Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 10 (2012) 744 – 751

Procedia Computer Science

The 9th International Conference on Mobile Web Information Systems

Performance Analysis of Web Services on Mobile Devices

KamalEldin Mohamed[a], Duminda Wijesekera[b] a*

[a,b]George Mason Univeristy, 4400 University Dr., Fairfax, VA 22030, USA

Abstract

With the recent rapid development of mobile devices in terms of processing power, memory and storage capabilities coupled with the advancements of wireless technology in terms of higher data transmission rates such as 3G and 4G, it has now become feasible to host Web services on mobile devices. In this paper we propose a lightweight framework for hosting Web services on mobile devices. We further evaluate and provide a comparative analysis for hosting RESTful Web services versus SOAP-based Web services on our framework. Our experimental results and analysis indicate that RESTful Web services are less resource-consuming and more efficient for the implementation and provisioning of Web services from resource-constrained mobile devices.

Keywords: Web services; SOAP; REST; Mobile web server; Lightweight framework.

1. Introduction

Web Services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or Web-based Web services that are built on top of open standards such as TCP/IP, HTTP, Java, and XML [1].

As mobile devices and wireless technologies continue to rapidly grow and significantly advance, Web services technology recognizes mobile computing as an area to which it should expand. When combing smart mobile devices and Web services; i.e. running a Web service on a mobile device, one can greatly increase the functionality of mobile devices to interact with its environment. This makes a wide range of new functionalities and features possible especially in the field of ubiquitous computing.

While consuming Web services from mobile devices is more common nowadays, hosting Web services on mobile devices can further provides many useful benefits. For example, it enables the owner of the mobile device to have ultimate control for administering, managing and securing their mobile-hosted Web services anytime anywhere. In location-based applications, hosting Web services on a mobile device can help tracking individuals' exact locations at any time; skilled professionals such as doctors and nurses can be more quickly located in case of emergencies. In a supply chain management system, the updated services offered by an individual can be available through the Web services hosted on his/her mobile device. Moreover, mobile Web services can be useful in polling-based applications that require using and triggering the most recent data which is changing dynamically [2].

* Corresponding author. Tel.: +1-571-244-1434; *E-mail address*: kmohamed@gmu.edu

doi:10.1016/j.procs.2012.06.095

Running Web services on mobile devices opens a wide range of new possibilities and solves heterogeneity and interoperability issues. However the performance characteristics are important to ensure the mobile device is able to handle multiple requests while the device itself is able to function normally. In this paper we propose a lightweight framework for hosting Web services on mobile devices. Our framework supports and facilitates the provision of SOAP-based as well as RESTful Web services from smart mobile devices without the need for intermediaries. To evaluate the overall performance of our proposed framework, it has been experimented on virtual as well as on physical mobile devices. Our experimental results indicate that RESTful Web services are more suitable and less resource consuming compared to SOAP-based Web service for the implementations of Web services on mobile devices environment.

2. Related Work

The advancements in mobile devices and broadband wireless technologies increased the interests of research communities towards mobile Web services.

Asif, M., Majumdar, S. and Dragnea, R. [3] propose a lightweight Web service provider toolkit that can be implemented in any object oriented language to facilitate the hosting of Web services on a number of handheld devices. The toolkit is based on a stack of four layers. The "Transport Layer" in which HTTP is used as the main transport mechanism for the exchange of SOAP messages. The "Security Layer" uses SSL and TLS to secure the incoming and outgoing SOAP requests/responses messages. A "SOAP Engine Layer" parses the SOAP messages and uses RPC as the binding technique. The "Target Web service Layer" maintains a list of deployed Web services in an XML file. The toolkit scalability was tested on a PDA running Windows Mobile 5.0 with an Intel processor at 624MHz and 64MB memory, but it was found to be limited to support only a dozen concurrent Web services client requests.

Kim, Y. and Lee, K. [4] propose a framework for hosting Web services on mobile devices. It consists of several components such as a "Request/response Handler", a "SOAP Engine", a "Publication/discovery Manager", a "Migration Manager" and a "Context Manager". The framework was tested under a real-world application using Windows mobile phones and was proven to support Web services on a mobile environment. However, the framework still lacks a fine-grained migration policy as well as it uses Bluetooth as a communication media between the mobile clients and the mobile Web service host which makes the application of the framework very limited to short range connections.

Riva, C. and Laitkorpi, M. [5] investigate how to apply the REST principles to the design of mobile services. They identified several issues such as latency and data format that need particular attention when applying REST concepts to mobile environment. They also developed a prototype for providing a REST photo Web service and tested it with different mobile clients such as Nokia S60 smart phones. However, they only focused on consuming RESTful Web services on mobile devices and did not address the provision of Web services from a mobile host.

AlShahwan, F. and Moessner, K. [2] proposed two frameworks to allow the provisioning of mobile Web services, one framework is based on REST architectural principles and the other is based on SOAP. They conducted tests on a Nokia N80 mobile running Symbian OS. They have shown that RESTful based framework is more suitable for mobile environment.

3. Background

There are mainly two approaches for building and implementing Web services; SOAP-based and RESTful Web services. Both approaches have their distinct advantages and disadvantages to interfacing to Web services, but it is always up to the developer to decide on which approach could be best depending on each particular implementation scenario.

3.1. Traditional (SOAP-based) Web services

Simple Object Access Protocol (SOAP) can form the foundation layer of Web services protocol stack thereby providing a way to communicate between applications running on different operating systems with different technologies and programming languages. This XML-based protocol consists of three parts: an envelope, which

defines what to be included in the message and how this massage should be processed, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing procedure calls and responses [6][7]. Figures (1-a) and (1-b) show sample request and response SOAP messages where the payload is embedded within the SOAP envelope. Moreover; SOAP has three major characteristics: Extensibility (security and WS-routing are among the extensions under development), Neutrality (SOAP can be used over any transport protocol such as HTTP, SMTP or even TCP), and Independence (SOAP allows for any programming model) [7][8].

3.1.1. Advantages

- Portability: SOAP is a platform-independent and thus portable.
- Firewall-friendliness: SOAP is capable of getting past firewalls which are totally blocking for other protocols. This is possible due to using the HTTP protocol [7].
- Use of open standard: SOAP is based on the open Standard XML to format data. As a consequence, SOAP becomes easily extendable and well supported [7].
- Interoperability: SOAP relies on open instead of vendor-specific technologies and thus enables distributed interoperability and loosely coupled applications [6][7].
- Resilience to changes: It is unlikely that future modifications of SOAP infrastructure will have any impact on applications using the method, as long as no significant serialization changes are made to SOAP specification [7].

3.1.2. Disadvantages

- Operation interface: Useful information such as operation details and data are encapsulated within the services, just exposing only one endpoint of API and all operations use the POST method [9][10].
- Complexity: It is time-consuming to serialize and deserialize native languages into SOAP messages. Furthermore, the WS-* protocol stack is also complex so that only programmers can understand how to deploy a service [8][9].
- Interoperability: Since a specific service interface is defined for each service, a client must be bound to a specific WSDL. Once the WSDL has changed, the client has to follow these changes [9].
- Performance: Much information in the SOAP and WSDL is redundant and meaningless. It increases the network communication volume and server-side payload and it is difficult to support the proxy and cache servers because clients cannot identify the useful information straightforwardly from the URI and HTTP [9][10].

3.2. RESTful Web services

RESTful Web services are “Resources” that are identified by unique URIs. These resources are accessed and manipulated using a set of uniform methods (GET, POST, PUT and DELETE) where each resource has one or more representations (XML, JSON, Text, User-defined, etc) that are transferred between the client and the service during a Web service invocation. RESTful Web services are perceived to be simple because REST leverages the existing well-known W3C/IETF standards (HTTP, XML, URI and MIME type) as well as the necessary infrastructure that has already become pervasive [11][12]. As shown in Figures (1-c) and (1-d), requests and responses for RESTful Web services are typically HTTP messages that are far less in size compared to SOAP messages. Since in REST architecture a resource can be directly identified by its URI, therefore extensive SOAP parsing can be avoided that is required for invoking a service [13].

3.2.1. Advantages

- Addressability: Resources are marked with global URIs. They are accessible once they are exposed on the Web rather than require a separate resource discovery and location mechanism [6][10].
- Links and Connectedness: Resources are linked with each other by using hyperlinks which point to valid future states in representations. What most important is that, state transfer can be implemented by following the links [12].

- Statelessness: Each request includes all the necessary information for the servers to understand, so each transaction is independent and unrelated to previous ones. Servers do not need to keep states between requests [14].
- Uniform Interface: Resources are manipulated using a fixed set of HTTP methods (GET, PUT, DELETE and POST) without a special code to deal with each one. These methods (except POST) are Idempotent [10].

3.2.2. Disadvantages

- Encoding a large amount of input data in the resource URI is impossible because the server either refuses such requests or crashes [6].
- It may also be challenging to encode complex data structures into URI as there is no commonly accepted marshalling mechanism. Inherently, the POST method does not suffer from such limitations [6][10].

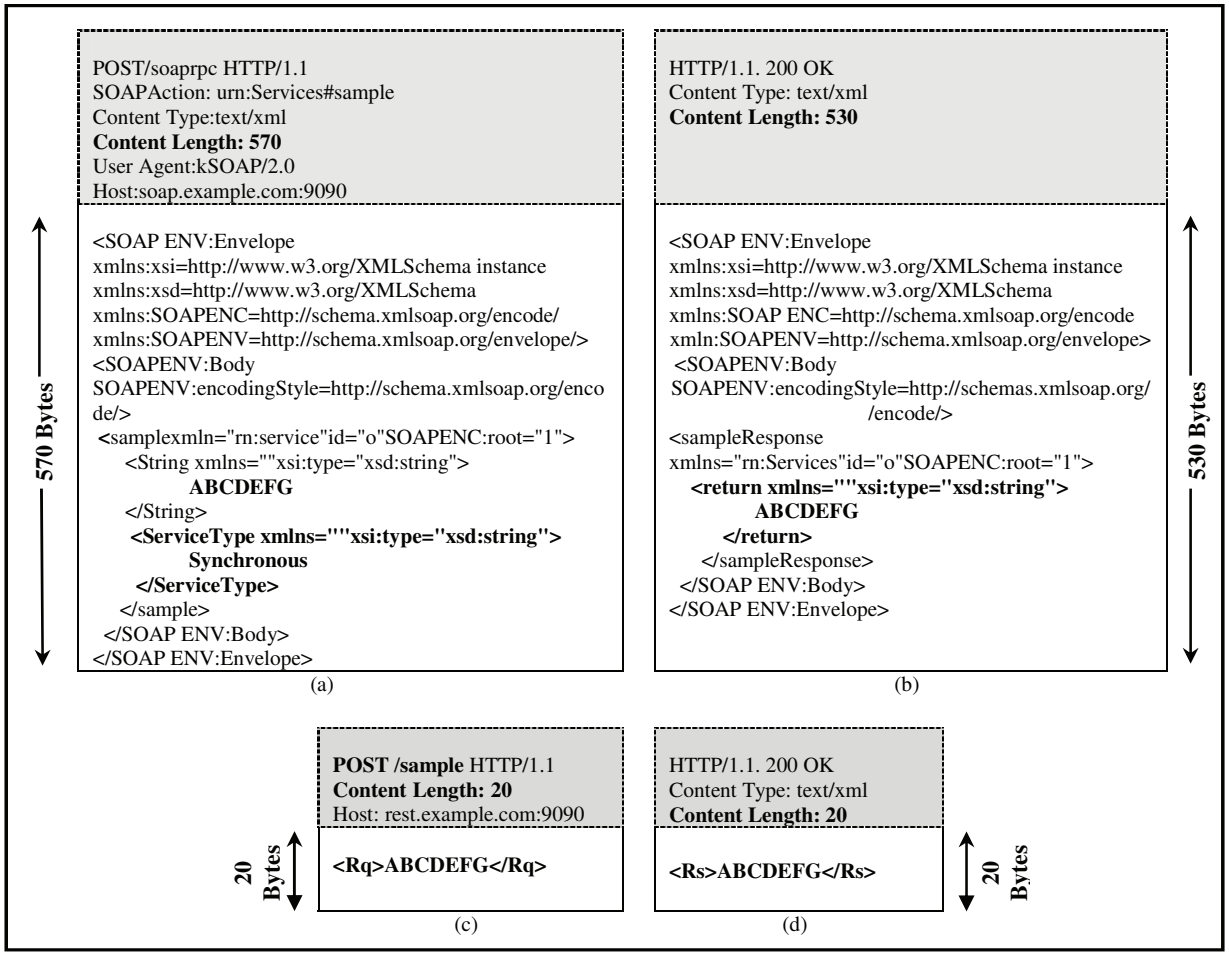

Figure 1. Sample SOAP web service (a) request; (b) response compared to sample RESTful web service (c) request; (d) response.

Based on the above analysis and while considering mobile devices environment in particular, we can find that direct implementations of SOAP-based Web services may introduce a significant and an unacceptable performance overhead on the resource-constrained mobile devices due to the increasingly thick SOAP messages and its expensive parsing requirements which demand for high resources (processor power, memory, battery and bandwidth). Conversely, REST architecture style supports direct interaction with Web services using the standard Uniform Resource Identifier (URI) and can greatly facilitate avoiding application performance degradation factors

such as SOAP message length and parsing. The flexibility and loose coupling that RESTful Web services can afford are beneficial to both, mobile Web services hosts as well as mobile clients.

4. A lightweight framework for mobile Web services

Our mobile Web service framework facilitates the hosting of SOAP-based as well as RESTful Web services into mobile devices. Figure 2. shows the framework building blocks:

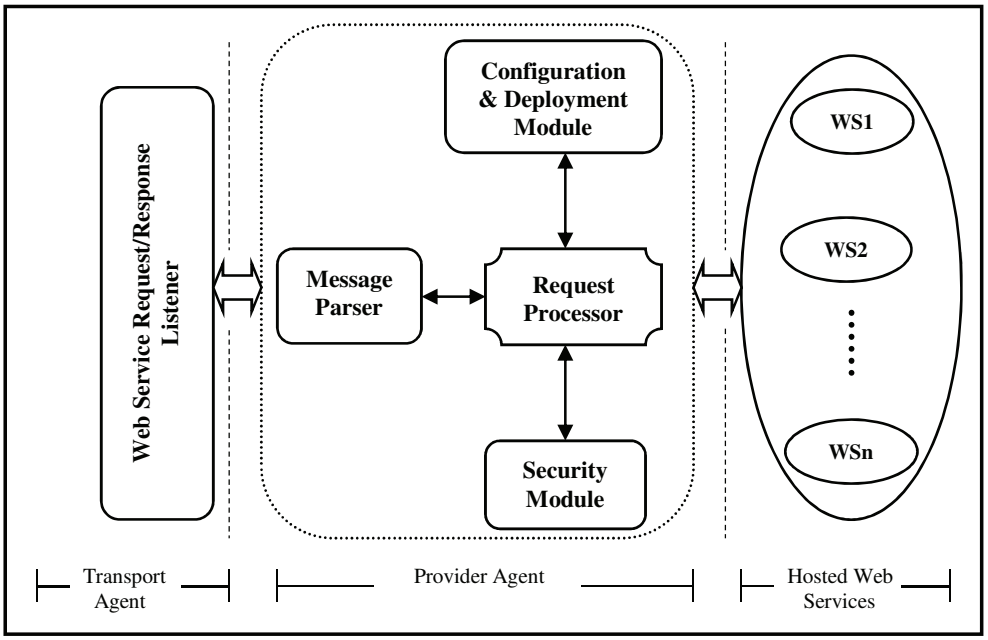

Figure 2. Framework building blocks for mobile web services host.

- **Web service request/response listener:** is capable of handling concurrent Web services requests and responses. Our "Request/Response Listener" transports clients Web service requests to the "Provider Agent" and returns the Web service responses back to the clients.
- **Request processor:** is used to analyze, process and coordinate all users' requests for all the locally hosted Web services. Its main functionalities are dispatching all incoming requests to be parsed, invokes the particular method on the requested Web services accordingly and generates the Web service responses to the clients. Further, if the requested Web service(s) require security, the "Request Processor" coordinates with the "Security Agent" for verifying and/or validating the required security parameters.
- **Configuration and deployment module:** is primarily responsible for configuring and deploying Web services into the mobile host using a lightweight Web server that consumes as minimum system resources. We use Jetty Web Server [15] that has a small footprint and capable of serving static and dynamic contents either from a standalone or embedded instantiations.
- **Message parser:** extracts and obtains all the values necessary to invoke the requested Web service. Our framework utilizes push and pull parsers that are far less memory consuming, easier and suitable to use in mobile environment compared to the DOM parsers that are more memory intensive [12]. For SOAP-based Web services, the "Message Parser" deserializes the incoming message requests into Java objects and serializes the Web service responses into response contents using kXML pull parser [16].
- **Security module:** provides a core set of security services for the hosted Web services that include authentication and authorization.
- **Hosted Web services:** acts as a container for all Web services that are deployed into and currently hosted by the mobile Web service host.

5. Experimental analysis and evaluation

5.1. Test bed setup and specifications

For all our experimental testing and analysis we setup a test bed that consists of the following components:

- **Testing Workstation**: HP Notebook with Windows7 Home Edition x64, 2.4GHz Processor, 4GB Memory.
- **Development Environment**: Eclipse Helios SR-2 with Android SDK 14 integrated.
- **Mobile Devices**: Physical and virtual Android 2.3.3 mobile devices with 256MB Memory, 100MB SD card.
- **Library**: Jetty Web server ver. 8.0.1, SQLite ver. 3.6.22 [18], HTTP client, kSOAP2 [17] and kXML2.
- **Development language**: Java.
- **Wireless LAN**: 54 Mbps bandwidth.
- **Web server stress tool**: Apache JMeter ver. 2.6 [19].

Our framework has been experimented on a test bed that consists of three Android mobile devices with one device acting as a Web service host and the other two devices are the Web service clients. Figure 3. shows the initial screens of the Android mobile devices; where the host Android is assigned a local IP address and a custom port number that Web service clients (within the same LAN environment) can connect to either wirelessly or via the Android Debugging Bridge (ADB) tools. We developed a "Dictionary" Web service in Java, as our Web service testing candidate where the "Word & its Meanings" are stored in a SQLight database that is integrated into the Android Web service host and can be exposed as either a RESTful or a SOAP-based Web service.

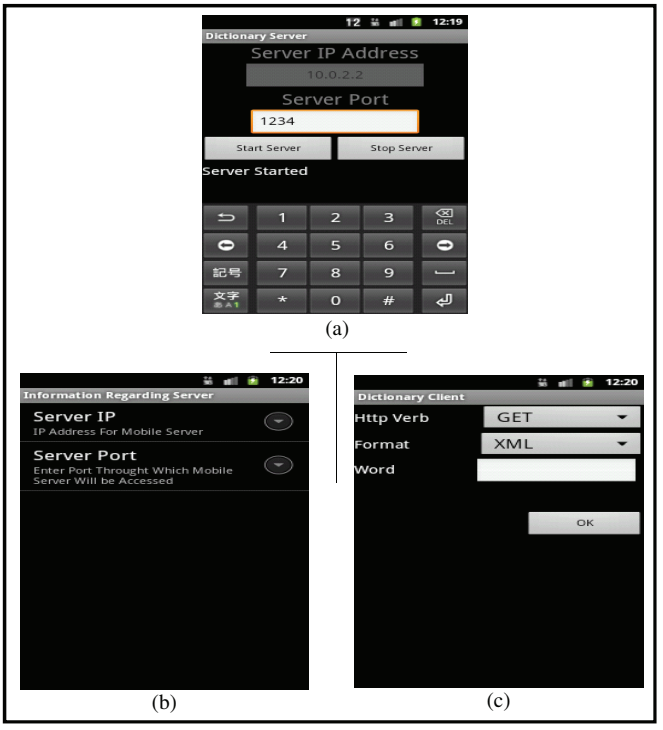

Figure 3. (a) Mobile web services host App. ; (b) & (c) Client App. initial screens.

5.2. Performance test

During this test, we measured the ability of the mobile Web services host on how well it performs under different load conditions with variable amount of stress level; i.e., the effect of variable message sizes with variable number of request loads [20]. For all the performance tests we conducted on our framework, we observed that RESTful Web service always outperform SOAP-based Web services. For example; Figure 4. show a performance load test during

an “HTTP GET” and an “HTTP PUT” Web service sessions with 21 Web service clients (users) ramping up at one user per second. Further, during several performance tests with higher loads we also observed that when the message size increases by a factor of 10, 15, and/or 20, the average processing time as well as the average processing time slightly increase in RESTful Web service and highly increase in SOAP-based Web services. This is due the fact that it takes more time to extract the SOAP envelop and parse the SOAP request, whereas in RESTful Web services the required data is already contained within the HTTP request body.

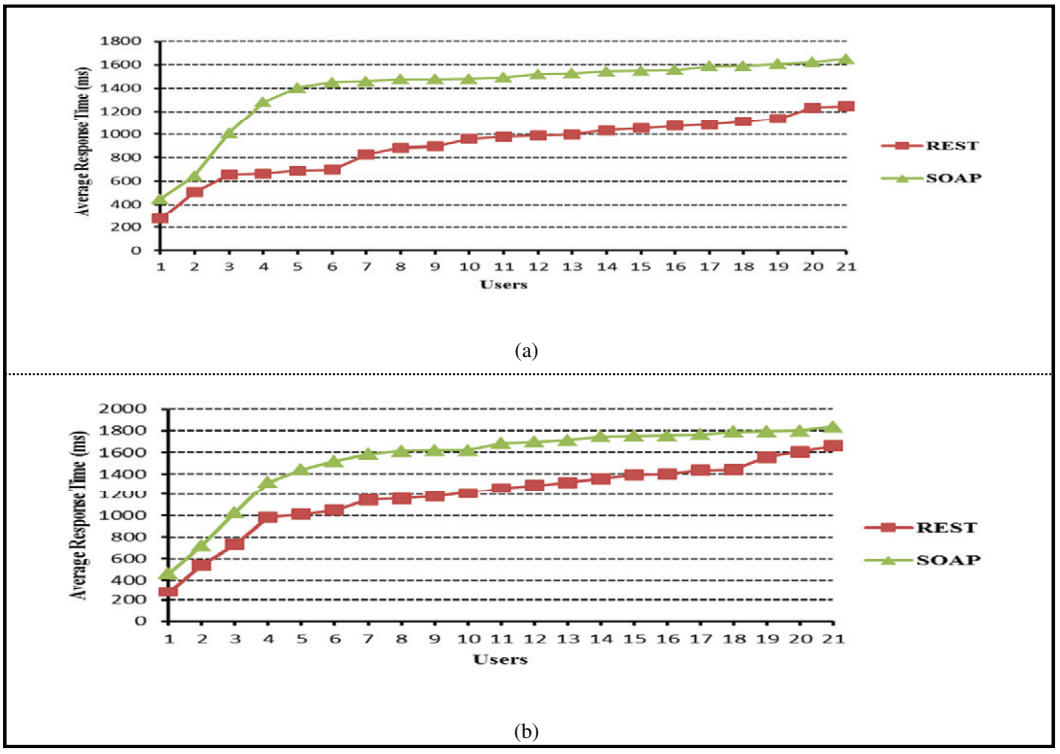

Figure 4. Mobile host average response time for SOAP-based and RESFul web services during concurrent (a) HTTP GET requests; (b) HTTP PUT requests.

5.3. Light- weightiness

Our framework has a small footprint of approximately 1.64-to-2.37 MB including all libraries and dependencies. However, at run time we carefully measured the internal resources utilization in terms of CPU usage and memory consumption. For all the tests we conducted, SOAP-based Web services consumed about 6-to-11% CPU power and 9-to-18 MB memory; However, in RESTful Web services resource consumption stayed consistently at a much lower rate where the CPU utilization remained between 1-to-3% of the overall mobile host CPU processing power and the memory consumption stayed within the range of 8-to-10MB.

5.4. Scalability test

Scalability is closely related to the performance of a Web service. During this test, we measured the total number of successful responses under a heavy load of simultaneous users’ requests. A Web service that serves users’ requests is not much of a use if many of these requests are not handled properly and returned with error responses [21]. For all the tests we carried out, RESTful Web services are also observed to be more scalable than SOAP-based Web services. The mobile Web service host have stayed highly responsive and error-free for about 68 concurrent users for RESTful Web services, whereas in SOAP-based Web services, the mobile host could not support more than 50 concurrent users without dropping Web service requests.

6. Conclusion

In this paper we proposed a lightweight framework for hosting and provisioning SOAP-based as well as RESTful Web services from mobile devices. Our framework is based on the existing Web services and mobile technology standards and is independent of any intermediaries. Our research study yielded that RESTful Web services, compared to SOAP-based Web services, are easier to build, deploy, publish and invoke specially in mobile devices environment. Our framework has been tested for a number of different scenarios; for the majority of our experimental analysis, RESTful Web services clearly outperformed and showed several advantages over using SOAP-based Web services for mobile devices. These advantages include less request/response message sizes, more scalability and less resource consuming which further reduce the overhead on mobile devices and wireless bandwidth making RESTful Web services attractive alternative for the implementations of Web services on resource-constrained mobile devices.

References

[1] IBM, **"**WebSphere Business Integration Adapters", Accessed May 2011 form http://publib.boulder.ibm.com/infocenter/wbihelp/v6rxmx/index.jsp?topic=/com.ibm.wbia_adapters.doc/doc/webservices/webservices17.htm

[2] AlShahwan, F., Moessner, K., "Providing SOAP Web Services and RESTful Web Services from Mobile Hosts", Internet and Web Applications and Services (ICIW), 2010 Fifth, PP. 174 - 179.

[3] Asif, M., Majumdar, S. and Dragnea, R., "Hosting Web Services on Resource Constrained Devices", ICWS 2007. IEEE International Conference on 9-13 July 2007. pp. 583-590.

[4] Kim, Y. and Lee, K., "A Light-weight Framework for Hosting Web Services on Mobile Devices", Web Services, 2007. ECOWS '07. Fifth European Conference on 26-28 Nov. 2007. PP. 255 - 263.

[5] Riva, C. and Laitkorpi, M., "Designing Web-Based Mobile Services with REST", Service-Oriented Computing - ICSOC 2007 Workshops, Lecture Notes in Computer Science, 2009, Volume 4907/2009.

[6] Albreshne, A., Fuhrer and Pasquier, J**. "**Web Services Technologies: State of the Art", 2009. Unpublished. Accessed May 2011 from http://diuf.unifr.ch/drupal/softeng/sites/diuf.unifr.ch.drupal.softeng/files/file/publications/internal/WP09-04.pdf

[7] "SOAP", accessed June 2011 from http://en.wikipedia.org/wiki/SOAP

[8] W3C Working Group, "Web Services Architecture", accessed May 2011 from http://www.w3.org/TR/ws-arch/#introduction

[9] Meng, J. , Mei,S. and Yan , Z., "RESTful Web Services: A Solution for Distributed Data Integration", Computational Intelligence and Software Engineering, 2009. CiSE 2009.

[10] Cox, J.., Harvey, D. and Ramsbrock, D. "SOAP vs. REST For Mobile Services" accessed May 2011 from http://blogs.captechconsulting.com/blog/jack-cox/soap-vs-rest-mobile-services

[11] Fielding, R.T., "Architectural styles and the design of network-based software architectures", PhD Thesis, University of California, Irving, 2000.

[12] Richardson, L. and Ruby, S., "RESTful Web Services"; First Edition, O'Reilly Media, 2007.

[13] Aijaz, F., Ali, S., and Chaudhary, M. and Walke, B., "Enabling High Performance Mobile Web Services Provisioning", In Proceedings of the 2009 IEEE 70th Vehicular Technology Conference Fall, p. 6, Anchorage, Alaska-USA,IEEE, 2009.

[14] Hamad, H., Saad, M. and Abed, R., "Performance Evaluation of RESTful Web Services for Mobile Devices", International Arab Journal of e-Technology, Vol. 1, No. 3, January 2010.

[15] "Jetty://", accessed June 2011 from http://www.eclipse.org/jetty/

[16] "kXML2", accessed July 2011 from http://kxml.sourceforge.net/kxml2/

[17] "kSOAP2", accessed Aug. 2011 from http://ksoap2.sourceforge.net/

[18] "Distinctive Features of SQlite", accessed Aug. 2011 from http://www.sqlite.org/different.html

[19] "Apche JMeter", accessed Oct. 2011 http://jmeter.apache.org/

[20] Tian, M., Voigt, T., and Naumowicz, T., Ritter, H. and Schiller, J. "Performance Considerations for Mobile Web Services", Computer Communications Journal, 27 (11), (2004). pp. 1097-1105.

[21] Kikkert, S., "Performance of Web services on Mobile Phones", 2010. Unpublished. Accessed Aug. 2011 from http://svn2.assembla.com/svn/thesis_rug/Sources/paper-sander.pdf