Web Service Deployment and Management with UPnP

Wei-Lun Huang (黃偉倫)
National Taiwan University, Taiwan, R.O.C.
Email: d97922012@csie.ntu.edu.tw

Tzao-Lin Lee (李肇林)
National Taiwan University, Taiwan, R.O.C.
Email: tl_lee@csie.ntu.edu.tw

Chiao-Szu Liao(廖喬思)
National Taiwan University, Taiwan, R.O.C.
Email: r97922085@csie.ntu.edu.tw

***Abstract*—Web service gateway [1] developed by IBM Corporation provides a single point of access and validation of web service requests. However, it doesn't reduce the complexity of web services deployment and management from the intranet's point of view. It is necessary to let web service gateway knows what happens to the supporting web service servers in the intranet to archive rapid deployment of web services. Therefore, a mechanism using UPnP protocol [2] to exchange information between web service gateway and web service server is proposed. With this mechanism, web services can be deployed rapidly and flexibly as the number of intranet business applications is growing now a day. Besides, web services load balance will be automatically achieved with this mechanism. Furthermore, this idea can also be applied on the SOA-based (Service-oriented architecture) [3] system. SOA components will spontaneously find available bindings in the intranet without complex configurations and even combine other SOA components to a new integrated service self-acting.**

***Index Terms*—UPnP, Web Service Gateway, SOA.**

I. Introduction

Web service gateway can be a bridge between web service servers and outside users. It offers an integrated thought of all provided web services in the intranet and transmits the entire requests from outside users to the corresponding web service server inside. The most important is it just loses a little performance which we will prove in chapter 4. However, the status of web service can't be updated automatically when the service provider joins, leaves or even some events occur. In other words, the status information in the web service gateway is not in real time because the corresponding settings have to be modified by web service manager. Therefore, we need to provide a tunnel between web service servers and web service gateway for message exchange. UPnP is the mechanism we used and we will explain how to do it and show if it works well.

II. Deploy And Manage Web Services With UPnP

Web service gateway developed by IBM Corporation provides a single point of access and validation of web service requests. However, web service gateway permits only the connection from the client outside the firewall to the web server inside the firewall smoothly and centralizes the deployment from web service requesters' point of view. Briefly speaking, it doesn't reduce the complexity of web services deployment from the intranet's point of view. Person who is in charge of the web service deployment and management still has to talk to the web service gateway. Therefore, a mechanism using UPnP protocol to exchange information between web service gateway and web service server is proposed. With this mechanism, web services can be deployed rapidly and flexibly as the number of intranet business applications is growing now a day.

A. Web Service Gateway

The concept of web service gateway was brought up by IBM and implemented in IBM WebSphere application server in several years ago. The structure of web service gateway is as following figure.

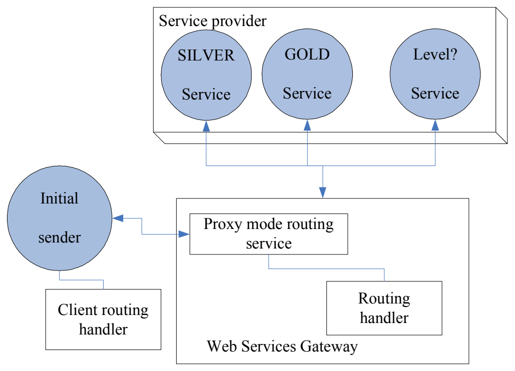

Figure 1. Web service gateway structure provided by IBM's website [4]

The main idea of web service gateway is to route the outside web service request to the corresponding web service server inside and returns the response. For example, as in figure 1, there are several web service servers as the service providers in the intranet. If the client wants to send requests to the SILVER service, this client sends its requests to the gateway, then the gateway will check its routing table, route the request to the SILVER service server and return the response from the SILVER service server. However, every time the service provider inside creates or drops a web service, it is necessary to modify the settings in the routing table of the web service gateway and export the modified WSDL file. This deployment won't be done automatically because the gateway knows nothing about the situations in the intranet.

B. Deployment and Management with UPnP

Using web service gateway is great but there are some problems to be solved. It is necessary to let web service gateway know what happens to the supporting web servers in the intranet. In order to solve this problem, every web services provider in the intranet is equipped with an UPnP server and the web service gateway is equipped with an UPnP controller. During the UPnP discovery phase, the gateway as the UPnP controller will find all web service servers as the UPnP servers in the intranet with the UPnP protocol. During the description phase, every web service server describes its web services, so the gateway can import all WSDL files from all the web service servers to generate a new WSDL file in order to provide the list of all available services to the client from internet. After these two phases, the gateway will know the routing table automatically. Deployment of web services will be done spontaneously. The architecture graph is shown below.

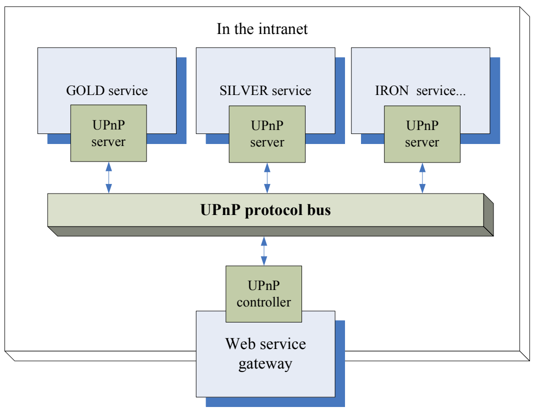

Figure 2. The architecture of web service

This is nice, but it's not over yet. What if some web services be turned off or turned on as time goes by? What if a new web services server will be added to the intranet after these two phases? Don't worry about it. UPnP protocol has a flexible mechanism to solve all of the above problems. Every web services server can monitor the status of its own web services and provides UPnP event notification which can also be subscribed by the web service gateway. Therefore no matter what the status is changed by the web services server, the gateway will be notified and can update its routing table on line. The gateway will always be updated with the current status of the intranet, so it can respond to the route request more correctly.

Distributed deployment is achieved. However, how about centralized management? Can centralized management be possible with the help of UPnP protocol? The answer is yes, and the solution is straightforward. Web services server should export the command like “Start” or “Stop” of its own services as UPnP action. During the UPnP description phase, the gateway gathers all the actions for the entire web services servers from the intranet. Henceforth the gateway can control those web services on demand.

With this mechanism, the only one deficiency is that since all requests by the web service client from the internet must be sent to the web service gateway and forwarded to the designated web services server, the bottleneck appears be in the gateway. But in fact, with this mechanism there are no restrictions on the number of web service gateway. Therefore, many gateways can be added as necessary without copying the settings from one gateway to another. When another gateway is online, deployment will be done by the UPnP procedures again.

III. IMPLEMENTATION ISSUES

A. Deployment with UPnP

It's important that web services servers should export the WSDL files during the UPnP description phase to archive zero configuration. So, every web services server will be augmented with an UPnP action named “GetWSDL” as in the following:

```
......
<action>
        <name>GetWSDL</name>
        <argumentList>
                <argument>
                        <name>WSDL</name>
                        <direction>out</direction>
                        <relatedStateVariable>
                                thisWSDL
                        </relatedStateVariable>
                </argument>
        </argumentList>
</action>
......
<stateVariable sendEvents="yes">
        <name>thisWSDL</name>
        <dataType>string</dataType>
</stateVariable>
......
```

Using this action, the gateway can get all the WSDL files from the web services servers to generate a new WSDL file. We can notice that the “sendEvents” attribute of stateVariable named “thisWSDL” is set to “yes”. Therefore, when a web services server modify its service status, the gateway will be notified and get the current WSDL again on line.

B. Management and Control with UPnP

Beside the “GetWSDL” UPnP action, web services server must export more actions like “Start/Stop service” in order to provide a way for the web service gateway to control web service. Thus, the management of all the web services can be done by the gateway itself. Furthermore, the web services server can provide extra actions to describe the status like CPU loading or memory usage. With these extra actions the gateway will know the health condition of each web services server. And the gateway can manage these web services more efficiently. It’s reasonable to conclude that, the more the gateway knows the less the configuration is needed by the developer. The centralized management is made possible with the help of UPnP protocol.

C. Achieving Web Services Load Balance

This UPnP mechanism can also provide a solution to achieve the load balance among the web services servers. If the web service gateway finds the web services provided by different web service servers with the same name ,the same parameters and even the same namespace from two WSDL files such as the following provided by different web service servers from the intranet:

```
......
<message name="thesameRequest">
        <part name="request" type="xs:string"/>
</message>
<message name="thesameResponse">
        <part name="response" type="xs:string"/>
</message>
<portType name="thisport">
        <operation name="thesameFunction">
                <input message="thesameRequest"/>
                <output message="thesameResponse"/>
        </operation>
</portType>
......
```

It is transparent that these two web service servers provide totally the same functional web service. When the request of this web service are sent to the gateway, the gateway can route it to different web service servers alternately in order to reduce the workload of more heavily loaded server. Furthermore, the gateway can not only route it alternately but with the detection of the status of web services server the gateway can route the request to the not so busy server. By this way, the web services load balancing is realized automatically. Again, the more the gateway knows, the less the configuration required.

IV. Performance Evaluation

We construct a simple web service gateway system and implement our UPnP mechanism in it for performance evaluation.

In the first evaluation, two different kinds CPU load web services (High/Low) are created. The testing web service pseudo codes are as following:

Low CPU load web service pseudo code:

```
fuction string echo(string p)
{
    return p;
}
```

High CPU load web service pseudo code:

```
fuction int sumof(int p)
{
    int sum=0;
    for(int i=0; i<p; i++)
        sum+=i;
    return sum;
}
```

Then, 1000 times web service requests are simulated with/without web service gateway architecture to test the impact of the performance on web service gateway in different situations. The response time result graph is as following figure.

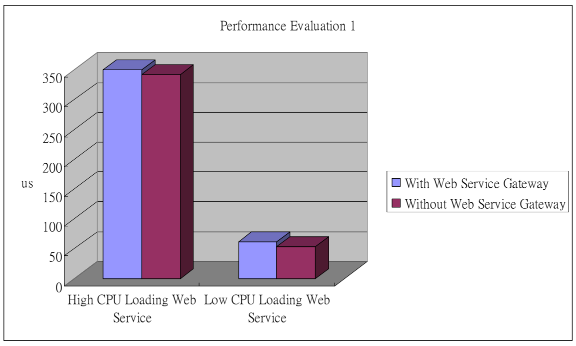

Figure 3. The average response time with/without web service gateway architecture.

We can notice that the average response time is almost similar even if the web service gateway exists whether in high or low CPU load web service. Briefly, web service gateway will not lose a lot of performance.

Next, we test the performance of the automatically archived web service load balance described in 3.3. In this test, web service gateway will automatically find there are two web service servers providing the same web service and route the requests to them. In the low CPU load web service condition, the efficiency is not improved if we used web service gateway. This is because the cost of routing request is almost similar with providing service. However, In the high CPU load web service condition, the response time of with web service gateway is shorter than without it shown in left part of Figure 4 which tells us that the automatically archived web service load balance can deal with requests more quickly in high CPU load web service.

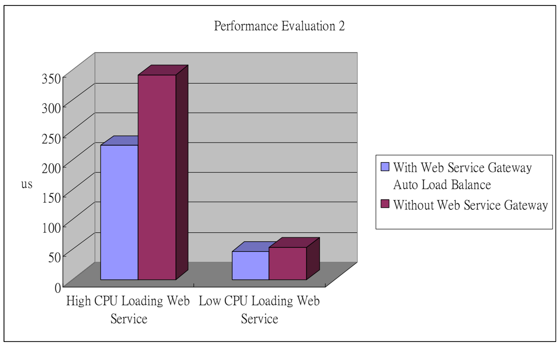

Figure 4. The average response time with/without web service gateway auto load balance.

V. CONCLUSION

As shown in this paper, the UPnP mechanism we proposed really makes web service gateway architecture more flexible and intelligent in deployment and management. Furthermore, the idea we proposed can be applied on the SOA-based system. With this mechanism, SOA components will automatically find available bindings in the intranet without complex configurations and even combine other SOA components to a new integrated service spontaneously. Therefore, the information exchanged by SOA components makes SOA-based system deployment and management more efficient.

REFERENCE

[1] Chandra Venkatapathy and Simon Holdsworth, “An introduction to Web Services Gateway,” http://www.ibm.com/developerworks/webservices/library/ws-gateway/, May 2002.

[2] Universal Plug and Play Forum, http://www.upnp.org/.

[3] OASIS Reference Architecture for Service Oriented Architecture 1.0, Public Review Draft 1, Apr. 23, 2008.

[4] Michael Ellis, “Employ the IBM WebSphere Web Services Gateway,” http://www.ibm.com/developerworks/webservices/library/ws-routing/, Sep 2004.