A Novel Web Service Directory Framework for Mobile Environments

Rohit Verma, Abhishek Srivastava
Department of Computer Science and Engineering,
Indian Institute of Technology Indore, India
Email: {phd12110101, asrivastava}@iiti.ac.in

***Abstract*—Mobile devices are evolving as a new computing platform and a common means to provide access and process digital information. In the move to achieve ubiquitous computing, the role of mobile devices and web services cannot be understated. Mobile devices are predominantly in use for accessing web services. The same mobile devices are now becoming a feasible option for small and medium size enterprises to host and provide personalized web services to clients. The motivation being the minimal infrastructure requirements and configuration costs. Technology today enables the provision of web services over hand-held mobile devices, realizing a web based service-oriented architecture in a mobile environment. For this, an efficient service discovery mechanism is required. It is difficult to adapt traditional approaches of managing web service directory for mobile environments; this is mainly due to the dynamic arrivals/departures of mobile devices in network zones. In this paper, we propose a model for web services in mobile environments to maintain a service directory using the XMPP (eXtensible Messaging and Presence Protocol). The proposed model enables mobile devices to manage web service directory without requiring high-end computers and high management cost. This paper presents proposed architecture, design concept, system components and workflow of the framework. Moreover, a comparative study of the proposed approach and the traditional UDDI (Universal Description, Discovery, and Integration) registry is presented.**

Keywords: Mobile Web Services, Service Discovery, Service Directory.

I. INTRODUCTION

Advancement in technology has made mobile devices an important computing platform from merely being a communication device. Mobile devices such as smart phones, PDAs, tablets have become a new way of information management and enterprise productivity. Mobile devices are easily available and are dispersed amongst crowd of people. Such mobile devices have the potential to form a high computational resource pool, if harnessed appropriately. One way of efficiently utilizing the capabilities of mobile devices is to use them to host web services. To the best of our knowledge, mobile devices are predominantly used for accessing web services but little work has been done to enable hosting them [1] [2] [3]. There are several challenges in hosting web service over mobile devices. These include and not limited to battery and network constraints, limited computational power of mobile devices, dynamic service registry management, dependency on legacy web service architectures involving SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language), UDDI (Universal Description, Discovery, and Integration) [4].

With an increasing number of dynamic and personalized web services hosted over mobile devices, discovery of such services becomes an issue. The services on offer in such environment tend to become 'volatile', owing due to the fact that hundreds of mobile devices join and leave the system at any instance of time. In such a scenario, centralized management of web services through a service registry (e.g. UDDI registry [5]) as is done conventionally is neither feasible nor practical. Chief among the reasons for this is that a failure of a mobile device (due to network outage, battery issues) prior to notifying the centralized service registry increases the probability of outdated information at the registry. Moreover multiple mobile devices hosting web services arrive and depart from the network. This would result in the need for frequent updations of the central service registry.

The main objective of this paper is to facilitate service oriented architecture [6] over mobile devices without the involvement of high end server. Service oriented architecture over mobile devices has several advantages: reduced cost, cost sharing, resource aggregation, improved scalability, increased autonomy, private and ad-hoc services, dynamic information sharing. In the move to achieve service oriented architecture, mobile devices should act as a service provider, service consumer and service registry as well. In proposed work, we focus on deploying service registry over mobile devices. The big issue however is that traditional approaches of service registries cannot be directly deployed over mobile devices. The reasons: mobile devices are computationally less powerful, network issues, battery life, the 'mobility' of mobile devices. These limitations can make service registry hosted over mobile devices prone to unexpected service outage and make them a single point of failure.

In this paper, we therefore focus primarily on a technique to facilitate service registry management in mobile environments. We propose the use of XMPP (eXtensible Messaging and Presence Protocol) for maintaining the service directory for service discovery. The XMPP protocol is widely used for messaging and chatting applications by virtue of features such as contact management, contact search, presence leverage through the network, server-less working for distributed application, inherent XML (Extensible Markup Language) support. Our approach is to manage service registry locally and providing service directory in the form of a contact list as in messaging application along with its availability status using XMPP.

The advantage of the proposed architecture is that it can be used in a scenario where there is little or no preexisting infrastructure. Examples of such scenarios are: war-front areas,

post-disaster relief management, dynamic environments (highway, ad-hoc networks, meeting areas), collaborative working sites. Furthermore, the proposed architecture would provide all service registry related information and operations using mobile devices such as service discovery, service updates, service availability. Mobile devices today are omnipresent in the world and can provide services to other computational devices, if not independently, in a collaborative manner. We propose to use heterogeneous and loosely coupled mobile devices in a collaborative manner to manage service directory along with hosting web services. Our approach suggests that each mobile devices should act as a web service provider and service registry as well as may act as service consumer (requester). The proposed architecture has potential to enable new/small businesses to provide services over mobile devices to customers with minimal dependency on infrastructure. We propose to use existing protocols and technologies with minor modifications for providing web services registry service over the mobile devices. In this paper, we compare and contrast proposed approach and traditional UDDI approach for managing service registry from the perspective of mobile devices

The rest of the paper is organized as follows. Section II is the related work section. Section III presents the proposed architecture and inline comparison with UDDI service registry model, followed by IV briefly talks about the various application scenarios of the proposed system. Finally Section V concludes the paper with a brief discussion on future work.

II. RELATED WORK

Mobile computing has evolved and become popular in the last few years. It has today become a means for anytime-anywhere computations. Mobile devices are commonly used for simple computations, accessing the Internet, checking email, etc. Accessing web services over mobile devices is also no exception. Service providers target mobile device users and provide them a diverse range of web services.

With the parallel growth in mobile technology and networking technologies, web services for mobile devices is becoming a new paradigm. The idea behind mobile web services is to enable mobile devices to host, access, provide and integrate web services and information. Substantial work has been done on enabling mobile devices to access web-services [7] [8] [2].

On the other hand, however, hosting web-services over mobile devices is a relatively new area of research. The potential of mobile web services was first discussed by Berger et al. [9]. Work has been done on hosting web services over mobile devices [10] [1] [2] [11] based on SOAP as well as REST [3] [12]. AlShahwan et al. [13] provides a comparison between the SOAP and REST framework from the point of view of utilizing em in mobile environments.

Most of the work done on mobile web services utilizes a standard directory system with UDDI for web service discovery. A few group have explored fresh ideas for web service discovery in mobile environments.Recent work by Elgazzar et al. [14] discuss about the concept of personalized web service discovery based on the use of context information and user preferences. However they made use of existing WSDL documents for context-aware web service discovery. Al-Masri et al. [15] propose a device aware service discovery mechanism, MobiEureka, for mobile environments. MobiEureka make use of input keywords and recommends relevent mobile services to the client devices. Sapkota et al. [16] discusses use of shared memory for web service discovery. Their work provide a shared memory for web services to publish their service descriptions and make it available for service discovery. This approach could be adopted for mobile devices. The goal of our paper is to propose a novel approach for managing the service directory for web services hosted over mobile devices. This would enable mobile devices to manage service registries on their own, drastically reducing the cost and dependency on infrastructure. This could potentially also help in providing services in dynamic networks such as vehicular networks, mobile ad-hoc networks (MANETs).

XMPP stands for Extensible Messaging and Presence Protocol [17] [18]. It is an open Extensible Markup Language (XML) based protocol for near-real-time messaging, presence, and request-response services. XMPP is the formalization of the protocol developed by the Jabber open-source community in 1999. As per RFC 3920 [17], XMPP is the protocol for streaming XML elements in order to exchange structured information between two network endpoints in real time. XMPP is used predominantly in instant messaging (IM) applications. Various extension for XMPP have been proposed over time such as extension for serverless mode [19], service discovery mode [20], ad-hoc command support (ad-hoc session support) [21], SOAP support for web services [22]. Bernstein [23] proposed to use XMPP for intercloud topology, security. authentication and service invocations, this work is closely resembles to ours. However, our work focuses on the directory management in mobile based service oriented architecture. We focus on managing service directory in distributed manner on resource constrained mobile devices.

We propose the use of XMPP for providing the web service directory over mobile devices. XMPP is widely used for messaging applications on mobile devices and has proven to be efficient. Features of XMPP such as *presence*, *contact management* (that indicate the communication status of partners) are proposed for maintaining updated service directory information in a distributed manner. XMPP would be used to provide web services on top of existing protocols. To the best of our knowledge, this is the first instance where an architecture is being proposed for web services directory over mobile devices using XMPP.

III. ARCHITECTURE

Mobile device hosted web services mainly exist in peer-to-peer, personal or crowdsourced environment where the hosts are personal mobile devices belonging to the 'crowd'. The focus of this paper is on the problem of dynamic service discovery and publishing in such volatile environments. Protocols and technologies from W3C such as SOAP, UDDI, WSDL are ill suited to such dynamic environments. This work is appropriate for cooperative and personal web services hosted over mobile devices in a local area. We now discuss the design concept and architecture components of the proposed architecture:

A. Design Concept:

The proposed architecture for managing the service directory maintains a list of available mobile web service providers along with their respective availability information. The architecture is based on a publish/subscribe system that exhibits the availability and requests for a web service. XMPP is used, as it is generic, has an open and extensible architecture. Further, it is an instant messaging protocol, that uses XML for supporting request and responses. XMPP is part of the application layer and is built on-top of TCP/IP (Transmission Control Protocol / Internet Protocol), hence is lightweight as compared to SOAP-based protocols such as UDDI. XMPP is therefore suitable for mobile devices that have resource constraint issues.

Moreover UDDI provides a tightly coupled architecture which is less suitable for mobile environment. UDDI architecturel [5] has UDDI data entities (businessEntity, businessService, bindingTemplate, tModel, publisherAssertion, subscription), various UDDI services and API sets, UDDI Nodes for supporting node API set, UDDI Registries. This base architecture makes UDDI difficult to host on mobile and resource constraint devices. Our architecture defined operation of web services in the form of XMPP stanzas or messages while UDDI makes use of WSDL.

AlShahwan et al. [13] and Srirama et al. [24] show that RESTful web services are relatively more suitable for mobile environment. This fits in well with our proposal for offering a directory service as both REST and XMPP utilize TCP and HTTP for communication. The proposed model facilitate service registry deployment over mobile devices. However our approach blends well with any type of web service implementation either SOAP based or REST based. As model would only provide web service directory service, subsequent negotiation between service consumer and service provider (which may be specific to type of service) would be peer-to-peer. This service negotiation is out of the scope of this paper.

XMPP, as per RFC 3920, was designed as a client-server protocol. However, the XMPP extension XEP-0174 supports serverless messaging suitable for WiFi and ad-hoc technologies. This could be useful for private and ad-hoc web services for dynamically publishing and discovering web services. Our architecture consist of mainly two types of services for managing the decentralized service registry:

1) Advertising Service: This service (similar to the service present in the central service registry) advertise hosted service. This service is present at all the mobile devices hosting any web service. For advertising, this service make use of XMPP roster management.
2) Query Responder Service: This service responds to any incoming service discovery request by other mobile devices. This service keeps listening for all query request and responds to them by passing the hosted web service parameters needed for accessing the same.

In the proposed architecture each mobile device may act as a service provider as well as a service consumer. As a service consumer it would find web services from service roster and subsequently invoke the web service after negotiating with provider. As a service provider mobile devices would host services and would publish their hosted services with the XMPP service roster (as shown in Fingure 1).

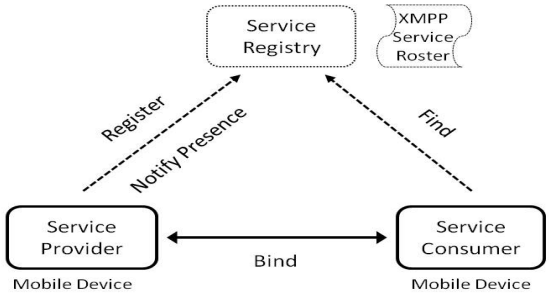

Fig. 1. Service Oriented Architecture Triangle

B. Components:

The main primitive components of the architecture shown in Figure 2:

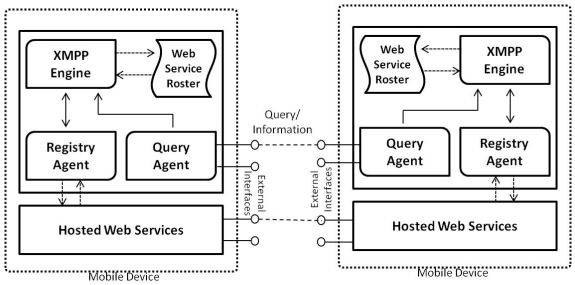

Fig. 2. Service Discovery Architecture

- Registry Agent: This component does the registration on the service registry for the hosted mobile web services. Web services send requests to the registry agent and this agent forwards the request to the XMPP engine. Every time a web service or hosted mobile device becomes unavailable, the registry agent gets notified. Furthermore this agent forwards the status to the XMPP engine for updates in the roster.
- Query Agent: This component accepts and processes the queries from other mobile devices. Following are the main functionalities of this agent:
  - Query Processing: Incoming queries are accepted and processed. Incoming queries seek mobile web services and their availability status.
  - Query Generation: When the native mobile device (XMPP engine) needs to know the status of a remote web service for the web service roster update, the query agent generate the query on behalf of the mobile device.

  The query agent has an external interface. These interfaces are the point of contact for any communication from external mobile device. Query generated are sent to the external mobile device using these interfaces and query are responded back through the same.

Query responder service, as discussed in III-A, run by this component.

- XMPP Engine: This component is the heart of the architecture. This manages web service roster or directory of the web services in the vicinity along with the availability information. Availability information shows whether a web service is available or not at present moment. This is one of the important feature of proposed architecture, as availability of a web service hosted on mobile device is always uncertain, due to dynamic nature of mobile devices. XMPP engine might generate request for remote mobile device (seeking for the status of web service hosted at it) and send it to the query agent. Advertising service as discussed in previous subsection manages web service roster using XMPP engine.

We use the term mobile devices for any smart mobile phone, PDA, tablet which is capable of hosting and providing a web service. Dynamic nature of these mobile devices could be due to unpredictable mobile battery, varying nature of mobile signals, geographical mobility of these devices.

C. System Anatomy:

This subsection presents technical details of the architecture presented in III-B.

Jabber Identifiers:

Each mobile device within the network is addressed by a Jabber Identifier (JID). JID is similar to an email address and is uniquely addressable: `localID@domainID/resourceID`. The `localID` of the mobile device for JID could be chosen using an ad-hoc networking technology such as *multi-cast domain name system* as suggested in RFC 6762, dynamic host auto configuration IP range as suggested in RFC 5735 and 3927. The `domainID` is fixed for the local network. The `resourceID` identifies a web service hosted on the mobile device. The `localID` depicts the type of the web service. This enables mobile devices on the network to look out for only 'interested' web services. It is analogous to the chat groups in chatting applications, where like people with common interests can chat with each other. For example, a mobile device in the "local.mobile" domain hosting a web services for "weather" updates for "XYZcity" could have `weather@local.mobile/XYZcityWS` as its JID. Determination and delivery at the addresses are done as per the XMPP core RFC 3920. JID identifies XMPP engines and query agents of mobile devices distinctly.

Communication:

The proposed model makes use of XMPP [17], [25] stanzas for communication. These stanzas comprise XML based discrete units of structured information sent over XML streams. As shown below, each stream starts with an 'xmlns' namespace declaration `<stream:stream xmlns:stream= http://etherx.jabber.org/streams />`. There are mainly three stanza types used in the architecture: `<message />`, `<presence />`, `<iq />` (as shown in Figure 3).

```
|--------------------|
| <stream>           |
|--------------------|
| <presence>         |
|   <show/>          |
| </presence>        |
|--------------------|
| <message to='foo'> |
|   <body/>          |
| </message>         |
|--------------------|
| <iq to='bar'>      |
|   <query/>         |
| </iq>              |
|--------------------|
| ...                |
|--------------------|
| </stream>          |
|--------------------|
```

Fig. 3. XMPP stanzas

```
<xs:schema
 xmlns:xs='http://www.w3.org/2001
   /XMLSchema'
 targetNamespace='http://etherx.
   jabber.org/streams'
 xmlns='http://etherx.jabber.org/
   streams'>
```

Message stanza is the basic XMPP stanza that works on the "push" mechanism. This stanza pushes information from one entity to another entity. For the purpose of managing and updating the service directory the message stanza type that is used is called `headline` . The `headline` type message expects no reply and is suitable for pushing broadcast content to various devices. A common use of `headline` type message stanza is as an envelope for service description to manage the service directory.

Presence stanza is used to multi-cast the presence information of a mobile device. Each presence stanza includes brief information on the hosted service (status message), along with its availability information. We use 'Online' and 'Offline' as the primitive presence types in the proposed architecture. The online or available status is broadcast when the web service hosted over the mobile device is available. Similarly offline or unavailable is used when the web service is not available. The status of a web service hosted over the mobile device is propagated through the registry agent to the XMPP engine and this reflect on the service roster. XMPP enables obtaining selected updates from selected mobile devices.

IQ stanza is short for Info/Query stanza. It is based on a request-response mechanism and guarantees a response to a query. The data content request in the IQ stanza is of `type get` and is similar to the HTTP `GET` method. The result of the query is of `type result`. Any communication involving query agents primarily makes use of the IQ stanza. These play an important role in getting information from other mobile device that host web service.

D. System Workflow:

The workflow of the proposed architecture is presented here. A comparison with UDDI is also presented. The main functions of the architecture are as follows:

Web Service Registration:

The hosted web service sends a request to the registry agent for registration. The registration agent sends a request to XMPP engine. Request for registration is an IQ stanza as shown below:

```
<!-- Request for registration -->
<iq type='get' id='regreq1'>
 <query xmlns='jabber:iq:register'/>
</iq>

<!-- Response from XMPP engine -->
<iq type='result' id='regreq1'>
 <query xmlns='jabber:iq:register'>
  <servicename/>
  <provider/>
  <owner/>
 </query>
</iq>

<!-- Registration Details -->
<iq type='set' id='regreq2'>
 <query xmlns='jabber:iq:register'>
  <servicename>BinConWS</servicename>
  <provider jid="calculator@testenv.mobile
                /ProgrammerCalc/">
  <owner>Programmer Calc Ltd</owner>
 </query>
</iq>

<!-- Registration Success response -->
<iq type='result' id='regreq2'>
 <query xmlns='jabber:iq:register'/>
</iq>
```

After the completion of the registration process the web service is updated to the service roster. This roster is cooperatively updated with other devices. The registration process in traditional UDDI is done using the *publisher APIs set* exposed by the UDDI, such as save_service, save_business, save_binding, save_t_model. These APIs are used to save detailed information on the web service, which may not be necessary in case of mobile based web services. Moreover this information would tend to become heavy for mobile devices to process or transport.

Figure 4 shows the information stored in a typical UDDI registry [4]. The information is produced by the web service provider to the UDDI registry through publisher APIs (The information is transported as XML tags, we are not showing the XML for the UDDI structure owing to space constraints here. Interested readers may refer: http://goo.gl/cn8VaP). Deploying such a UDDI registry over a mobile device would tend to become heavy owing to limited computational power and network constraints. The UDDI registry would also lag in managing the dynamic nature of mobile devices.

Web Service Discovery:

Web service discovery in the proposed model is meant to suit machines as well as the humans. Humans can search and select a web service from the service roster, which is similar to a buddy list in an Internet messaging application. The web service directory is managed in the form of a roster along with the presence/availability information on the services. This

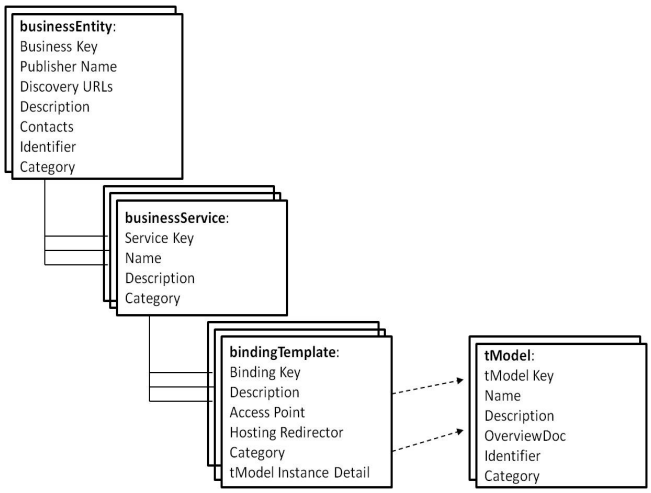

Fig. 4. UDDI Registry Entry

feature of presence information is pertinent to dynamic mobile environments. Multiple mobile web service providers join and leave the network dynamically. We therefore propose a web service roster on each device. This enables the mobile devices to perform service discovery locally. If information on a web service is not available in the local roster, the discovery request can be passed to the other mobile devices via the query agent. This communication takes place as per XEP-022 [26].

We propose two types of service discovery: Direct Discovery and Category based Discovery. These service discoveries are performed using the IQ stanza of XMPP. Direct discovery is performed by parsing the local roster and is based on the required web service from a provider. An example of direct discovery is discovering the web service in local roster providing temperature conversion from Celsius to Fahrenheit. Category based discovery for a web service is performed, when a list of available service of a particular category is required. Following is the example of category based discovery. It shows a directory roster retrieval related to a 'calculator category'.

```
<!-- Request for discovery -->
<iq from="consumer@testenv.mobile/new"
 id="new123"
 to="calculator@testenv.mobile"
 querydesc="Roster List for Calculator"
 type="get">
 <query xmlns="http://jabber.org
      /protocol/disco#items"/>
</iq>

<!-- Discovery Response with three items-->
<iq from="calculator@testenv.mobile"
  to="consumer@testenv.mobile/new"
  id="new123"
  querydesc="Roster for Calculator"
  type="result">
 <query xmlns="http://jabber.org
         /protocol/disco#items">
  <item
   jid="calculator@testenv.mobile/
          ArithmeticCalc"
   WSDesc="Simple Arithmetic
```

```
          Calculator"
    WSProvider="ArithmeticCalc.
          testenv.mobile">
    <group>calculator</group>
   </item>
   <item
    jid="calculator@testenv.mobile/
           ScientificCalc"
    WSDesc="Scientific Calculator WS"
    WSProvider="ScientificCalc.
           testenv.mobile">
    <group>calculator</group>
   </item>
   <item
    jid="calculator@testenv.mobile/
           ProgrammerCalc"
    WSDesc="Programmer Calculator WS"
    WSProvider="ProgrammerCalc.
           testenv.mobile">
    <group>calculator</group>
   </item>
  </query>
</iq>

<!-- Response for direct service discovery
      request from Programmer Calc WS -->
<iq from="calculator@testenv.mobile/
          ProgrammerCalc"
   id='new234'
   to="consumer@testenv.mobile/new"
   type='result'>
   <query xmlns='http://jabber.org
            /protocol/disco#items'>
     <item jid="calculator@testenv.mobile
              /ProgrammerCalc/BinConvWS"
        WSDesc="Binary Converter Service"/>
     <item jid="calculator@testenv.mobile
             /ProgrammerCalc/GCDWS"
        WSDesc="GCD calculation"/>
   </query>
</iq>
```

Once the web service consumer has discovered the appropriate provider, the consumer receives the binding information from the provider itself. Web service discovery in a UDDI registry is done via public inquiry APIs of the UDDI, such as find_service, find_binding, find_business, find_tModel. The service discovery is performed centrally by the UDDI registry server, which requires high computational capability. This is because the consumer requests the UDDI registry server which in turn does the query search centrally and responds to the consumer with the results. The complexity and structured nature of the UDDI data structure as shown in Figure 4 further makes searching difficult. Processing a discovery query over a mobile device using UDDI would therefore be cumbersome. Moreover UDDI registries make use of deployed database systems. (For example Apache jUDDI makes use of Derby database (as packaged component) for managing and providing service directory.) These database systems would drastically slow down the mobile devices. Though traditional UDDIs enable consumers to query the registry and are effective in centralized system, they are ill suited to mobile environments.

Web Service Binding:

Web service binding information is necessary to use a particular web service. It includes the technical information on a web service, such as the end point, required parameter values, return type. In proposed model, message stanza is used for this purpose. Once a web service provider is discovered by the service consumer, the consumer retrieves the binding information from the provider. This approach enables management of the service directory independent of the type of service implementation (SOAP or REST). An example of the binding information from a service provider (calculator@testenv.mobile/ProgrammerCalc/BinConvWS) to the service consumer (consumer@testenv.mobile/user1) is as follows.

```
<message
 from="calculator@testenv.mobile
           /ProgrammerCalc/BinConvWS"
 to="consumer@testenv.mobile/user1"
 id="WSDescp1">
 <body>
  <overview>
    Programmer Calculator WS
  </overview>
  <provider>
    programmercalc.testenv.mobile
  </provider>
  <owner>123abc</owner>
  <binding type="SOAP1.1"
    transport="HTTP"
    Port="80" />
  <method m_id="M1"
    name="IntToBin"
    methoddesc="Convert supplied
             Integer to Binary"
    endpoint="http://programmercalc
                .testenv.mobile/
                BinConvWS/IntToBin"
    parameter="Integer"
    returntype="String">
  <method m_id="M2"
    name="BinToInt"
    methoddesc="Convert supplied
              binary to integer"
    endpoint="http://programmercalc
                .testenv.mobile/
                BinConvWS/BinToInt"
    parameter="String"
    returntype="Integer">
 </body>
</message>
```

The above description is specific to a service and would vary between web services. In the above example minimum information is included, more details could be included. However this information is between the service provider and the service consumer. The message stanza provides technical information for binding. Subsequently WSDL or WADL (Web Application Description Language) document can be exchanged directly between service consumer and service provider. This exchange is out of the scope of this paper.

In the case of a UDDI registry, the service binding infor-

mation is retrieved from UDDI registry server using t_model and WSDL documents. Moreover, several service providers do not register with UDDI registries due to the complexity involved (or due to unavailability of global UDDI registry). Global UDDI registries are also not updated owing to the fact of broken SOA triangle [27]. Hence, as a general practice, consumers/developers performs service/binding information discovery via web search engine with query parameter for *filetype* as *wsdl* (for SOAP based web services) or *wadl* (for REST based web services) for an unknown and new web service. Some query may result in multiple technical and binding documents. Selection of an appropriate web service may need human intelligence and analysis. Furthermore result selection may be dependent on ones choice and analysis. Besides, retrieval of wsdl/wadl document from service provider is also a common practice. Our architecture eases mobile web service consumers from manual search and analyzing multiple non-relevant binding information of various web services. Moreover, discrete web services from new service provider are presented at a common platform. Web service consumer/developer can search services from the service roster directly or in category.

Presence Notification:

This is one of the novel features of the proposed architecture. The service roster manages presence information for a service. This presence information is manged for each Jabber ID. This jabber ID may identify a web service or service provider or web service category. The presence information is similar to the availability information in an Internet messaging application. The presence is shown as 'Online' or 'Offline' as discussed earlier. An example of a presence stanza used to notify presence information is shown below.

```
<presence
 from="calculator@testenv.mobile
           /ProgrammerCalc/"
 to="calculator@testenv.mobile">
 <show>available</show>
 <status>
   "Programmer calculator WS"
 </status>
</presence>
```

Each category of service in the above example "calculator@testenv.mobile" manages the presence and status information of the web services (calculator@testenv.mobile/ProgrammerCalc/). This information reaches other provider's roster through a query agent as discussed in [28].

Roster Sharing:

Roster sharing is required for managing the latest information on services in rosters of various mobile devices. The message stanza is used to perform roster sharing. Message stanzas are exchanged for: Addition, Deletion, Modification of service items to service roster. Extension of XMPP [29] suggests the use of an attribute *"action"* with the value as add or delete or modify. The message stanza has an x tag, which contains items for sharing. Following is a message stanza example exchanged for roster sharing between mobile devices.

```
<message from="calculator@testenv.mobile
           /ProgrammerCalc"
  to="calculator@testenv.mobile
           /ScientificCalc">
  <body>Roster to add</body>
  <x xmlns="http://jabber.org/protocol
            /rosterx">
    <item
      action="add"
      jid="calculator@testenv.mobile
           /ProgrammerCalc/BinConvWS"
      WSDesc="Binary Converter Service">
    </item>
    <item
      action="add"
      jid="calculator@testenv.mobile
             /ProgrammerCalc/GCDWS"
      WSDesc="GCD calculation">
    </item>
  </x>
</message>
```

E. SETUP

The above comparisons were done in restricted lab environment. For the UDDI registry server jUDDI [30] version 3.1.5 was used. We then deployed UDDI server on a Cent OS Linux Server 6.2 with Apache Tomcat version 6.0.26. This server has Java run time environment version 1.7.0_45. SoapUI [31] version 4.6.3 was deployed on a Windows 7 machine for invoking and inspecting web services offered by UDDI, this simulated web service consumer behavior.

For the XMPP server we chose Apache Vysper [32] from several available options such as tigase, openfire, ejabberd. The reason being that Apache Vysper and Apache jUDDI are from the same foundation. This helped in the initial benchmarking. The other reason is that Apache Vysper was able to readily embedded into the existing set of web services. For the XMPP server and web service we used the Java run time environment version 1.7.0_45.

IV. APPLICATIONS

The proposed architecture can have several applications. It would enable mobile devices to more efficiently host web services and publish information on these using XMPP. A few scenarios where this architecture would come in handy are:

- *War front activities:* The war-front is a place where it is difficult or even impossible to access or setup fixed infrastructure for hosting web services. The proposed architecture would be very handy in such locations. With multiple devices present in such areas hosting secure web services for enemy troop location, discrete terrain map, war front update, armory updates, communicating war instructions could be made possible. There is a greater chance of mobile devices entering or leaving the network frequently and randomly. The proposed architecture could provide web service availability information with ease in such scenario. Moreover registering a new mobile device is fast and with minimum hassle.

- *Disaster relief:* Our architecture could prove useful in places where existing infrastructure has been demolished by disaster or natural calamity. It could be used by the disaster response team or local volunteers, who are hosting web services for disaster management, relief plans update, damage study etc. using their mobile devices. The proposed architecture could help directory in providing a flexible yet trustworthy information on web services in the vicinity.
- *Custom web service for any computation:* Small or new business organizations could potentially benefit from this architecture by being able to publish their hosted web services with minimal infrastructure and minimal costs. The proposed architecture could enable new businesses to provide a feed of their working and web services to customers.

V. CONCLUSION

In this paper we proposed a service discovery approach that make use of XMPP (Extensible Messaging and Presence Protocol) for web services hosted over mobile devices. The web service discovery mechanisms applied to traditional services such as UDDI and other centralized registry systems not fit for unreliable and dynamic mobile environments. Hence we proposed a new architecture for managing web service directories using XMPP. XMPP has been shown to be effective in managing the service directory, sending service updates, and service availability information. Service availability awareness is crucial in mobile environments as the network is nomadic and dynamic. The proposed architecture works on top of existing networking protocols and works well with available web service protocols. We intend to extend our research and explore possibilities in the following areas: personalized and subscription based web service rosters, QoS aware web service directory in decentralized environments.

REFERENCES

[1] K. Mohamed and D. Wijesekera, "A lightweight framework for web services implementations on mobile devices," in *Mobile Services (MS), 2012 IEEE First International Conference on*, 2012, pp. 64–71.

[2] M. Adacal and A. B. Bener, "Mobile web services: A new agent-based framework," *Internet Computing, IEEE*, vol. 10, no. 3, pp. 58–65, 2006.

[3] C. Riva and M. Laitkorpi, "Designing web-based mobile services with rest," in *Service-Oriented Computing-ICSOC 2007 Workshops*. Springer, 2009, pp. 439–450.

[4] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services: Concepts, Architectures and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[5] Oasis, "Uddi version 3.0.2 spec technical committee draft [last accessed january 30, 2014]," http://uddi.org/pubs/uddi-v3.0.2-20041019.pdf, 2004.

[6] M. P. Papazoglou, "Service-oriented computing: Concepts, characteristics and directions," in *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*. IEEE, 2003, pp. 3–12.

[7] A. B. Mnaouer, A. Shekhar, and Z. Y. Liang, "A generic framework for rapid application development of mobile web services with dynamic workflow management," in *Services Computing, 2004.(SCC 2004). Proceedings. 2004 IEEE International Conference on*. IEEE, 2004, pp. 165–171.

[8] S.-T. Cheng, J.-P. Liu, J.-L. Kao, and C.-M. Chen, "A new framework for mobile web services," in *Applications and the Internet (SAINT) Workshops, 2002. Proceedings. 2002 Symposium on*. IEEE, 2002, pp. 218–222.

[9] S. Berger, S. McFaddin, C. Narayanaswami, and M. Raghunath, "Web services on mobile devices-implementation and experience," in *Mobile Computing Systems and Applications, 2003. Proceedings. Fifth IEEE Workshop on*. IEEE, 2003, pp. 100–109.

[10] S. N. Srirama, M. Jarke, and W. Prinz, "Mobile web service provisioning," in *Telecommunications, 2006. AICT-ICIW'06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*. IEEE, 2006, pp. 120–120.

[11] R. Tergujeff, J. Haajanen, J. Leppanen, and S. Toivonen, "Mobile soa: service orientation on lightweight mobile devices," in *Web Services, 2007. ICWS 2007. IEEE International Conference on*. IEEE, 2007, pp. 1224–1225.

[12] L. Li and W. Chou, "Cofocus-compact and expanded restful services for mobile environments." in *WEBIST*, 2011, pp. 51–60.

[13] F. AlShahwan and K. Moessner, "Providing soap web services and restful web services from mobile hosts," in *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on*. IEEE, 2010, pp. 174–179.

[14] K. Elgazzar, P. Martin, and H. S. Hassanein, "Personalized mobile web service discovery," in *Services (SERVICES), 203 IEEE Ninth World Congress on*. IEEE, 2013, pp. 170–174.

[15] E. Al-Masri and Q. H. Mahmoud, "Mobieureka: an approach for enhancing the discovery of mobile web services," *Personal and Ubiquitous Computing*, vol. 14, no. 7, pp. 609–620, 2010.

[16] B. Sapkota, D. Roman, S. R. Kruk, and D. Fensel, "Distributed web service discovery architecture," in *Telecommunications, 2006. AICT-ICIW'06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*. IEEE, 2006, pp. 136–136.

[17] P. Saint-Andre, "Rfc 3920: Extensible messaging and presence protocol (xmpp): Core," Internet Engineering Task Force (IETF) proposed standard, Tech. Rep., 2004.

[18] P. Saint-Andre, "Rfc 6120: Extensible messaging and presence protocol (xmpp): Core," Internet Engineering Task Force (IETF) proposed standard, Tech. Rep., 2011

[19] P. Saint-Andre, "Xep-0174: Serverless messaging," Standards track, XMPP Standards Foundation, 2008.

[20] J. Hildebrand, P. Millard, R. Eatmon, and P. Saint-Andre, "Xep-0030: service discovery," 2008.

[21] M. Miller, "Xep-0050: Ad-hoc commands," 2005.

[22] F. Forno and P. Saint-Andre, "Xep-0072: Soap over xmpp," 2005.

[23] D. Bernstein and D. Vij, "Intercloud directory and exchange protocol detail using xmpp and rdf," in *Services (SERVICES-1), 2010 6th World Congress on*. IEEE, 2010, pp. 431–438.

[24] S. N. Srirama, C. Paniagua, and J. Liivi, "Mobile web service provisioning and discovery in android days," in *Proceedings of the 2013 IEEE Second International Conference on Mobile Services*. IEEE Computer Society, 2013, pp. 15–22.

[25] E. P. Saint-Andrew, "Rfc 3921: Extensible messaging and presence protocol (xmpp): Instant messaging and presence, october 2004," Internet Engineering Task Force (IETF) proposed standard, Tech. Rep., 2004.

[26] P. Hancke and D. Cridland, "Bidirectional server-to-server connections," 2012.

[27] A. Michlmayr, F. Rosenberg, C. Platzer, M. Treiber, and S. Dustdar, "Towards recovering the broken soa triangle: a software engineering perspective," in *2nd international workshop on Service oriented software engineering: in conjunction with the 6th ESEC/FSE joint meeting*. ACM, 2007, pp. 22–28.

[28] P. Millard, P. Saint-Andre, and R. Meijer, "Xep-0060: Publish-subscribe," *Jabber Software Foundation*, 2006.

[29] P. Saint-Andre, "Xep-0144: Roster item exchange," 2005.

[30] "Apache juddi [last accessed january 30, 2014]," https://juddi.apache.org/.

[31] "Soapui [last accessed january 30, 2014]," http://www.soapui.org/.

[32] "Apache vysper [last accessed january 30, 2014]," http://mina.apache.org/vysper-project/.