Providing SOAP Web Services and RESTful Web Services from Mobile Hosts
Feda AlShahwan, Klaus Moessner
Centre for Communications Systems Research, University of Surrey, Guildford, Surrey GU2 7XH, UK
{F. AlShahwan, K. Moessner} @surrey.ac.uk

Abstract—providing non interrupted Web Services from resource limited mobile devices in a light weight processing manner is a significant need. This paper takes a step towards achieving this goal. An implementation of two frameworks for providing Web Services from resource constraints devices is presented. The first framework is implemented to provide SOAP Web Services and the second one is implemented to provide RESTful Web Services. Each of those two frameworks has its own features and shortcomings. This paper provides a comparison between SOAP-based frameworks and RESTful-based frameworks. This comparison is carried out to allow us to decide on the framework that best suits mobile environment capabilities and fulfils our requirement of providing mobile Web Services continuously with a light weight processing requirement. This comparison is undertaken and analyzed using three different test scenarios. Our preliminary work shows that a RESTful-based MWSF is more suitable for mobile environments.

Keywords—RESTful-based MWSF; SOAP-based MWSF

I. INTRODUCTION

A Mobile Web Services framework allows deploying, publishing, discovering and executing of Web Services in a mobile communications environment using standard protocols. This technology has evolved from advances in the mobile device technology, rapid growth of Web Services development and progression of wireless communication in parallel with widespread use of internet applications. A mobile Web Service is an evolving area that affects real life daily applications. However it is still in its early stages and there are lots of challenges to be overcome. Those challenges result from mobile resources constraints, mobility issues and intermittent wireless network.

Web Services can be classified into two main categories: RESTful and SOAP-based Web Services. This classification is based on the architectural style used in the implementation technology.

SOAP stands for Simple Object Access Protocol. It is an object oriented technology that defines a standard protocol used for exchanging XML-based messages. It is defined as protocol specification for exchanging structured information in the implementation of Web Services in computer networks [1]. The specification defines an XML based envelope for exchanging messages and the protocol defines a set of rules for converting platform specific data types into XML representations.

REST stands for Representational State Transfer; it is a resource oriented technology and it is defined by Fielding in [2] as an architectural style that consists of a set of design criteria that define the proper way for using web standards such as HTTP and URIs. Although REST is originally defined in the context of the Web, it is becoming a common implementation technology for developing web services. RESTful Web Services are implemented with Web standards (HTTP, XML and URI) and REST principles. REST principles include addressability, uniformity, connectivity and stateless. RESTful Web Services are based on uniform interface used to define specific operations that are operated on URL resources.

Both SOAP and RESTful-based Web Services are used for implementing Web Services. However, each has its own distinct features and shortcomings that make it more or less suitable for certain types of application as shown in Table 1.

Table 1: Comparison of SOAP/RESTful-based Web Services

<table>
<thead>
<tr>
<th>Criteria</th>
<th>SOAP-based WS</th>
<th>RESTful-based WS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server/ Client</td>
<td>Tightly coupled</td>
<td>Loosely coupled</td>
</tr>
<tr>
<td>URI</td>
<td>One URI representing the service endpoint</td>
<td>URI for each resource</td>
</tr>
<tr>
<td>Transport Layer Support</td>
<td>All</td>
<td>Only HTTP</td>
</tr>
<tr>
<td>Caching</td>
<td>Not Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Interface</td>
<td>Non Uniform Interface (WSDL)</td>
<td>Uniform Interface</td>
</tr>
<tr>
<td>Context aware</td>
<td>Client context aware of WS behaviour</td>
<td>Implicit web service behaviour</td>
</tr>
<tr>
<td>Data Types</td>
<td>Binary requires attachment parsing</td>
<td>Supports all data types directly</td>
</tr>
<tr>
<td>Method Information</td>
<td>Body Entity of HTTP</td>
<td>HTTP Method</td>
</tr>
<tr>
<td>Data Information</td>
<td>Body Entity of HTTP</td>
<td>HTTP URI</td>
</tr>
<tr>
<td>Describing Web Services</td>
<td>WSDL</td>
<td>WADL</td>
</tr>
<tr>
<td>Expandability</td>
<td>Not Expandable (No hyperlinks)</td>
<td>Expandable without creating new WS (using xlink)</td>
</tr>
<tr>
<td>Standards used</td>
<td>SOAP specific standards (WSDL, UDDI, WS-Security)</td>
<td>Web standards (URL, HTTP methods, XML, MIME Types)</td>
</tr>
<tr>
<td>Security/Confidentiality</td>
<td>WS-security standard specification</td>
<td>HTTP Security</td>
</tr>
</tbody>
</table>

This paper discusses those features and helps understanding of the features and shortcomings to assist in selecting the
most appropriate architectural style that will be used as a tool for building and providing mobile Web Services.

The paper is organized as follows. In the next section a short introduction to the main issues that address provision of Web Services from mobile devices. In addition it presents some real applications that would benefit from the work explained in this paper. Then the current state of the art for providing Web Services from mobile devices is described in brief. This is followed by a presentation of the two different mobile host frameworks and an outline of their features and issues. Moreover a critical analysis and comparison between the two framework works is given. Finally some conclusions from the preliminary work are presented in the last section.

II. BACKGROUND AND MOTIVATION

The integration of Web Services with mobile devices has many useful benefits. First it supports automatic and autonomous self configuring distributed systems without interfering with the main functionality of the mobile host which is making phone calls. Also hosting Web Services on mobile devices has an enormous number of useful applications. For example location based applications can be provided from mobile devices. This has proved to enhance the performance of companies who have employees deployed in the field. For example a mobile host with built in Global Positioning System GPS receiver allows tracking of products and goods.

Health care applications are further evidence of the kind of application provided by hosting Web Services from mobile devices. They might be useful for both doctors as well as patients. For example deploying an appropriate service on a doctor’s mobile allows tracking professionals’ location and context to handle emergency cases. Also health care services can be extended and provided from patients’ mobile devices. This takes place by exposing a remote tele-monitoring service on the patient’s mobile host that allows monitoring their conditions using log files with the aid of some measurement devices[3]. Moreover mobile Web Services can be useful in polling-based applications that require using and triggering the most recent data which is changing dynamically. Context-based applications are another application discipline that benefits from hosting Web Services from mobile devices. Accessing the user profile of the mobile host and sharing the contents with others could be a useful application. However these frameworks are based on legacy implementation. In addition, the resource limited mobile devices, the intermittent network connections and the frequent context and location change of mobile host act as a barrier against the easy development of this area.

In addition to the large number of useful applications that can be provided from hosting services on resource constrained mobile devices, the continuous provision of non interrupted location based services from mobile devices is important. The significance of this evolves from the need to provide the latest instant information before it becomes obsolete. However the volatile intermittent wireless environment and the limited mobile battery life impede the reliable continuous provisioning of these services. Another important motivation that leads us to this research is that the existing mobile hosting Web Service frameworks are built upon using heavy weight SOAP parsers to process the requests received by the limited processing power mobile devices. Hence it will consume the available limited resources and there will not be free resources for the mobile device to perform its core functionality. As a result the overall performance will degrade. Thus a framework that demands light weight processing has to be build that is based on REST architectural design.

Researchers have explored the possibility of providing SOAP-based Web Services from mobile hosts. The explored approaches allow providing mobile Web Services either in client/server environment [4-7] or P2P network environment [7-12] Riva et al.[13]have investigated mobile RESTful Web Services and developed a prototype for providing RESTful photo Web Service on standard devices and accessed it by different mobile clients. However they focused on consuming RESTful Web Services from mobile devices and did not address the provision of services from mobile host. Moreover Li et al. [14] tackle the adaptive and dynamic provisioning of RESTful-based Web Services and they have presented RESTful-based framework that provides Web Services to mobile clients but they did not allow providing these services from mobile devices, our solution builds beyond consuming mobile Web Services, it allows consuming and providing Web Services. The objective of our work is to reproduce and extend researchers’ work for developing SOAP-based mobile Web Services framework. A further objective is to explore the feasibility of providing RESTful Web Services from mobile devices and to compare between the two frameworks in terms of performance, scalability, reliability and resource consumption. This comparison is needed to allow us to define the most suitable tool for facilitating the provision of continuous reliable services in a light weight processing manner in mobile network environment. This definition is based on the acquired testing results.

III. IMPLEMENTATION

Web Services are not explicitly defined for the mobile wireless environment. The current standard Web Service frameworks are developed for static servers. In addition these standard frameworks are too large to be deployed on resource constraint mobile devices and they require a running time environment that is not available on mobile devices. Also providing Web Services from mobile hosts consumes a large amount of resources and drains the batteries within a short period of time.

Two frameworks have been developed to allow deploying, providing and executing Web Services from mobile devices. One supports RESTful-based mobile Web Services (RESTful-based MWSF) and the other supports
SOAP-based mobile Web Services (SOAP-based MWSF).

In implementing our frameworks, Java for Mobile Edition JME is used as the best language for launching applications on limited resource mobile devices. JME defines two configurations: the Connected Device Configuration (CDC) and the Connected Limited Device Configuration (CLDC). In this research CLDC has been selected because it is a low level specification, suitable for wide range of mobile devices with limited memory capacity. Thus CLDC achieves scalability and generality. APIs and libraries are added to support more features through Mobile Information Device Profile (MIDP). In this research MIDP 2.0 is chosen because it supports devices with limited network communication resources and device internal resources. Also it provides more networking functionality and it supports HTTP protocol. In addition it supports the Server Socket connection that is required for implementing mobile server. In general the two frameworks are identical in the main architecture. However they differ in the details of handling and parsing the request. Each framework consists of five main building blocks:

1. Web ServiceServlet: It deploys new services into the mobile device and invokes the requested service. It also supports the flexibility of allowing Web Service developers to customize the particular handling of requests and responses.

2. HTTP Listener: The main functions of it include listening to incoming requests through ServerSocket class, accepting incoming client’s requests, initiating new thread for each request to support concurrency and creating input and output stream for communication.

3. Request Handler: The main task for it is to process the request. However the way this task is carried out is different between the two frameworks. For example in SOAP-based MWSF the request handler will unwrap the incoming HTTP POST request to extract the hidden SOAP envelope then it will dispatch the envelope to the message parser. However the request handler for RESTful-based MWSF will extract the HTTP request directly and send it to the message parser.

4. Parser Module: The main function for it is to get the needed information for invoking a Web Service such as the name of the service, service URL and some parameters. Then the extracted information is sent to the ServiceServlet. However the way this is performed is different between the two frameworks. In SOAP-based MWSF, the SOAP parser deserializes the SOAP object and maps the data types into Java objects using kSOAP2 and kXML2 that are open source APIs for SOAP parsing. However in RESTful MWSF we have created our own string manipulator based parser. This parser will extract the needed information that resides explicitly in the client’s request.

5. Response Composer: It is responsible for interpreting the result then sending it back to the client.
So on a first claim the difference between the two architectures are fairly similar and there is no appearance different in complexity but the major different comes when we test the architectures’ performance, scalability and amount of resource consumption.

IV. RESULTS AND EVALUATION

The two frameworks are evaluated to confirm the correct behaviour of the designed frameworks and to acquire experimental data that allows critical analysis of both architectures. The evaluation is conducted using a small test-bed that consists of a mobile host developed on n80 Nokia mobile device running Symbian OS, MIDP 2.0 profile. It is connected in a wireless network through built in IEEE 802.11b interface and it provides services to a client that is simulated using Sun Wireless Toolkits 3.0 emulator.

The evaluation involves three different scenarios. The first set of experiments is done to test the performance of the mobile host. Performance is analyzed through measuring the effect of varying the request message size on the average processing time. Results in Fig. 3 and Fig. 4 show that the average processing time increases when the request message size increases.

Moreover the average processing time for SOAP-based MWSF is larger than the average processing time for RESTful-based MWSF for the same message request. This is because processing SOAP requests requires heavy weight parsers to un-wrap the SOAP envelope from the incoming HTTP POST request, then de-serialize the SOAP object and map the data types of the XML based message into Java objects. This is done to extract the hidden information needed for invoking the required Web Service. However processing RESTful requests uses light weight parser that is created by us to extract the information required for invoking a specific Web Services. Moreover the required information resides explicitly on the HTTP request. Thus RESTful-based MWSF has better performance than SOAP-based framework.

The second scenario evaluates reliability and scalability of the frameworks. This test is accomplished through testing concurrency where a number of clients send requests to the same host simultaneously. This is accomplished through initiating threads and loops on the client emulator. Then the average process time for each concurrent request is calculated. Results Fig. 5 and Fig. 6 show that as the number of concurrent requests increases, the average process time also increases. This increase is more obvious in SOAP-based framework where more time is consumed to parse the SOAP envelope and to manage the threads. However we observe that the increase in RESTful-based MWSF is almost steady. This is because RESTful Web Services support caching and demand light processes power. Hence RESTful-based MWSF is more rigid and robust to changes in the number of concurrent request.
After that, the two mobile hosts are stressed by adding more concurrent requests to measure the threshold value. It is defined as the maximum number of concurrent requests that can be handled without failure. It is observed that in Table 2 SOAP-based MWSF starts to reject requests earlier when the threshold is beyond 60 but RESTful-based MWSF starts to reject requests when the threshold is beyond 80. This is expected because processing SOAP-based requests requires more time. Thus the consumed response time is larger and the server queue of the SOAP-based framework will be occupied and filled within a short period of time. As a result there will be no more resources to accept new connections. Thus RESTful-based MWSF is more scalable and reliable than SOAP-based MWSF.

Table 2: Comparison of rejected requests between SOAP-based and RESTful-based MWSFs

<table>
<thead>
<tr>
<th>No of Requests</th>
<th>Avg rejected requests (SOAP)</th>
<th>Avg rejected requests (REST)</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>80</td>
<td>59</td>
<td>4</td>
</tr>
<tr>
<td>100</td>
<td>64</td>
<td>9</td>
</tr>
<tr>
<td>120</td>
<td>86</td>
<td>14</td>
</tr>
</tbody>
</table>

The last scenario is for testing resource consumption and measuring memory footprints. Results in Fig. 7 illustrate that the amount of consumed memory during processing Web Service requests is increased as the message size increases. As shown in the graph the amount of consumed memory in SOAP-based framework is much larger than the amount of consumed memory in RESTful-based framework for the same message size. The reason for this is that SOAP-based framework demands more memory footprint during processing. This consumed memory footprint is used to store general temporary parsed objects and to load the classes and kSOAP and kXML libraries.

V. CONCLUSION

This paper has reproduced and extended the approach followed by almost all researchers who attempt to provide and execute mobile Web Services. This approach is mainly based on SOAP technology. Then the feasibility of providing and executing RESTful-based Web Services from mobile devices is also explored in this paper. To the best of our knowledge, it is the first work that investigates providing RESTful-based Web Services from mobile devices.

The two frameworks that allow deploying, providing and executing SOAP/RESTful-based Web Services from resource limited mobile devices in an intermittent wireless network have been analyzed and compared.

This analysis is needed to select the most appropriate implementation technology that suits limited mobile environment capabilities and to facilitate providing mobile Web Services in a continuous light weight processing manner.

Our preliminary work shows that RESTful-based MWSFs have proved to be more suitable for the mobile environment because it does not require large weight parsers. In addition, it supports caching which will save the limited network bandwidth and increase reliability and scalability. Also it does not consume large amounts of mobile resources. Another feature of RESTful Web Services is the loosely coupled relation between the server and client because of the uniform interface that adds a balance towards using it for the mobile environment. This lightens the burden on limited resource mobile servers and allows continuous provision of mobile Web Services and only requires light weight processing power.

However, there are still issues that need to be solved in the future such as confidentiality and security levels. Moreover another limitation with RESTful Web Services is that they are only used for HTTP transport layer. Addressing mobility issues in mobile Web Service provisioning and adaptation of mobile Web Services is another area for future work.
REFERENCES

last access date 17-Feb-2010


Approach for Tele-monitoring of Measurement Devices. Available:

Device Hosted Web Service and Its Performance Analysis," in High
Performance Computing and Communications, 2008. HPCC '08. 10th

Execution Environment for Hosting Mobile Web Services," in Services
315-322.

services on mobile devices-implementation and experience," in Mobile

for Mobile Web Service Provisioning," in Enterprise Distributed Object
Computing Conference Workshops, 2006. EDOCW '06. 10th IEEE

[8] G. Gehlen and L. Pham, "Mobile Web services for peer-to-peer
applications," in Consumer Communications and Networking Conference,

Analysis of a SOAP Server for Mobile Devices". in Proc.11th European

web services" Computer Science - Research and Development, pp. 199-

Communication and Deployment of Time Independent Mobile Web
Services," in Web Services, 2008. ICWS '08. IEEE International
Conference on, 2008, pp. 797-800.

Architecture for Mobile Web Services via Converged Cellular and Ad Hoc
Networks," presented at the Proceedings of the 2008 The 3rd International

Services with REST," in Service-Oriented Computing - ICSOC 2007
Workshops: ICSOC 2007, International Workshops, Vienna, Austria,
September 17, 2007, Revised Selected Papers, ed: Springer-Verlag, 2009,
p. 439-450.

Services in Mobile Wireless Networks," in E-Business and Information
1-5.