*International Journal of*
*Emerging Research in Management &Technology*
*ISSN: 2278-9359 (Volume-2, Issue-11)*

**Research Article** | **November 2013**

# Proxy Server Based Data and Service Accessing In Mobile Devices

**R.Sudha,**
***Associate Professor***
***sudhanita@gmail.com***

**R.Bhaskaran,**
***Associate Professor***
***bhaskaranmdu@gmail.com***

*Dept. of Information Technology,*
*PSNA College of Engineering & Technology,Dindigul-624622*
*Tamilnadu, India.*

***Abstract***

***Local Server Based architectures can provide a more universal integration with Web servers since the common standard protocol, HTTP, is used instead of vendor-specific application programming interfaces. This paper describes an effective architecture which allows mobile users to acquire needed data through dynamic invocation of web service methods by merely providing search phrases and method argument values into a dynamically generated GUI. The architecture overcomes technical challenges that involve consuming discovered services dynamically by introducing a man-in-the middle (MIM) server that provides a web service whose responsibility is to discover needed services and build the client-side proxies at runtime. The architecture comprises the MIM server energy-consuming tasks that would run on the mobile device otherwise. Such tasks involve communication with remote system in the local area network, servers over the Internet, Parsing of XML files, and compilation of source code at the same time. Our design is based on the framework defined for realizing dynamic invocation of service, and stresses practical aspects as perceived by the user in terms of interface simplicity and effectiveness in returning desired results.***

***Keywords— Web service discovery, dynamic invocation, mobile devices, mobile computing***

## 1.INTRODUCTION

Mobile devices have some limitation for consuming web service like time delay, frequent unavailability etc. Sometimes wirless network may cause failures in the service discovery process such as providing services to the users, and it have some obstruct to completion of the user request. There also a problem of lower processing power, limited bandwidth, less memory, and finite battery power when compared to desktops. The process needs more resources to complete the process. So users cannot use mobile frequently. All the above suggest that architectures which target mobile devices should aim to minimize their interactions with the network and reduce their resource consuming processing whenever possible. These mentioned issues and challenges led to the configuration of the architecture which introduces a server in the middle whose role is to interface the mobile devices to the web services in a way the mobile devices are adapted to the capabilities for accessing the services provided by the web server. Here the most of the workload involving in the dynamic discovery of web services is undertaken by the server, thus mobile devices are relieved from the time and energy consuming tasks such as communicating with remote systems, internet servers and parsing of Web Service Description Language (WSDL) files.

For a mobile device to access the web service there are two types of solutions either the mobile device will directly interact with the service when the WSDL file is parsed by the mobile devices and no commercial compilers have been developed to date for such devices. This lead to the introduction of the server between the mobile device and the internet based web service. The server will do the parsing and build proxy for the service requested by the user. This process of building proxy by the local server and send it to the mobile device looks like a local process and it is available for the mobile whenever it is needed. Battery power is the limited constraint of the mobile devices. It becomes critical whenever the mobile devices are used for communicating with the internet, remote systems for searching the user request in the systems and discovering user requested services and invocation of their methods. I n summary the architecture allows mobile users to acquire needed data through dynamic invocation of web service methods by merely providing search phrases and method argument values into a dynamically generated GUI. The design is based on the framework defined in [3] the architecture realizes service invocation dynamically and the practical implementation of the interface as perceived by the user and returning the desired result.

There are two types of general techniques for accessing web services using SOAP and the REST approach. SOAP message could be sent to a websit that has web services enabled with the parameters neede for a search.The site would then return an XML formatted document with the resulting data.With the data being returned in a standardized machine parsable format,it can then be integrated directly in to a third party website or application.The process of web service invocation using SOAP messages involves that the client side proxy application encloses the user's request a form of method call by using the SOAP messages.This message is sent to the web server in which the requested service is present .This service will extract the call and executes the call.This execution of calls will produce the results and they are enclosed in the SOAP messages and sent to the requested client side proxy.This client side proxy will extract the message and send it to the requested client.Here the proxy is generated bygenerating the source file from the web service description language files which is present in the UDDI registries.This WSDL file consists of information that describes the web service,how to access it,the operations it performs,the types of parameters to be passed to each of the supported methods and the types of returned results.After the source file is generated it is compiled in to a proxy class that is

finally registered with the client application.This proxy allows the client application to make method calls as it were calling a local function. Web service access by using the REST stands for Representational State Transfer [2] states an architectural style provides an URI that represents the resource in the form of XML document which has links to other documents. This document contains the detailed information by which the client transfers from one state to the another by choosing from the alternative URLs represented in the XML document. To send data to the server, the service also provides a URI that allows the user to create an instance document which conforms to a schema publicized in an XML document, thus allowing the client to submit the input document as the payload of an HTTP POST.

## 2. RELATED WORK

Several approaches have been developed for enabling dynamic invocation of web services in mobile devices and analysis of performance of the MIM server. These are either conceptual solutions or the proposed architectures. The paper is based on the second type but differs from the proposed schemes in that it is complete, fully dynamic, employs generic technologies, and is not restricted to a particular platform. However the survey that follows depicts how the proposed solutions in the literature either only allow non dynamic access to deployed web services from mobile devices, or provide dynamic access to services, but not from mobile devices. Starting with the first class of solutions mentioned above, an application for Peer-to-Peer (P2P) web services is suggested in [6] for use in ad hoc networks. In this article, a heterogenous environment is being composed of mobile nodes with computing and communication capabilities. Each peer node equally acts as both, server and client. These nodes provide their services to other nodes in the distributed environment, and they are able to use remote services. Here the server based implementation is focused which enables the mobile devices to provide and publish their services.In this system P2P web services are implemented in java enabled mobile devices that was used to analyze the memory usage and response time of the SOAP server.In[2] a system was presented to study the resource consumption and performance of providing web services on mobile phones.The analysis of a prototype which was developed using personal java on a Sony Ericson P800 phone shows that the implementation is able to handle up to eight concurrent users with reasonable time delays.In[10] two architectures were presented for high and low end mobile devices using Java 2 Platform Micro Edition(J2ME),Web service API(WSA) and short messaging service(SMS),which define a specific architecture for each case.

WSA integrates basic support for web services invocation and XML parsing in to the runtime environment of the mobile device.A network application that resides on a WSA enabled device must include a stub,which is generated by a tool on a development workstation and then deployed to the device.The application can then employ the stub and the runtime to parse XML SOAP messages using JAXP and consumes the service using JAXRPC.To access web services using SMS,a gateway that translates short messages to HTTP requests was used to connect the SMS center to an application server,which in turn translates HTTP request or response to or from SOAP messages.Every time a web service needs to be accessed programmed and configuration is required in order to create a stub or to set up communication using SMS.

In another type of approaches servers are used in the middle between the mobile devices and web servers.In[9] it was argued that the normal web service architecture that employs requestor, a broker and service provider alone not allow for web service invocation dynamically that is suitable for mobile devices. It is suggested that a service gateway must be included between the requestor and the rest of the architecture. In such a solution it was recognized that additional code must be included on both the mobile and service side.In this paper results were analyzed from simulation shows that service access using gateway in the architecture is faster compared to that of service access using KSOAP protocol.

In [7] Halteran and Pawar discusses the requirements for nomadic mobile service provisioning and proposes the mobile service platform (MSP) as a supporting infrastructure and middleware which extends the service oriented architecture paradigm to the mobile device. The MSP design is based on the Jini surrogate architecture specification [12] which enables devices that can not directly participate in a Jini network to join a Jini network with the aid of a third party. MSP consists of an HTTP Interconnect protocol to meet the specifications of Jini surrogate architecture and provides a custom set of APIs to develop and deploy a nomadic mobile service.

Invocation of web methods dynamically involves development of few platforms technologies.The DynWsLib library[13] stands for a .NET library which enables to dynamically invoke XML web services at runtime.This generates the client side proxy and it does not need WSDL file during the proxy design time.However, according to one developers forum,this library worked consistently with services running on the local server,but not always trying to reference remote services.Moreover this library which is no longer supported does not provide an effective mechanism to handle specific types during the proxy generation process.

In [11] the work proposed was to design of automatic generation of the multimodal abstract user interface. This paper presents a novel architecture for discovery and invocation of mobile Web services through automatically generated abstract multimodal user interface for these services. A prototype has been developed to auto-generate user interface based on XForms and VoiceXml from a WDSL file. In this proposed architecture, the discovered Web services are invoked dynamically with a transparent mechanism. Moreover, the proposed architecture is a component-based architecture that provides its core functionality as Web services.

Finally, in the work [8] where Man In the Middle server is used in between the requestor and the service provider which does the maximum work of generation of proxy class by the process of text matching with the cached web service descriptions that are sent periodically from the UDDI registries by periodical requests from the MIM server. If the requested service is not present in the cached files then the server will send request for URLs to the web server and the web server will send the WSDL files. With these files the matching file is selected and the source code is generated.Then the source code is compiled by using libraries and build the client side proxy and then it is shipped to the mobile device.This client side proxy is responsible for generating the GUI dynamically which provides the service results requested by user in the mobile device.

## 3. PROPOSED SYSTEM DESCRIPTION

The architecture describes the deployment of a man in the middle server between the requestor and the rest of the architecture. This architecture is based on the concept of SOAP.

*3.1 Mobile Process*

Using internet in mobile have the disadvantages of lower processing power, limited bandwidth, less memory, and finite battery power when compared to desktops. In this process the mobile is connected with the MIM server. This process will reduce the disadvantages of mobile accessing the internet. The first step is to send the query phrase to the man in the middle server. This query phrase is used in the MIM server for the process of searching contents. After completion of work in MIM server it returns the proxy to the mobile. With such information, the mobile application generates a dynamic GUI for the user to supply values for the web method parameters, and then another GUI to display the results. These are the process done in the mobile device.

*3.2 Man In The Middle Server Process*

The MIM Server offers a web service which exposes a web method that the mobile device invokes and passes to it a search string. Man-in the Middle server does three types of processes such as, Text matching, Generating source files, and Build proxy. In text matching process to generate a short list of candidate web services based on matching their cached short descriptions with the user's supplied search string, and second, to identify the most appropriate web service among those short listed based on matching the methods' description found in their downloaded WSDL files with the user's string. The second process is the WSDL File Downloader which takes as input the URIs of the shorted-listed services, and downloads the files if they are not in the cache. Finally, the third main process is the Proxy Builder, which generates a client-side proxy class and sends it to the mobile device through FTP. This same process identifies the number and types of input and output parameters of the web method whose name was passed by the Text Matching process, and returns this information to the latter so it can be passed on to the device.

*3.3 Web Server Process*

Web server gets the WSDL URLs from the MIM server and accesses the information and sends the WSDL files to the MIM server. So this type of process is to speed up processing on the server and improve the scalability of the architecture, the described three processes are multithreaded and communicate through socket interfaces by receiving and sending messages, and in some case sending files using FTP. Moreover, we also improve the average response time of the server by caching the WSDL files while assigning them time-to-live (TTL) values and keeping track of their access rate. This keeps the list of cached WSDL files manageable and gives preference to those that are most used.

*3.4 Web Service Process*

Web service is the process used to connect the man in the middle server to multiple systems using local area network connection. So man in-the-middle server first sends the IP request to another system. This request is passed through the web service. After the connection is accepted, systems of man in-the-middle server and remote system start to share the files. So the mobile device gets the power to search the files from the remote system. For downloading the files from the remote system, first mobile device send the download request to the remote system, for this purpose short message protocol is used to send the request. After accepting the request, the file downloaded into the mobile device. This process is done by the connectivity of web services between the multiple systems. In this process more number of system are connected at a time. So process provides the more efficient.

## 4. IMPLEMENTATION

In the architecture the mobile device web service client application was implemented using the C# programming language in windows.NET Compact Framework. The client applicationn was installed in virtual pocket PC 2007 running the Windows Mobile operating system and set up to communicate with a wireless access point using Wireless LAN. The local server called MIM server is implemented in laptop with intel core processor with 1 GB memory.

*4.1 Discovering Web Methods*

In internet several UDDI registries have been available for desired services requested in order to provide API for the services. But in order to do the process of evaluation the actual process was restricted to create XMethods registry which has the list of XMethods which offers the set of services requested by the user for all the services listed. In the local MIM server the back ground process will calls this method and gets the service descriptions. The service descriptions are cached by storing in the MS access database table. With these web service descriptions the local server will generate the

proxy class which will be used by the pocket PC in order to interface with the web service discovered. In this paper we have cached only certain type of services such as information regarding the airline and health services. The service descriptions regarding these two fields are retrieved from the internet and they are stored in the database table. Whenever the user requests regarding airline or health care the MIM server will retrieve information according to the request phrase provided by the user. If the information is not present the user is asked to specify the search phrase little more accurately.

After the searching process step the next process is to obtain the web service description language files of the requested service. If the file is not present in the cache a request for the service is searched in the intranet by means of sending the IP request to another system. This process is done by using web service. After the connection was accepted the Man In the Middle server will starts to share files. After that the WSDL files are parsed in order to get the methods of the web service and their corresponding documentation. Initially the XML file is searched for the operation name (name of the web method) and the documentation. This web method is used for further process of proxy generation by the MIM server.

*4.2 Proxy Class Generation and Compilation*

After web services were discovered the dll file means the proxy is generated from the WSDL file. This process involves generating source code and then it is compiled. This is sent to the mobile device application for direct invocation of the discovered methods. The MIM server is used to get the service's details by using the URL of the discovered web service. More specifically, the service descriptions and schemas are retrieved by the Server and saved in a list. The server performs a set of steps that generates the instance of classes by which the source code for the user requested service's WSDL file is programmatically generated. After the source code is generated the assembly file is created through a set of additional steps that uses compiler parameters and assemblies that are specific to smart devices.

In our implementation the types in assembly are found and the methods in it are dynamically invoked. Here the types of web method's input and output parameters are returned to the mobile application on the pocket PC. With this information the mobile application generates a dynamic GUI. Here we have drawn text box for each input parameter with its name denoted in the label. After getting the input parameter from the userthe method is invoked at runtime. Further another dynamic GUI is generated which consist if the tabular column displayed with the contents of the output parameters. If the information retrieved has more number of results scroll bar is present in order to navigate to the whole set of results.

The developed mobile application is called the proxy server once the user starts the application the initial form is displayed which consist of the two buttons for retrieving information abou the health care report all over the world and airline information about all the flights in the airports all over the world. Whenever a health button is pressed to get the desired service a GUI is generated to provide information related based on the country and category wise.Whenever the country option is selected the GUI which displays a list of countries and the years. This enables the user to select the country and year wise search it will take the selected values as input parameters and display the result in another separate generated GUI. In the figure displayed below the user want to search related to the category and indicator wise selected by the user. Another GUI is displayed which consist of country name and the year and related ratio information in the generated GUI.


Fig. 1.1. Application screenshots: Home Page


Fig. 1.2. Application screenshots: Health page


Fig. 1.3. Application screenshots: Input Parameter


Fig. 1.4. Application screenshots: Output parameters

## 5. CONCLUSION

The presented architecture makes it possible for mobile device users to dynamically invoke web service methods that meet their needs. The implemented solution overcomes technical limitations, and also saves device battery power,thus extending its participation in the wireless network. The scalability study can be used to decide on deployment of MIM servers in the network: given the capacity of the server, the number and distribution of MIM servers can be determined, knowing the cumulative expected request rate from users.

## REFERENCES

1) C. Aggarwal, J. Wolf, and P. Yu, "Caching on the World Wide Web," IEEE Trans. Knowledge and Data Eng., vol. 11, no. 1, pp. 94- 107, Jan./Feb. 1999.
2) M.Chatti, S.Srirama, D.Kensche, "Mobile Web Services for Collaborative Learning," Proc. IEEE Int'l Worksh op Wireless Mobile and Ubiquitous Technology in Education, pp.129- 133,Nov. 2006.
3) R.Costello,"Building Web services the REST way," http://www.xfront.com/REST-Webservices.html 2011.
4) I.Duda M.Aleksy and T.Butter, "Architectures of Mobile Device Integration into Service Oriented Architecutures," Proc. Int'l conf. Mobile Business 2005.
5) R.Fielding and R.Taylor, "Principle Design of the Modern Web Architecture",ACM Trans., on Internet Technology, Vol 2, pp.115-125,2002.
6) G.Gehlen and L.Pham, "Mobile Web services for Peer-to-Peer Applications", Proc. Of IEEE Int'l Conf. On consumer Commn., and Networking, pp. 427-433, Jan.2005.
7) A.Halterern and P.Pawar, "Mobile service platform: A Middleware for Nomadic Mobile Service Provisioning", Proc.of IEEE Int'l Conf. Wireless and Mobile Computing and Commn.,(WIMOB), 2006.
8) Hassan Artail, Kassem Fawaz and Ali Ghandour, " A Proxy-Based Architecture for Dynamic Discovery and Invocation of Web Services from Mobile Devices" IEEE Transactions On Services computing, Vol. 5, No. 1, January-March 2012.
9) L. Li, M. Li, and X. Cu "The Study on Mobile Phone Oriented Application Integration Technology of Web Services, " Integration Technology of Web Services, " Proc. Of Int'l Conf. On Grid and Co-operative Computing (GCC), Apr. 2004.
10) O.Rendon, F.Pabaon, M.Vargas and J.Guaca, "Architectures for Web Services Access from Mobile Devices", Proc. Of Proc. Third Latin Am. Web Congress (LA-WEB '05), pp. 93-97, 2006.
11) R.Steele, K.Khankan and T.Dillon, "Mobile Web Services Discovery and Invocation through Auto Generation of Abstract Multimodal Interface", Proc. Of Int'l Conf. Information Technology : Coding and Computing (ITCC '05), vol. 2, pp. 35-41, 2005.

12) Sun Microsystems, "JINI Technology Surrogate Architecture Specification, "http://surrogate.JINI. org/sa.pdf, Oct. 2003.
13) C.Weyer, "Dyn Ws Lib Tutorial, :www.thinktecture.com/resources/software DynWsLib/default.html, 2011.