IF : 4.547 | IC Value 80.26 Volume-6, Issue-4, April - 2017 • ISSN No 2277 - 8160

Original Research Paper Engineering

Discovering and Invocating Web Services from Mobile Devices Dynamically Using a Weighted Load Balanced Proxy Based Server.

Eben Praisy Devanesam K M.Tech, Department of Computer Science and Engineering, SRGI Jhansi, India.

C.P.Singh Assistant Professor, CS/IT Department, SRGI Jhansi, India.

ABSTRACT Web services are built on top of open standard based web applications that interact with other web applications for the purpose of exchanging data. It is a piece of software that makes itself available over the internet. The most important requirement for dynamically accessing web services from mobile devices is to improve the processing speed and to save battery power. Our design is based on SOAP. This architecture makes use of a weighted load balancing proxy server to discover the needed web services from mobile devices and build their proxies. This algorithm stands out to be efficient in managing the load without consuming the low capability servers the most and efficiently utilizing the available server resource at any given instant of time.

KEYWORDS : Web Services, Dynamic discovery and invocation, Proxy servers, weighted load balancing algorithm, SOAP.

1.INTRODUCTION:

Mobile computing and the popularity of mobile devices has motivated interest in accessing web services from mobile devices efficiently and dynamically. A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service.. All the standard web services work using the following components- SOAP (Simple Object Access Protocol), UDDI (Universal Description, Discovery and Integration) and WSDL (Web Services Description Language). A web service takes the help of XML to tag the data, SOAP to transfer a message and WSDL to describe the availability of service.

There are three major components within the web service architecture which is the Service Provider, Service Requestor and the Service Registry.. The service provider implements the service and makes it available on the Internet. The next is the Service Requestor / Client which denotes any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request. The third component is the Service Registry which is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones.Service discovery is handled via Universal Description, Discovery, and Integration (UDDI).Service description is handled via the Web Service Description Language (WSDL).

However the limitations of mobile devices and the mobile environment pose great challenges for consuming web services. Thus mobile devices should aim to minimize their interactions with the network and reduce their resource consuming processing whenever possible. Also while making use of UDDI registries for service discovery, the mobile devices require multiple round trips in networks and frequent disconnections in the wireless network that limits the service discovery process which is expensive. This paper proposes an architecture which makes use of a weighted load balancing proxy server to discover the needed web services from mobile devices and build their proxies. After getting the proxy, a mobile device can invoke a particular method of the web service and get the desired results. Weight-based load balancing improves on the round-robin algorithm by taking into account a pre-assigned weight for each server. The request will be assigned to the server as per its weight. This algorithm stands out to be efficient in managing the load without swarming the low capability servers the most and efficiently utilizing the available server resource at any instant of time.

2 RELATED WORK:

To begin with Robert Steele et al. [1] presents a novel architecture for discovery and invocation of mobile Web services through automatically generated abstract multimodal user interface for these services.In this proposed architecture, the discovered Web services are invoked dynamically with a transparent mechanism. Moreover, the proposed architecture is a proxy based distributed architecture that is best implemented using a component-based implementation approach that provides its core functionality as Web services. It makes use of XSLT to generate multimodal XForm based interface. They also make use of an extended version of XForms through attaching modality specific handlers to the XML events. They have made use of VoiceXml Forms to implement voice modality handlers and attach them to certain XML events. However better resource utilization is an issue with this system.

M.Josephine Spinoza Jane et al. [2] proposes a system which shows that almost all the client processing has been transferred to the load balanced server. This server has a process which is used to download a list of web service descriptions and associated resource identifiers from UDDI registries as a response for the user request. At first the server performs string matching process in which a list of available web services based on string matching scheme is created. This list is then matched with the user's supplied search string and the most appropriate web service method is identified. Next the WSDL files are downloaded. Finally the proxy class generator generates the client-end proxy class which is transferred to the mobile device. They make use of a load balanced server for better resource utilization. It helps in the process of distributing service requests across a group of servers.

For dynamic invocation of web services methods, C.Weyer et al [3], have developed a few platform-specific technologies. The DynWsLib library is a .NET framework based technology that works by generating the client-side proxy class at runtime. However this library does not provide an effective mechanism to handle specific types during the proxy generation process.

The Proxy Factory project [4] which builds on DynWsLib, tries to address the issue of handling specific types during proxy generation process. But more specifically it tries to take advantage of the unification of communications technologies provided by .NET. However ProxyFactory cannot run on the Compact Framework (CF) (lightweight version of .NET that targets "smart" devices) because the class that is needed for serialization is not available in this framework.

3. PROPOSED SYSTEM:

We design this architecture based on SOAP. This architecture (Fig 1.1) incorporates a weighted load balanced proxy server which acts as a Man in the Middle server. This server will be used by mobile

devices to discover the needed web services and build their respective proxies. Once we get a proxy, then the particular mobile device can invoke a particular method of the web service and get the desired results. This proxy server offers a web service which can be invoked by the mobile device and is then passed on to a search string. The server compares the submitted string to short description of Internet Web Services which are cached and then generates a list of services that match the user's string the best. This proxy server downloads the WSDL files which is used to identify the most appropriate web service. Then a source code is generated from WSDL file of the selected service and is compiled to generate client side proxy class at runtime. Finally the mobile application generates one dynamic GUI to supply values for web methods by the user and another GUI to display the results. Additionally it performs an efficient load balancing using weighted load balancing algorithm such that at any given time, the available servers can be efficiently used. This program can be written in java using Android framework. It runs on Apache Tomcat server accessing a HSQL database on an Eclipse IDE. The results can also be displayed using an android emulator.

3.1 PUBLISHING SERVICE DESCRIPTION

The functional purpose of a UDDI Registry is to represent data and metadata about Web services. Standard based mechanism to catalog, classify and manage Web services efficiently is offered by a UDDI registry, so that they can be discovered and invoked by other applications.

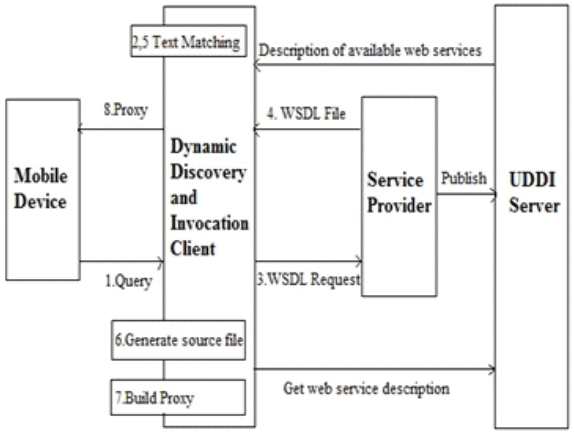

Fig 1.1.Architectural Diagram of the proposed system.

The interaction framework of UDDI registry (Fig 1.3) involves business entities, business services, binding templates and tModel. Businesses and Services and their references to detailed documentation are stored in the registry. After publishing your service into the UDDI registry, a client looks up the service in the registry and service binding information is received. The client uses this binding information for invocation of services. We make use of the jUDDI registry.

3.2 DISCOVERING WEB SERVICES

In order to invoke a web service, a search query of the client's desire is sent to the proxy server by using User Interface. The proxy server searches the cache for the already downloaded WSDL files. If it does not find it there, the server queries the UDDI registry with the service name. It throws an exception if the service name is unavailable. If it's available then it gets the binding details for the service. The next step is to obtain the WSDL files of the selected services. Each of the needed WSDL files is now parsed in order to get the web methods of the web service and their essential documentations. For each service, the parser returns an array of operations, which is a class consisting of two strings: the web method's name and its documentation. It then automatically generates a xml response to the client application. This xml response has the description of the web service and the number of input parameters and service providers name. The user selects the service of his interest and finally the server processes this request for invocation of the service.

3.3 PROXY CLASS GENERATION AND INVOCATION

After identifying the web service, the proxy is built from its WSDL file. This task involves generating a class (source code), compiling it and then sending it to the mobile device where it is used for direct method invocation. All the involved code runs on the MIM server. The URL of the discovered web service by the MIM server is used to get the service's details. With this information, the mobile application generates a dynamic GUI. Finally after getting the user's input through the generated GUI, the method is invoked at runtime. We make use of Apache Axis 2 which is an implementation of SOAP in order to implement the Java web services.

3.4 WEIGHTED LOAD BALANCER

There is sometimes a tendency for the low performance resources to be overloaded with requests leaving the high performance resources idle. So we assign a weight to each server while configuring the weighted load balancing server thus the load gets distributed across servers. We make use of Apache camel for routing mechanisms.

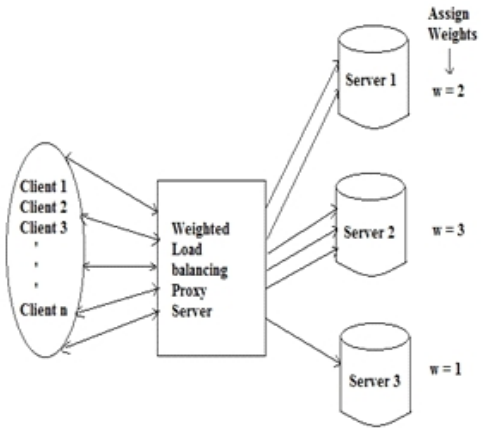

Fig 1.2 Mechanism of weighted load balancing.

Fig 1.3 Screenshot

4. CONCLUSION

This paper provides a mechanism to dynamically discover and invoke web services for mobile devices additionally performing load balancing on the servers. Since load balancing of web services which are discovered and invoked dynamically using a proxy server is essential, we make use of an efficient algorithm such as weighted load balancing algorithm on the proxy server which acts as a Man in the Middle server to efficiently manage the load by utilizing the available server resource at any instant of time. In this manner we aim to minimize the interactions of the mobile with the network and reduce consuming of resources whenever possible.

REFERENCES

[1] Mobile Web Services Discovery and Invocation Through Auto-Generation of Abstract Multimodal Interface Robert Steele rsteele@it.uts.edu.au Khaled Khankan kkhankan@it.uts.edu.au Tharam Dillon tharam@it.uts.edu.au University of Technology, Sydney,PO Box 123 Broadway, NSW 2007 Australia

[2] Dynamically Discovering and Accessing Web services in Mobile devices using a load Balanced Proxy based Server.M.Josephine Spinoza Jane & Prof M.Somasundaram.

Anna university, Chennai. IEEE publication 2013
[3] C.Weyer, "DynWsLib Tutorial," http://www.thinktecture.com/resources/software/DynWslib/default.html, 2011
[4] CodePlex, ProxyFactory Home Page, www.codeplex.co / ProxyFactory, 2011
[5] Configurable Composition and Adaptive Provisioning of Web Services." Quan Z. Sheng, Member, IEEE, Boualem Benatallah, Member, IEEE, Zakaria Maamar, and Anne H.H. Ngu. 2009
[6] Hassan Artail and Kassem Fawaz and Ali Ghandour (2012), " A Proxy Based Architecture for Dynamic Discovery and Invocation of Web Services from Mobile Devices", IEEE Transaction on Service Computing, Vol 5, No.1, pp. 99 – 115.