility function.

Web Service Discovery: Upon receiving a web service request, the broker sends a discovery request to the UDDI registry which then returns the addresses of the WSPs providing the web service. This information is used by the broker to find out which WSPs to be polled. We did not model the publishing of the web services by the web service providers as it is outside the scope of this paper. The locations of the web service replicas are assigned to the UDDI registry during initialization of the simulation.

Replica Selection Policy: The WSP that gives rise to the least *EU(n)* is chosen. This selection is based on the

communication delay between the gateways and the WSP's workload. If there are more than one WSP giving rise to the same *EU(n)*, then one WSP is chosen randomly.

Random

In order to compare the performance of the minU policy, we simulated the random selection policy which only performs web service discovery and randomly chooses a WSP among a set of WSP replicas.

4. Sample Results and Analysis

A sample set of initial simulation results is presented in this section. More results are forthcoming [7]. The open model simulation is conducted with four gateways, four homogeneous WSPs and four WSRs (see Figure 3). A ring topology is used for interconnecting the gateways. A Poisson process is used to model the arrival of requests at a gateway. Each WSR generates requests with the same mean inter-arrival time. A centralized UDDI Registry is used and three different types of web service requests are generated with equal probability. All the WSPs provide the three web service types, thus there are four replicas of each web service type in the system. Each web service type is characterized by a given mean service time (see Table 1). The service time for the web services are normally distributed which enables us to effectively control the variability of service times. The link service times (see Table 2) are exponentially distributed and are based on the measured round trip times for internet ping messages sent to different locations in the world from our lab. The values of *a*, *b* and *p* are 0.6, 0.4 and 0.5 respectively. Thus a higher weight is associated with *WSPLoad* in comparison to *CommDelay*. This is appropriate for a typical system in which the server load is higher than the loads of the communication links. Table 1 and 2 show the simulation parameters. These values characterize typical systems in which computation times associated with the web services are higher than the communication link service times. Note that service discovery requires some processing, thus the UDDI registry service time is chosen to be higher than the communication link service times. Results of experiments using other system and workload parameters are forthcoming [7].

Table 1: Simulation parameters: Web Services

Web Service Type	Mean Service Time (ms)	Std. Dev (ms)
1	4000	500
2	6000	750
3	8000	1000

Table 2: Simulation Parameters: System Components

Component	Mean Service Time (ms)
Link 1	230
Link 2	20
Link 3	100
Link 4	140
UDDI Registry	500

Figure 6 compares the minU and the random selection policies in terms of web service response times under different load conditions.

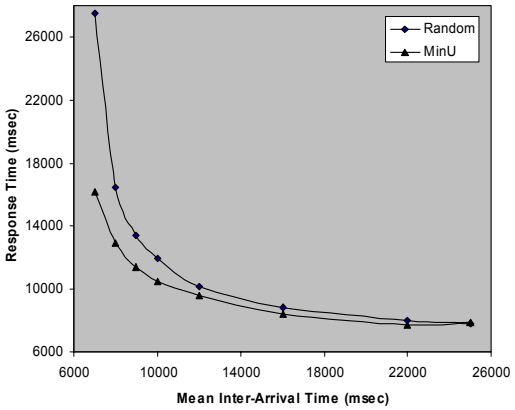

Figure 6: Response Time Comparison of minU and random selection policies

The minU replica selection policy performs better in terms of response time especially at high loads as shown in Figure 6. This demonstrates the effectiveness of using knowledge of *WSPload* and link delays that are not considered by the random policy in replica selection. However at very low loads, both the policies give rise to similar performances.

5. Conclusions and Future Work

In this paper we proposed a web service replica selection framework extending the traditional web service architecture by introducing a broker component which performs the selection procedure. The selection procedure is transparent to the web service requester, and avoids the need for the web service requester to directly interact with the UDDI registry. In selecting the web service replica the broker considers dynamic factors such as the communication delays between the gateways and the workload of the web service providers, thus achieving a high system performance. Since an extranet is a geographically distributed network, it is logical to place the broker module inside the gateways. Simulations were conducted to analyze the performance of the minU web service replica selection policy. The simulation results presented in this paper showed that the minU policy performed better than a

random policy especially under high load conditions. Further expriments with various different parameter combinations are being conducted for comparing the performance of the two strategies.

Our future work consists of simulating a larger network. We also plan to investigate the impact of using different web service replica selection policies and changing topologies in the context of extranets. Furthermore, we plan to consider the effect of changing the values of *p*, *a* and *b* as well as varying the number of web service replicas and the workload parameters.

6. References

[1] J. López, J.J. Ortega and A Maña "An User Authentication Infrastructure for Extranet Applications," in *Proceedings of the 33rd Annual International Carnahan Conference on Security Technology*, Madrid, Spain, 1999, pp. 354-362.

[2] J. A. F. da Silva, N. das Chagas Mendonca, "Dynamic Invocation of Replicated Web Services," in *Proceedings of WebMedia and LA-Web 2004*, Rebeirao Preto-SP, Brazil, October 2004, pp. 22-29.

[3] S. Krishnamurthy, W.H. Sanders, and M. Cukier, "A Dynamic Replica Selction Algorithm for Tolerating Timing Faults," in *Proceedings of The International Conference on Dependable Systems and Networks*, 2001, pp107-116.

[4] G. Alonso, F. Casati, H. Kuno and V. Machiraju, "Web Services", Springer, Berlin; New York, 2004.

[5] W3C Working Group Note, "Web Services Architecture: 3.4 Web Service Discovery," February 2004. [Online]. Available at: http://www.w3.org/TR/ws-arch/wsa.pdf. [Accessed Feb. 28, 2006].

[6] P K. Sinha, "Distributed Operating Systems Concepts and Design", IEEE Press, New York, 1997.

[7] K. Frounchi, J. Ibrahimi, P. Chandrasekaran, "Web Service Replica Selection in an Extranet", Final year undergraduate engineering project report, Department of Systems and Computer Engineering, Carleton University, Ottawa, April 2006.