Ronny Hans, Manuel Zahn, Ulrich Lampe, Apostolos Papageorgiou, Ralf Steinmetz: **Energy-efficient Web Service Invocation on Mobile Devices: The Influence of Compression and Parsing**. In: Proceedings of the 2nd International Conference on Mobile Services (MS 2013), p. 1-6, Institute of Electrical and Electronics Engineers (IEEE), June 2013. ISBN 978-0-7685-5029-9.

Energy-efficient Web Service Invocation on Mobile Devices: The Influence of Compression and Parsing

Ronny Hans, Manuel Zahn, Ulrich Lampe, Ralf Steinmetz
Multimedia Communications Lab (KOM)
Technische Universität Darmstadt
Darmstadt, Germany
Email: {firstName.lastName}@KOM.tu-darmstadt.de

Apostolos Papageorgiou
Network Research Division
NEC Laboratories Europe
Heidelberg, Germany
Email: apostolos.papageorgiou@neclab.eu

***Abstract*—In recent years, there has been a rapid growth in the number of smartphone applications, many of which rely on Web services as key building blocks. Unfortunately, the use of such applications and services requires substantial amounts of energy, which is specifically problematic in the context of battery-constrained mobile devices. In this paper, we examine the potential for energy-efficient mobile service consumption through fine-grained experiments. Our results indicate that energy savings of up to 21.5% may be achieved through the sophisticated use of compression, while the choice of an appropriate parsing strategy may yield savings of up to 53.4%. The results of our work facilitate the development of more energy-efficient, service-based mobile applications.**

***Keywords*-Mobile applications; Service orientation; Web services; Energy consumption; Energy savings**

I. Introduction

In 2012, the global smartphone shipments reached 700 million units, which marks an increase of about 43% compared to the previous year [1]. According to the research firm DisplaySearch, sales are expected to exceed one billion units in 2016 [2]. In line with this development, hundred of thousands of new applications are being developed for the different cellular platforms. For example, the number of applications in the *Google Play Store*[1] corresponded to more than 800,000 in early 2013. As a consequence, virtually any desired functionality can be found in the form of an app today.

An essential part of distributed applications are *Web services*, which constitute a popular implementation of Service-Oriented Architectures (SOAs). Web services are software artifacts designed for machine-to-machine communication, which interact via a network [3]. They provide a wide range of functionality and can be composed to more complex applications. Web services are accessed by different clients using open and standardized interfaces [4].

The basis for more and more complex software applications is capable hardware, which, as well, developed rapidly in recent years. Current smartphone models feature fast Central Processing Units (CPUs), variable communication interfaces, like Wireless Local Area Network (WLAN), Third-Generation Cellular Network (3G), or Long Term Evolution (LTE), and multiple sensors, such as Global Position System (GPS) or acceleration detectors [5].

Unfortunately, the development of batteries cannot keep up with these rapid advancements in smartphone technology. Based on technical progress, the battery capacity, i. e., energy supply, increases by about 4% per year. However, the average useable time between two charges of a cellphone is predicted to reduce by 4.8% annually because of increasing utilization [1]. Hence, batteries can be seen as the major bottleneck in mobile devices, and due to the mobility aspect of smartphones, it is almost impossible to increase the battery in size.

In case of Web services, which are accessed via Internet, wireless interfaces are required. However, frequent use of these interfaces causes a major part of energy consumption within smartphones [6], [7]. Compared to the standby time, the battery lifetime is only a fraction if such interfaces are used. For example, an Apple *iPhone 5* provides up to 225 hours of standby battery life, but only 8 hours with permanent 3G or LTE usage [5].

In this work, we analyze the influence of compression and parsing mechanisms on energy consumption when invoking Web services from a mobile device. For that purpose, we conduct empirical measurements using common WLAN and 3G networks, based on the prototypical implementation of a mobile Web service client.

The remainder of this paper is structured as follows: We describe the methodology and the implementation of our experiments in Section II. In Section III, we provide the results of our experiments and a detailed discussion of practical implications. Afterwards, in Section IV, we give an overview of related work. Finally, we conclude with a summary and outlook in Section V.

II. Experimental Methodology and Implementation

In the following, we describe the constraints, methodology, and implementation of our experiments to ensure complete

[1] http://play.google.com/

traceability. As outlined in the introduction, we focus on two different optimization methods to reduce the energy consumption for Web service invocation, namely, different compression and parsing mechanisms.

For the measurements of energy consumption, we use a Symbian-based *Nokia E71* smartphone. To monitor the energy consumption, we apply the *Nokia Energy Profiler* in version 1.2[2]. At first glance, the Nokia E71 may appear a bit old-fashioned in comparison to contemporary smartphones. However, due to the fact that, e. g., modern Android smartphones do not come with reliable built-in power meters, we decided to use the Nokia smartphone.

To enable comparability of each single measurement, the experiments were performed under comparable circumstances and procedures. This includes identical phone conditions, such as no unnecessary background applications and no superfluous activated components. Furthermore, we ensured identical network conditions, such as the same SIM card and WLAN access point, and conduct all measurements within a defined time frame and at the same location. By waiting a specified normalization period, potential inactivity timers were considered in performing the test invocations. In order to account for random outliers, each measurement was repeated three times, and the individual results were subsequently averaged.

For the experiments, we developed a Java 2 Micro Edition (J2ME) based test program for the Symbian platform. It is capable to invoke different Web services and to present the results on the smartphone display. The Web service *Global Weather*[3] from WebserviceX.NET[4] has been chosen as test object. Thereby, we used the method *GetCitiesByCountry* to retrieve all French cities. Including protocol overhead, the response has a size of 16 KB.

A. Web Service Invocations with Compressed Messages

The amount of data which is transferred between the mobile device and a Web service determines the timespan that a wireless interface remains in an active and high energy state. The higher the amount of data, the higher the energy consumption. By using compression techniques, the amount of data and thus the energy consumption could be reduced. Furthermore, with a lower amount of data, less data packages for data transmission are required, which results in a minor package loss [8] and consequently, a lower ratio of package retransmissions.

In return, more computation power is necessary, which results in additional energy consumption. Due to the fact that mostly text-based Extensible Markup Language (XML) files are used for Web services, compression could play an important role in reducing the amount of data. According to Johnsrud et al., two different compression techniques could be applied for Web services invocations: Generic compression and XML-aware compression [9].

The objective of generic compression mechanisms is the reduction of arbitrary documents. Prior to processing, such documents have to be decompressed first. XML-aware compression methods were specifically developed for XML documents. There are methods available that preserve the structure and hierarchy of the XML information. Instead of a human-readable text format, such methods usually use a machine-readable format to store the information [9].

In our experiments, solely the SOAP body payload of the server response message was compressed. This approach has the advantage of the SOAP header remaining uncompressed and processable, as suggested by Johnsrud et al. [9] and Tian et al. [10]. Doing so, proxy servers or other intermediate servers, e. g., for load-balancing or authorization, are capable to handle such SOAP messages without adding additional functionality. Besides, the payload of the server response is usually responsible for most of the traffic, whereas the volume of the header is negligible.

For these experiments that focused on compression mechanisms, we extended our initial test program with a decompression and compression functionality. Three compression algorithms have been implemented: *Range Coding* (RC) via a free Java implementation[5], WAP Binary XML (WBXML) via kXML[6], and ZIP via jazzlib[7]. For ZIP compression, we used the highest ZIP compression level.

The compression ratios of the different algorithms are shown in figure 1. RC achieves the worst results with an average compression ratio of 59.7% (i. e., a reduction in size of 40.3%), WBXML performs slightly better with 47.5%. The ZIP compression performed best with 12.6%.

Usually, Web service hosts generally do *not* support compression. Hence, we simulated the traffic using comparable invocations. The payloads of these invocations are similar to the compressed payloads. For example we simulated RC compression of French cites (which gives a message size of $5,461$ Bytes) by requesting Mexican cities (with a message size $5,412$ Byte). Subsequently, the actual server response was discarded and instead, a hard-coded, compressed response was processed, which had been stored on the smartphone.

B. Web Service Invocations with Different XML Parsers

Information exchange between Web services is normally based on XML messages, more precisely on SOAP. Such messages require parsing, which results in high computational effort [11]. As a consequence, during Web service requests,

[2]http://www.developer.nokia.com/Resources/Tools_and_downloads/Other/Nokia_Energy_Profiler/

[3]http://www.webservicex.net/ws/WSDetails.aspx?CATID=12&WSID=56/

[4]http://www.webservicex.net

[5]http://www.winterwell.com/software/compressor.php

[6]http://kxml.sourceforge.net/

[7]http://jazzlib.sourceforge.net

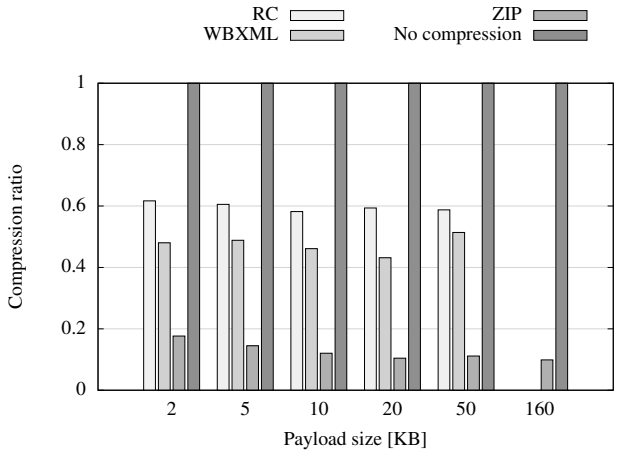

Figure 1. Message compression ratios, by payload size and compression mechanism.

the CPU has an important influence on the energy consumption. Further, during the processing of SOAP messages, the whole Web service invocation is delayed, which results in a longer active state of wireless interfaces [10], and hence, also increased battery drain. Thus, the performance of the used parser influences the CPU load and therefore, the energy consumption.

Ray and McIntosh [12] distinguish between two parsing *strategies*: *Tree-based* and *stream-based* parsing. When following a tree-based strategy, the parser builds the entire document and stores it in memory. Only after finishing this process, access to the actual information is possible. In contrast, using a stream-based strategy, the parser directly allows callbacks to the programming code during the processing of XML documents.

According to Knudsen, three principal *types* of parsers can be differentiated [13]: A *model parser* parses a document completely and stores a binary representation in memory. Only after the process is finished, the information can be accessed. Hence, model parsers follow a tree-based parsing strategy. A *push parser* also parses a document completely, but notifies a listener within the underlying application when it encounters specific parts of the document. Lastly, a *pull parser* processes just a small portion of a document controlled by an application, which requests the next parts in a piecewise manner. Both push and pull parses pursue a stream-based strategy.

In our experiments, four different parsers have been used: kXML (push parser), a modified version of tinyXML[8] and nanoXML[9] (both model parsers), and ASXMLP[10] (pull parser). The parsing measurements were performed with deactivated wireless interfaces. Each parsing measurement consisted of ten consecutive parsing processes of a 10 KB SOAP message.

Furthermore, we analyzed the influence of the element depth within a SOAP document regarding the energy consumption. Thereby, we took a file of constant size and varied the depth of elements from one to three.

III. Experimental Results

In this section, we present the results of the energy measurements for the previously described experimental settings.

A. Web Service Invocations with Compressed Messages

Initially, we ran a test series of ten continuous Web service invocations with compressed messages. We expected a decrease in transfer energy due to smaller message sizes. In return, we expected an increase in computation energy due to decompression of the compressed messages. This increase should be quantitatively lower than the decrease, hence yielding energy savings. A continuous invocation should enhance this effect.

However, the UMTS results refuted our assumption. The compression algorithms caused an increase in energy consumption of about 3.1% for WBXML, 4.9% for RC, and even 22.1% for ZIP. The results show that the more the decompression algorithm utilizes the CPU, the more energy is required.

An in-depth examination of individual measurements with and without compression explains the higher energy consumption. In the light of the longer processing time at decompression, the next invocation will be delayed for exactly this time span. Therefore, the wireless interface remains longer in high power mode. With continuous invocations, these delay time spans will be accumulated. The inactivity timers will be activated later and the energy consumption increases due to longer energy intensive states. Exemplary, this is shown in Figure 2 for a measurement without compression and with ZIP compression.

The second test series focused on non-recurring invocation. Thereby, we determine the payload size at which a single Web service invocation with compression is energetically reasonable. With a single invocation the decompression time should be irrelevant regarding the wireless interface, because the inactivity timers should be triggered immediately after the data transfer. Figure 3 shows a comprehensive picture of the UMTS energy consumption measurements.

Up to a payload size of 20 KB, the energy consumption stays at 44 Joules with and without compression. At 50 KB payload size, the energy saving with compression is substantial. While the compression algorithms remain at 44 Joules, the energy amount without compression increases to approximately 52 Joules. The tendency shows that the compression algorithm with the best compression ratio achieves the lowest energy consumption. With a payload

[8]http://sourceforge.net/projects/txml
[9]http://devkix.com/nanoxml.php
[10]http://asxmlp.sourceforge.net

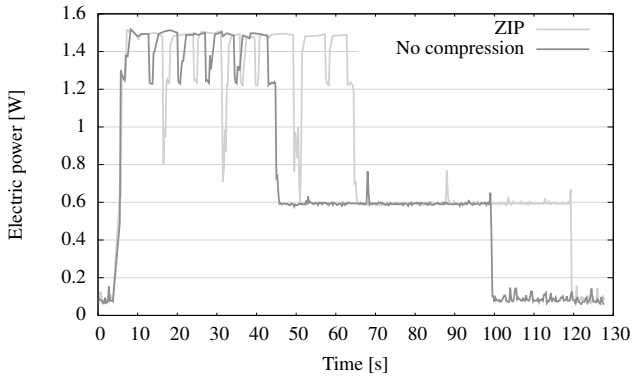

Figure 2. Electric power over time using *UMTS*, by compression mechanism.

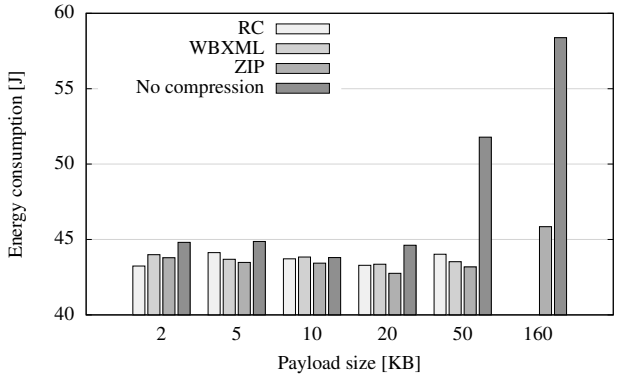

Figure 3. Average energy consumption using *UMTS*, by payload size and compression mechanism.

size of 50 KB and 160 KB, energy savings in the orders of 16.6% and 21.5% can be realized, respectively. In absolute terms, this corresponds to 8.6 Joules and 12.5 Joules.

At payload sizes from 10 KB to 160 KB, the test series with WLAN shows no notable differences between compression and no compression, as shown in Figure 4. The energy consumption exhibits a minor upward tendency with growing payload size. These findings confirm previous results: Once the WLAN connection is established, the data transfer is relatively energy-efficient [14], [15].

B. Web Service Invocations with Different XML Parsers

The energy consumption measurements with different parsers were performed with a tenfold repetition of a parsing process. The parsers differ from each other: The higher the processing time, the more energy is needed. Table I outlines this correlation. If a parsing process is triggered, the CPU is working at full capacity.

If we take a closer look at the underlying parser implementation concepts, the push parser (kXML) and pull parser (ASXMLP) have notably lower energy consumption

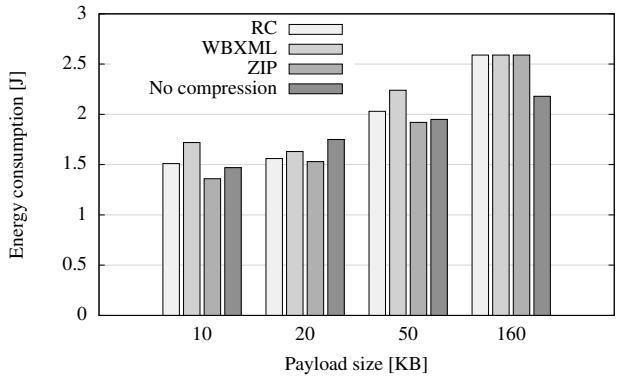

Figure 4. Average energy consumption using *WLAN* in conjunction with small intervals, by payload size and compression mechanism.

Table I
AVERAGE PROCESSING TIME AND ENERGY CONSUMPTION, BY PARSER.

Parser	Processing time [s]	Energy consumption [J]
kXML	93.5	58.29
tinyXML	197.5	122.90
nanoXML	106.25	66.09
ASXMLP	92.50	57.18

than model parsers (nanoXML and tinyXML). While kXML and ASXMLP exhibit almost the same energy consumption, 15% respectively 115% additional energy is needed for nanoXML and tinyXML. Vice versa, using ASXMLP instead of tinyXML yields energy savings of 53.4%. tinyXML is designed for simplicity and small size, rather than for speed of processing[11]. This could be an explanation for the high energy demand. Generally, the evaluation indicates that stream-based parsers are to be preferred over model-based parsers.

Additionally, we conducted measurements depending on the number of elements per item (i. e., city in the result list) within the XML file. Like before, we measure the energy consumption depending on different parsers and with a tenfold repetition of the parsing process. In this context, we varied the number elements per city from one to three. To keep the SOAP file at a constant size of 10 KB, we injected additional characters for each item. On the one hand, the results confirm the previous results regarding the efficiency of the parsers. From two elements onwards, we receive the same order as before, namely ASXMLP in lead, with kXML, naoXML, and tinyXML trailing. On the other hand, the results show that a high nesting depth should be avoided when using XML files.

IV. Related Work

A large body of research exists with a focus on energy consumption and energy savings in mobile devices. However,

[11] http://www.grinninglizard.com/tinyxml

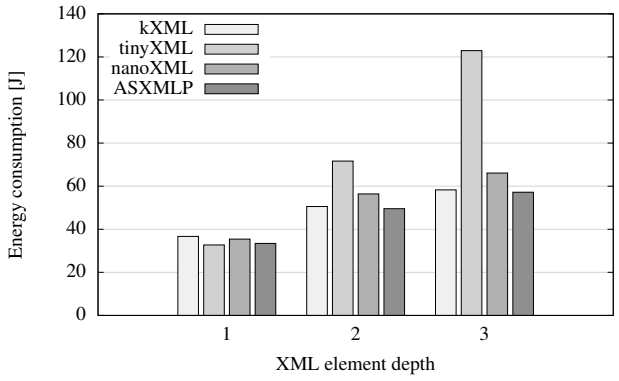

Figure 5. Average energy consumption, by XML element depth and used parser.

only a few papers specifically focus on Web services. With respect to the impact of compression, some works exists, but there focus is on performance aspects, such as latency, rather than energy consumption. To the best of our knowledge, no scientific works explores the influence of different parsers on energy consumption in mobile devices.

Tian et al. [10] analyze how mobile devices can benefit from compression of Web services messages, especially when a poor connection like second-generation General Packet Radio Service (GPRS) is used. They pointed out that Web services transmit four to five times more data than traditional dynamic Web applications do. Therefore, compression could help to reduce the additional traffic. In return, additional CPU time is required for decompression. In their experiments, Tian et al. find that compression reduces the time interval between the initial request and the presentation of the result for poor connections. This holds true for connection types such as GPRS. For better connectivity, such as Bluetooth or WLAN, no significant improvement in response time was found. Because of the negative influence of compression on server performance, the authors propose a dynamic approach, were clients have different option to request compression, and recommend dynamic compression on low bandwidth connections.

Johnsrud et al. [9] focus on compression to reduce of transmitted data and the time it takes from the user interaction until the result is shown on the mobile device. Thereby, they had a closer look to two different compression techniques. The used the generic ZLIB compression method and the XML-specific binary encoding *Efficient XML* (EFX). With these two techniques, they compressed XML payloads up to a size of 24 KB. Further, they analyzed the effects within different wireless technologies, namely UMTS, GPRS and EDGE. Regarding the reduction of transferred information, EFX achieve slightly better results than ZLIB. Still, with both algorithms, the amount of data can be reduced substantially. Thus, if the network usage is charged by number of bytes transferred, costs could be reduced. Regarding the response time, the best relative results with respect to time reduction are achieved using GPRS. In the case of UMTS, the effect of compression is quite small, because the the high bandwidth results in a comparatively small transfer time, compared to a relatively large processing, coding, and encoding time for the SOAP payload.

Werner et al. [16] implement a new parser and compressor for SOAP messages, which replaces XML tags with binary identifiers. For that purpose, they implement a push-down automate. This mechanisms leads to lower memory and CPU utilization. The evaluation results show a possible compression of all XML files, independent of their size and structure, to a size of 1% to 15% percent of the original file.

Barr and Asanovic [17] pointed out that the transmission of a bit via wireless interfaces can require over $1,000$ times more energy than a 32-bit computation. To reduce energy consumption the authors analyzed different families of compression algorithms and recognize that compression may cause an increase of energy consumption when data is compressed before sending. They found out that the choice how and whether compression should be applied depends on hardware aspects, like CPU, memory, and network connection, but also on software factors like compression rate. Because these factors change in new generations of devices and applications, the authors stress that a permanent re-evaluation compression mechanisms is required. Further, Barr and Asanovic explain that compression and decompression of data have different demands regarding energy consumption. Based on this observation, the authors recommend to use asymmetric compression and decompression instead of a single tool for both.

In summary, to the best of knowledge, this work is the first to empirically examine the energy consumption of mobile Web service invocations, based on various compression and parsing mechanisms and under different network conditions. Thus, our work not only provides valuable recommendations to practitioners on optimizing the energy consumption of mobile applications, but also points to future research directions in this area.

V. Conclusion and Outlook

Energy is a scarce resource in mobile devices, and the current technical development indicates that in the foreseeable future, there will not be a substantial improvement in battery capacity. Hence, the knowledge of energy-aware application development is crucial. In this paper, we provided a comprehensive study on optimizing the energy-efficiency in mobile Web service-based applications.

As first contribution, we analyzed the reduction of energy consumption through the use of different compression algorithms. For Web service invocations, compression can reduce the energy consumption. The reduction depends on

used wireless interface, number of consecutive invocations, and message sizes. For single invocations using UMTS, we demonstrated possible energy savings of up to 21.5%, whereas consecutive invocations in UMTS can even lead to an increased consumption of up to 22.1%. However, our results also indicate that compression has no substantial influence on energy consumption if WLAN is used as network connection.

As second contribution, we examined the reduction of energy consumption through the use of different parsers. Certain options in the parsing methodology may yield energy savings of up to 53.4%. Moreover, stream-based parsers are to be preferred over model-based parsers. With respect to elements describing a single value within a XML file, a high nesting depth leads to a longer and more energy-intensive parsing process. For example, for the parser kXML, using one instead of three elements in depth leads to energy savings of 41.7%.

For future work, we can identify a couple of starting points. For example, additional experiments with Web service invocations in combination with compression would be insightful, such as utilization of LTE, variation of the signal-strength, or additional efficient compression methods like Efficient XML Interchange (EXI). Another potential extension exists in the repetition of the test invocations with a more powerful smartphone. In this context, the main objective could be to determine the CPU speed at which continuous Web service invocations with compression are energetically reasonable.

ACKNOWLEDGMENTS

This work has partly been sponsored by E-Finance Lab e. V., Frankfurt a.M., Germany (www.efinancelab.de). The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreements no. 285220 and 318201.

REFERENCES

[1] S. Analytics, "Cellphone Energy Gap is Widening," https://www.strategyanalytics.com/default.aspx?mod=pressreleaseviewer&a0=4656, April 2009.

[2] Displaysearch, "Smartphone Shipments to Pass One Billion in 2016, According to NPD DisplaySearch," http://www.displaysearch.com/cps/rde/xchg/displaysearch/hs.xsl/120912_smartphone_shipments_to_pass_one_billion_in_2016.asp, September 2012.

[3] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web Services Architecture," http://www.w3.org/TR/ws-arch/, February 2004.

[4] I. Melzer, *Service-orientierte Architekturen mit Web Services*. Heidelberg: Spektrum Akademischer Verlag, 2010.

[5] A. Inc., "Apple - iPhone 5 - View all the Technical Specifications," http://www.apple.com/iphone/specs.html, March 2013.

[6] G. P. Perrucci, F. H. Fitzek, G. Sasso, W. Kellerer, and J. Widmer, "On the Impact of 2G and 3G Network Usage for Mobile Phones' Battery Life," in *Proceedings of the 15th European Wireless Conference (EWC 2009)*, 2009.

[7] T. Pering, Y. Agarwal, R. Gupta, and R. Want, "CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces," in *Proceedings of the 4th International Conference on Mobile Systems, Applications, and Services (MobiSys 2006)*, 2006.

[8] N. Apte, K. Deutsch, and R. Jain, "Wireless SOAP: Optimizations for Mobile Wireless Web Services," in *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web (WWW 2005)*, 2005.

[9] L. Johnsrud, D. Hadzic, T. Hafsoe, F. T. Johnsen, and K. Lund, "Efficient Web Services in Mobile Networks," in *Proceedings of the 6th IEEE European Conference on Web Services (ECOWS 2008)*, 2008.

[10] M. Tian, T. Voigt, T. Naumowicz, H. Ritter, and J. Schiller, "Performance Considerations for Mobile Web Services," *Elsevier Computer Communications Journal*, vol. 27, no. 11, pp. 1097–1105, 2004.

[11] J. Sharkey, "Google I/O 2009 Developer Conference: Coding for Battery Life," http://www.google.com/events/io/2009/sessions/CodingLifeBatteryLife.html, May 2009.

[12] E. T. Ray and J. McIntosh, *Perl & XML*. Sebastopol: O'Reilly, 2002.

[13] J. Knudsen, "Parsing XML in J2ME," http://www.oracle.com/technetwork/systems/index-155764.html, March 2002.

[14] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications," in *9th ACM SIGCOMM Conference on internet Measurement Conference (IMC 2009)*, 2009.

[15] A. Rice and S. Hay, "Decomposing Power Measurements for Mobile Devices," in *Proceedings of the 2010 International Conference on Pervasive Computing and Communications (PerCom 2010)*, 2010.

[16] C. Werner, C. Buschmann, Y. Brandt, and S. Fischer, "Compressing SOAP Messages by using Pushdown Automata," in *Proceedings of the 2006 International Conference on Web Services (ICWS 2006)*, 2006.

[17] K. Barr and K. Asanovic, "Energy Aware Lossless Data Compression," in *Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*, 2003.

All online references were last accessed and validated in March 2013.