Performance Analysis of Ubiquitous Web Systems for SmartPhones

Katrin Hameseder, Scott Fowler* and Anders Peterson
Linköping University
Department of Science
and Technology
SE-601 74, Norrköping, Sweden
*Corresponding author: Email: scott.fowler@liu.se
Tel: (+46) 11 363298; Fax: (+46) 11 363270

***Abstract*— As smartphone clients are restricted in computational power and bandwidth it is important to minimise the overhead of transmitted messages. This paper identifies and studies methods that reduce the amount of data being transferred via wireless links between a Web service client and a Web service. The goal is to improve the end-to-end service execution time by reducing the bottleneck presented by the limited bandwidth in an ubiquitous environment. Measurements are performed in a real environment based on a web service prototype providing public transport information for the city of Hamburg in Germany, using actual wireless links with a mobile smartphone device. The existing SOAP based web service is evaluated against a REST based web service using the data exchange formats JSON, XML and Fast Infoset.**

***Keywords*: SmartPhones, XML, iPhones, Ubiquitous, Measurements, Web Service**

I. Introduction

The Internet growth over recent decades has induced increasing demands for high speed and ubiquitous access. With the introduction of smartphones (e.g. iPhone) the market for powerful mobile devices has exploded. Cisco predicted that mobile traffic would increase by a factor of 39 times between 2009 and 2014 [3]. In the third quarter of 2010, smartphones accounted for 19.3 percent of overall mobile phone sales [2]. Figure 1 shows the percent of worldwide smartphone sales based on operating systems which produced record sales of more than 81 million communication devices in the third quarter of 2010 [2]. Due to the limitations of mobile devices and particularly of smartphones, it is important to find efficient solutions to transfer information from a server to a mobile client.

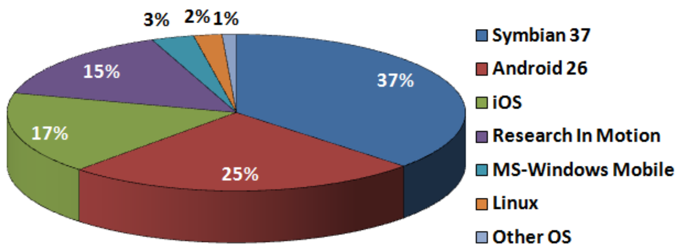

Fig. 1: *Share of worldwide smartphones end users by Operating System for 2010*

Vendors offering state-of-the-art smartphones have these devices running on different operating systems, which often leads to compatibility problems when porting desktop services. One possibility to avoid such problems is to use web services that act as mobile clients and provide the features of a desktop service for mobile devices independently of the operating system or client application of the mobile devices. However, limitations in the smartphones' CPU main memory and RAM strongly bound the capabilities for such services. Further, as mobile devices and particularly smartphones only have limited power supplies it is of importance to design the service in a way that ensures that as less power as possible is consumed and that the processed data is kept in case of power loss.

The focus of this paper is on Web services, and methods for improving their usability in mobile networks. Mobile devices usually connect to Web services over wireless connections. These wireless links lead to higher response times due to limited bandwidth and high latency compared to wired connections. This paper identifies and studies methods that reduce the amount of Web service data being transferred via wireless links to and from a Web service client. The goal is to improve the end-to-end service execution time by reducing the bottleneck presented by the limited bandwidth. As an additional benefit, the communication cost will be reduced on links that typically are charged on a per byte basis.

The contribution of this paper is an evaluation of response time and total bytes used as a means to enable access to Web services from mobile units with limited computational and memory resources, e.g. mobile smartphones. Measurements were performed in a real environment based on a web service prototype providing public transport (PT) information for the city of Hamburg in Germany using actual wireless links with a mobile smartphone device. Travel information is a good example of information requested from users en route and hence an application of special interest for smartphones.

An evaluation of the existing SOAP based web services of the public transport schedule system has been done in order to determine how to enable efficient access for smartphone clients. The use of the SOAP web service style leads to a high amount of overhead in the request and response messages and is not applicable for smartphone clients. Compares of

SOAP web services against REST web services using the data exchange formats JSON, XML and Fast Infoset were developed to verify that the use of a REST web service as well as the choice of the data format leads to improvements in terms of:

- transferred data volume
- serialisation/deserialisation time
- request–response time

The paper is organized as follow: In Section II we present work related to our contribution. Section III presents the experiment setup and the results are presented and discussed in Section IV and V. Section VI concludes the paper.

II. Related Work

Most of the available web services use XML for the representation of the data that has to be transferred between the web server and the mobile client [4]. As the overhead that is involved when processing XML is a problem for mobile devices one possibility to make the access of information more efficient for mobile devices is to use techniques that reduce the overhead of XML. Johnsrud et al. [5] analysed the efficiency of using different XML compression techniques. The first compression technique is known as generic compression and is characterised by the fact that compressed XML messages have to be decompressed in order to be processed by the application. The second compression technique that was considered is XML-aware compression, where the information structure and hierarchy of the XML document is preserved after compression. The performed analysis shows that the best compression rate was achieved when the use of compression was dependent on the message size [5]. However having a wireless mobile device constantly changing compression rate will result in an increased overhead and the processing on the mobile client.

Another approach to deal with the limitations of mobile devices and the involved problems when using SOAP based web services was proposed by Schmutzler et al. [6]. Their performance analysis of SOAP-XML, WAP Binary XML based on SOAP and JXTA used measurements of the data traffic, transmission delay and error rate (failed requests). The results show that the use of XML-based SOAP web services is very inefficient in terms of data traffic and transmission delay. In contrast, the use of WAP Binary XML based on SOAP results in smaller messages that are transmitted over the network which in turn gives a lower transmission delay. JXTA is more efficient in terms of transmission delay if messages with larger message size are sent. However, fixed sized messages are not guarantee in wireless mobile network.

The research of Lai et al. [7] focuses on SOAP web services and the analysis of the use of the different SOAP bindings HTTP, TCP and UDP. The results showed that the use of the SOAP bindings HTTP and TCP resulted in higher overhead as well as less throughput compared to UDP. However, the provided features therefore depend on the used platform. This often leads to compatibility problems when porting desktop applications to mobile devices.

In this paper we make an evaluation of the overall performance for different ways of accessing a web-based service from a smartphone. The evaluation is based on a real-world PT application — an area, which we believe is a good representative for smartphone usage. To the best of our knowledge there is no similar study made in the literature.

III. Implementation

For the comparison an existing SOAP web service which provides PT information for the city of Hamburg has been used. In order to enable a comparison with REST style web services a new REST web service (called GTI web service) has been implemented which provides the same functionality as the existing SOAP web service. Thereby three existing features, namely init (get general PT information), checkname (request existing start/destination points to search for a route), getroute (request an optimised PT route from a start to a destination point) were used. Both web services are accessed by using HTTP. There are several vendors for smartphones on the market. For practical reason, we have limited our experiments to one of them, Apple's iPhone. Therefore a native iPhone application has been implemented that accessed both web services. As the data that is used in the HTTP request mainly affects how much information is returned from the web service response it is important what kind of data is used in the HTTP request. In our implementation we define two types of messages, one simple and one detailed:

- simple message
  - simple request: A request that provides the web service method with information and expects a simple response.
  - simple response: A reply to a simple request that contains the minimum of results from a specific web service method
    - init: general information, no optional properties
    - checkname: one start/destination point
    - getroute: one optimal route - route contains one trip part
- detailed message
  - detailed request: A request that provides the web service method with information and expects a detailed response.
  - detailed response: A reply to a detailed request that contains the maximum of results from a specific web service method
    - init: general information, a number of additional properties
    - checkname: all start/destination points that match the requested information
    - getroute: one optimal route - route contains several trip part

The use of these two message types provides the possibility to determine the range of delay times that an end-user will have to expect. Table I exemplifies the simple and detailed message type for the init web service method.

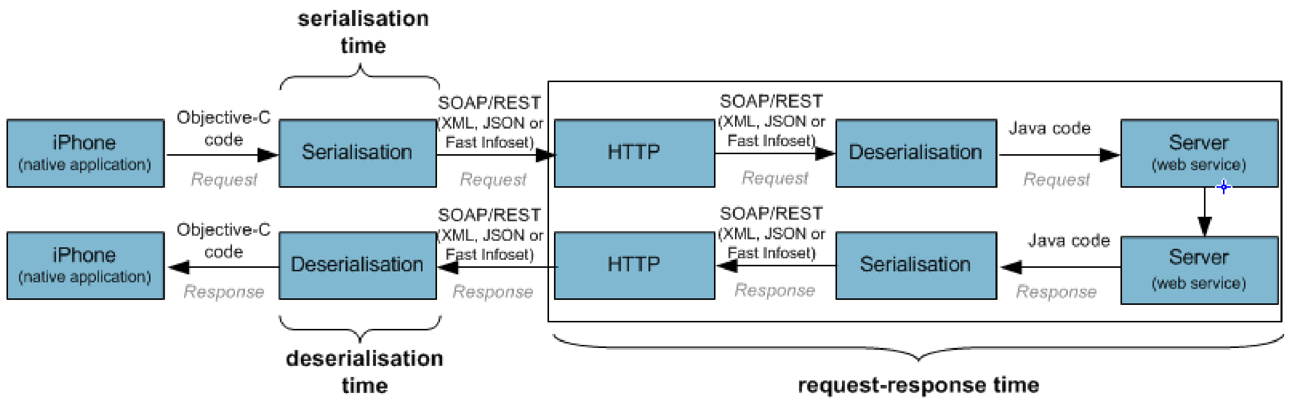

Fig. 2: *Serialisation and deserialisation process*

TABLE I: *Message types - init web service method*

Simple Request	Simple Response
`<gti:InitRequest />`	`<gti:InitResponse>` `<returnCode>OK</returnCode>` `<beginOfService>29.07.2010</beginOfService>` `<endOfService>12.12.2010</endOfService>` `<id>03.22.41.01.01</id>` `<dataId>22.49.02</dataId>` `<buildDate>30.07.2010</buildDate>` `<buildTime>15:08:56</buildTime>` `<buildText>Jahresfahrplan 2010</buildText>` `</gti:InitResponse>`
Detailed Request	**Detailed Response**
`<gti:InitRequest >` `<properties>` `<key>Schedule</key>` `</properties>` `</gti:InitRequest>`	`<gti:InitResponse>` `<returnCode>OK</returnCode>` `<beginOfService>29.07.2010</beginOfService>` `<endOfService>12.12.2010</endOfService>` `<id>03.22.41.01.01</id>` `<dataId>22.49.02</dataId>` `<buildDate>30.07.2010</buildDate>` `<buildTime>15:08:56</buildTime>` `<buildText>Jahresfahrplan 2010</buildText>` `<properties>` `<key>Schedule.color.deftext$EN_GB</key>` `<value>other validities</value>` `</properties>` `<properties>` `<key>Schedule.validityItem0</key>` `<value>Fahrplan ab 27.07.2010</value>` `</properties>` . . . `</gti:InitResponse>`

In order to measure the performance indicators it was necessary to place measurement points. All these were placed within the iPhone application to avoid synchronisation errors between different platforms. Based on the performance indicators the iPhone application was used to measure:

- transferred data volume
  - content length of HTTP request
  - content length of HTTP response
- elapsed time during
  - serialisation process
  - deserialisation process
  - the execution of the HTTP request and the reception of the HTTP response

In this context the transferred data volume is defined as the number of bytes of the message that is placed in the HTTP body of the request and response respectively, using UTF-8 encoding.

The calculation of the elapsed time is performed using `NSDate` to represent a single point in time and is immutable. An object that represents the current date and time was created using the method `date`. For the determination of the elapsed time the instance method `timeIntervalSinceNow` is used which returns the time interval between the calling of the `NSDate` object and the current date and time. The return type is `NSTimeInterval` which specifies a time interval in seconds [1]. The return type `NSTimeInterval` refers to a double precision, floating-point value and yields sub-millisecond precision. The floating-point value enables a calculation of the time difference in milliseconds.

For computing the serialisation/deserialisation time measurement points were placed right before and after the invocation of the respective method from the third party library.

Figure 2 illustrates the entire serialisation and deserialisation process that is performed when a web service method is invoked by the iPhone test application.

The results were stored in an appropriate format after the measurements were conducted on the iPhone in order to perform further analysis. For this purpose a separate performance test web service was created that receives the measurement results that were gathered using the iPhone application. Figure 3 illustrates the process of storing the gathered measurement results.

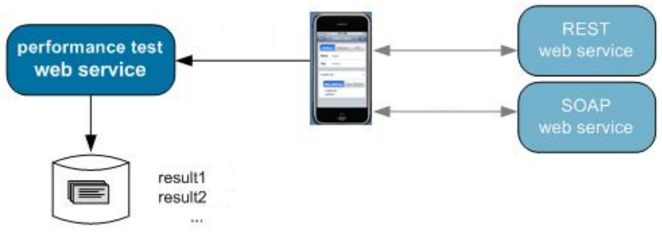

Fig. 3: *Storing of measurement results of performance test*

IV. Results

This paper presents and illustrates the measurement results of the performance indicators

A) transferred data volume [bytes]
B) serialisation/deserialisation time [ms]
C) request–response time [ms]

The result table of the performance indicator transferred data volume (Table II) shows the message size of the different message types (simple/detailed) that were sent and received from the different web service methods (init, checkname and getroute) during the performance test. For the performance indicators serialisation/deserialisation time and request–response time show the mean value and the standard deviation, that were calculated from a sample size of 30 measurements are presented.

The gathered results enable a comparison of the SOAP web service against a REST web service. Additionally it is also possible to compare the different data exchange formats that have been used. The corresponding results of the existing SOAP web services are included in the overview tables (Tables II, IV, V, III).

A. Transferred data volume

Table II illustrates the data volume that is transferred when the web service methods of the REST web service using the data format JSON, XML, Fast Infoset and the SOAP web services are accessed using the simple and detailed message type. As there are no libraries for the serialisation/deserialisation of Fast Infoset messages available for the iPhone the data volume of Fast Infoset was determined by converting the XML message, that was transmitted to the web service, to a Fast Infoset message. The table shows that the used data format (XML, JSON or Fast Infoset) has a significant impact on the number of bytes that are transferred.

TABLE II: *Results: transferred data volume [bytes]*

		transferred bytes [bytes]			
		simple message type		detailed message type	
		request	response	request	response
inits	REST JSON	37	55	235	758
	REST XML	78	139	434	1118
	REST Fast Infoset	76	104	278	731
	SOAP	816	820	14047	15371
checkname	REST JSON	141	135	206	1999
	REST XML	295	289	375	3063
	REST Fast Infoset	221	215	244	1276
	SOAP	2579	2573	1388	9856
getroute	REST JSON	475	463	1088	8240
	REST XML	645	636	1449	10569
	REST Fast Infoset	410	417	771	2954
	SOAP	5226	5225	41763	154155

When the XML data format is used (in the case of REST XML and SOAP) then the highest amount of bytes are transferred. This is the case for the requests and responses of both simple and detailed messages. It can be observed that the use of the data format JSON or Fast Infoset leads to a reduction in the number of bytes that are transferred. While the use of JSON results in the smallest number of bytes for simple messages, Fast Infoset shows the smallest number of bytes for detailed messages.

The SOAP web service uses XML for the data format — therefore are the results of the SOAP web service compared with the web service when using XML. When comparing the new REST XML web service to the old SOAP based web service a reduction of the amount of data of at least one third can be observed.

B. Request–response time

Table III illustrates the average–standard deviation (request–response) time when the web service methods of the REST and the SOAP web service are accessed using different network connections (2G and 3G) on the iPhone. The values of the request–response time when using a 3G network connection are lower than using a 2G networks, because the 3G network connection provides higher data rates.

TABLE III: *Results request–response time [ms] when using iPhone network connection*

		request-response time [ms]			
		simple message type		detailed message type	
		average	standard deviation	average	standard deviation
inits	REST JSON	344.83	49.47	471.86	122.07
	REST XML	315.69	37.55	426.96	94.25
	SOAP	575.93	110.86	529.98	56.44
checkname	REST JSON	326.75	51.33	549.92	61.99
	REST XML	344.93	79.55	662.89	90.29
	SOAP	958.66	96.65	943.55	65.58
getroute	REST JSON	355.43	63.32	3134.09	420.82
	REST XML	368.57	41.39	3048.43	465.75
	SOAP	1870.33	277.38	5559.70	408.21

C. Serialisation/Deserialisation time

Table IV illustrates the elapsed time during the serialisation process that was performed in order to create the HTTP request to access the SOAP web service and the REST web service methods of the using the data formats JSON, XML. As the request messages of simple and detailed type do not differ significantly in the message size (see Table II, checkname) the measured serialisation times of simple and detailed request messages are consequently in a similar range. It can be seen in Table IV that the serialisation of XML messages requires less time than the serialisation of the same message represented in JSON (e.g. when comparing JSON and XML). Furthermore message sent to REST web service methods (JSON and XML) are faster serialized than messages with SOAP.

Table V illustrates the time that is required for the deserialisation of the HTTP response that is received from the web service methods when using SOAP and REST web service. The table shows that the deserialisation of XML messages requires significantly more time than that of the same message represented in JSON (e.g. when comparing REST JSON and REST XML). The deserialisation of the messages that are received from the web service methods of the SOAP web services requires much more time than the deserialisation

TABLE IV: *Results: serialisation time [ms]*

		serialisation time [ms]			
		simple message type		detailed message type	
		average	standard deviation	average	standard deviation
inits	REST JSON	1.51	0.02	2.23	0.04
	REST XML	0.64	0.02	0.94	0.02
	SOAP	3.34	0.06	3.34	0.04
checkname	REST JSON	4.24	0.08	4.24	0.08
	REST XML	1.97	0.02	2.00	0.01
	SOAP	10.13	0.04	10.12	0.07
getroute	REST JSON	11.20	0.09	11.19	0.07
	REST XML	4.51	0.08	4.53	0.06
	SOAP	15.96	0.10	15.91	0.11

of messages that are received from the REST web service methods using JSON and XML.

TABLE V: *Results: deserialisation time [ms]*

		deserialisation time [ms]			
		simple message type		detailed message type	
		average	standard deviation	average	standard deviation
inits	REST JSON	2.02	0.03	3.87	0.10
	REST XML	2.89	0.04	4.45	0.05
	SOAP	7.42	0.16	9.50	0.20
checkname	REST JSON	2.55	0.04	11.36	0.41
	REST XML	6.37	0.13	47.12	1.19
	SOAP	4.18	0.06	37.52	0.46
getroute	REST JSON	5.93	0.13	40.68	0.59
	REST XML	16.17	0.48	101.50	2.23
	SOAP	29.59	3.39	93.81	9.581

Comparing the REST web services when XML is used as the data format with the SOAP web service, it can be seen that the use of the REST web services leads to a reduction in serialisation time of about 80%. The deserialisation time when using the JSON and XML web service instead of SOAP is increased by at least 20%.

V. Further Discussion

A comparison of the different data formats of JSON, XML and Fast Infoset shows that the use of the XML data format results in the highest number of transferred data volume (see Table II). This is due to the fact that XML is characterised by its rich verbosity, meaning that it requires a separate start-tag and end-tag for describing the content. As JSON does not use end-tags at all for the description of the content, the resulting number of bytes is smaller. Fast Infoset shows a similar number of transferred bytes as does JSON for simple messages. As for the JSON, the explanation lies in the elimination of end-tags, which in Fast Infoset replaced by index tables which store all elements that occur within the document. This approach is inefficient for messages that mostly consist of unique elements but beneficial if a reproduction of existing content can be avoided by repeating given indices. In the considered application, the request messages usually contain only a small amount of data, and hence the use of Fast Infoset generally does not reduce the number of transferred bytes. The only messages that contain enough redundant textual content for a significant reduction of transferred bytes are the detailed response messages of the getroute web service method (see Table II getroute), which comprises of several trip parts that use the same redundant start- and end-tags. For request messages and simple response messages the use of JSON results in the lowest number of transferred bytes. It can be concluded that the use of the XML data format is not efficient in terms of transferred data volumes.

Beside the used data format, the use of the JSON, XML and Fast Infoset web services leads to a significant overall reduction in the number of transferred data volume in comparison to the use of the existing SOAP web service (see Table II checkname and getroute) which is not at all designed for smartphones. The reason is that in the considered application the use of SOAP introduces a constant overhead of 261 bytes needed for the SOAP Envelope and SOAP body elements. An additional source of overhead occurs due to the fact that the information sent from the SOAP web service methods is not adjusted to the requirements of smartphone clients. This means that, for instance, the getroute web service method of the SOAP web service returns three departure times for each route, although it is not possible to display this information on one single screen on the smartphone.

As the test application has been developed for iPhone it is important to mention that the measurement results of the serialisation and deserialisation process shown in Table IV and V depend on the used iPhone libraries, namely TouchJSON and KissXML. Therefore it is not possible to draw conclusions on the serialisation/deserialisation time of other smartphones.

The results of the performance test show that the serialisation process of XML is faster than that of JSON for all tested web service methods and message types. The serialisation into an XML message can, unlike JSON, be directly performed from the Objective-C object, which explains the fast serialisation results for XML, given in Table IV. The much longer serialisation time needed for the SOAP web service is due to its design and overhead. Also for the deserialisation the REST web service methods are faster than the SOAP web service, but the differences are smaller. The structure of the SOAP web service with a demanding overhead makes it interesting only for the most detailed message types. These are, however, not very interesting for a smartphone application, since the amount of information that can be shown at a time is limited by the screen size.

TABLE VI: *Summary performance test: best performing web service/data exchange format*

		simple message type		detailed message type	
		request	response	request	response
transferred data volume	init	JSON	Fast Infoset	JSON	Fast Infoset
	checkname	JSON	JSON	JSON	JSON
	getroute	Fast Infoset	Fast Infoset	Fast Infoset	Fast Infoset
serialisation time[1]	init	XML	-	XML	-
	checkname	XML	-	XML	-
	getroute	XML	-	XML	-
deserialisation time[1]	init	-	JSON	-	JSON
	checkname	-	JSON	-	JSON
	getroute	-	JSON	-	JSON
request-response time[1] 3G network connection	init	XML		JSON/XML	
	checkname	JSON/XML		JSON	
	getroute	JSON		JSON/XML	

[1]Fast Infoset was not considered due to lack of Fast Infoset support on the iPhone

VI. Conclusion

Table VI summarises the results of the performance test and contains the web service and used data exchange format that show the best results for a specific performance indicator. The best results for every single performance indicator were achieved by using the new implemented REST web service. The data exchange format (JSON, XML or Fast Infoset) that shows the best results differs depending on the performance indicators and message type. In some cases it was not possible to determine significant differences between the use of JSON and XML.

The overall results of the performance tests have proven that the use of the REST web service leads to an improvement in terms of transferred data, serialisation/deserialisation time and request–response time (in contrast) to the old SOAP web service. Which data format is the most efficient depends on the mobile platform for which the native client application is developed. As the web service is not tailor-made to a specific mobile platform it provides the support of JSON, XML and Fast Infoset. For the development of native iPhone applications the use of the data exchange format JSON is recommended. Although it was not possible to measure the request–response time and processing time of Fast Infoset messages the comparison of the transferred data volumes has shown that the use of Fast Infoset is quite promising for the future when more libraries for Fast Infoset are available.

The tests in this paper are limited to one single application. We believe that this application is a typical example for a public transport information system, which is an area of growing interest for smartphone applications that can enable travel information en route. To strengthen the conclusions, however, we would welcome more tests, also from other application areas.

Acknowledgment

This was part of Katrin Hameseder's Master's thesis, ("Providing access to a Public Transport Schedule system for Smart Phone Clients"), LiU-ITN-TEK–10/054, Dept. of Science an Technology, Linköping University, Norrköping, Sweden (2010), which was carried out in co-operation with Hamburger Berater Team in Hamburg, Germany. Scott Fowler was partially supported by the Swedish Excellence Center at Linköping – Lund in information Technology (ELLIIT).

References

[1] Apple Inc., *iOS Reference Library* [Online], Foundation Data Types Reference, Available: `http://developer.apple.com/mac/library/documentation/Cocoa/Reference/Foundation/Miscellaneous/Foundation_DataTypes/Reference/reference.html`, Accessed: 04. August 2010

[2] Canalys Research, *Gartner Says Worldwide Mobile Phone Sales Grew 35 Percent in Third Quarter 2010; Smartphone Sales Increased 96 Percent*, [Online] http://www.gartner.com/it/page.jsp?id=1466313.

[3] Cisco, *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2009-2014*, [Online] http://www.cisco.com/

[4] M. Adacal and A.B. Bener, *Mobile Web services: a new agent-based framework,* IEEE Internet Computing, 2006, vol.10, no.3

[5] L. Johnsrud, D. Hadzic, T. Hafsoe, F.T. Johnsen and K. Lund, *Efficient Web Services in Mobile Networks* IEEE European Conference Web Services, 2008. (ECOWS)

[6] J. Schmutzler, A. Wolff and C. Wietfeld, *Comparative Performance Evaluation of Web Services and JXTA for Embedded Environmental Monitoring Systems,* IEEE Workshops Enterprise Distributed Object Computing Conference Workshops, 2008 (EDOCW)

[7] K. Y. Lai, T. K. Phan and Z. Tari. *Efficient SOAP Binding for Mobile Web Services*, IEEE Conference on Local Computer Networks, 2005 (LCN)