Web services on embedded systems - A performance study

Christin Groba and Siobhán Clarke
Lero Graduate School of Software Engineering
Distributed Systems Group
School of Computer Science and Statistics
Trinity College Dublin
Dublin, Ireland
Email: grobac, Siobhan.Clarke@scss.tcd.ie

***Abstract*—A platform-independent communication mechanism is essential for the seamless integration of embedded devices into the Web of Things. Web services provide for such communication, though there remains an open question as to whether they are suitable for hardware-constrained devices. While there is a general perception that they are too resource-intensive, there are few scientific studies dedicated to answering this question. As a result, researchers or developers investigating applications for the Web of Things have little evidence to support a decision on the best technical solution. This paper presents the results of a study designed to assess the performance of Web services on embedded devices. For our investigation, we deployed Web services on Sun SPOTs, representing medium-sized wireless sensor platforms, and analysed disk space, message size, response time and energy consumption. Our study quantifies the overhead of Web services and provides empirical data on whether this Web technology is a suitable approach for embedded devices.** [1]

***Keywords*-Web service; SOAP; performance; embedded device; wireless sensor;**

I. Introduction

Embedded devices are installed into real-world objects to provide information about, or control over, their hosts. Traditionally, they are dedicated to a single application and commonly-used custom interfaces. With the emergence of the Web of Things, embedded devices become part of an open environment in which they serve multiple different applications and interact with peer devices that may differ in hardware and software features. Device heterogeneity and application diversity make communication via custom interfaces inflexible and impractical. A platform-independent communication mechanism would improve interoperability among embedded devices and enable their seamless integration into the Web of Things.

Web services are designed for interoperable machine-to-machine communication, allowing different parties to invoke remote methods or to exchange documents without mutual knowledge of internal implementation details. A Web service client communicates with a Web service provider through platform- and programming language-independent messages. SOAP-based Web services[2] encode these messages into a structured and typed data format that is based on XML.

However, a question remains as to whether Web services are suitable for hardware-constrained devices. Web services require additional payload information and use a verbose data encoding format. This leads to an increase in size and processing complexity of Web service messages. Therefore, as embedded devices are constrained in terms of memory, bandwidth, and energy supply, there is the general perception that Web services are too resource-intensive for this domain. However, there are few scientific studies dedicated to investigate this issue for tightly constrained hardware platforms. As a result, researchers or developers exploring new applications for the Web of Things have little evidence to support a decision on the best technical solution.

This paper presents the results of a study designed to assess the performance of Web services on embedded devices. For our investigation, we deployed Web services on Sun SPOTs, representing medium-sized wireless sensor platforms, and analysed disk space, message size, response time and energy consumption. The contribution of this paper is empirical data that quantifies the overhead of Web services and therefore supports a decision on whether this Web technology is a suitable programming approach for embedded devices.

The remainder of this paper is organised as follows: Section II summarises work related to the deployment of Web services on embedded devices. Section III describes the methodology applied to conduct the experiments and

[2]For the remaining of this paper our references to Web services assume SOAP-based Web services

to collect performance data. Section IV presents the result of the study and analyses the effect of Web service on disk space, message size, response time, and energy consumption. Section V discusses the implications of using Web services in a resource-constrained environment and highlights how emerging research addresses them.

II. Related work

Web services are a key enabler for a new generation of applications. Mobile devices, which not only consume but also provide information services, can be directly integrated into computer-supported collaborative work or supply chain management [1]. Web service-enabled sensors and factory equipment share the same communication architecture as applications on the business level and can, therefore, be integrated into the process flow of enterprise planning software [2].

A key question is, how to offer Web services for resource-constrained devices without compromising on standard compliance or limiting the device core functionality. Regarding these design requirements, light-weight tool support has been proposed for Java CDC [3] and CLDC [4] as well as C/C++ [5]. In addition, the device profile for Web services (DPWS) [6] specifies a minimal set of implementation constraints to enable Web service messaging, discovery, and eventing based on existing Web service specifications.

Wireless sensor platforms exhibit even stricter resource-limitations than mobile devices and gateway concepts have been used to integrate them with a Web service-based communication system. Emerging research, however, suggests enhancements for DPWS [7] and optimisations for Web services [8] to apply Web services directly on sensor nodes and to eliminate gateway solutions.

Although a performance evaluation is included in most of the above related work, its intention is to validate the proposed solution itself. Our work is dedicated to assess the performance of Web services on embedded devices using two existing open-source libraries. We provide empirical data on time and energy consumption that characterises Web service behaviour before any optimisations or enhancements are applied.

III. Methodology

Our experiments were conducted in a controlled test environment. We used two free-range Sun SPOTs that are representative for medium-sized wireless sensor platforms (see Table I). The SPOTs communicated over the air in a single hop keeping the network delay to a minimum.

The application scenario involved the exchange of temperature readings between the service provider, hosted on one SPOT, and the service client, hosted on the other SPOT. The service provider offered to send a list of the last n readings upon request. The service client specified the number of readings in his request and invoked the provider through a synchronous remote procedure call. A reading is a complex data type that consists of a value (double), time stamp (long), and sampler id (string).

Table I
Examples of wireless sensor platforms

	TelosB Mote	Sun SPOT	IMote2
Processor (bit/Mhz)	MSP430 16/8	ARM-7 32/180	PXA271 32/13..416
ROM/Flash (kB)	48 + 1024	4096	32768
RAM (kB)	10	512	256 + 32768
Radio (IEEE)	802.15.4	802.15.4	802.15.4
System	TinyOS	Squawk VM	TinyOS, Linux

We implemented the application scenario in three approaches: First, the light-weight conventional approach encoded the list of readings as a semicolon and hash separated string not using Web service technology (Listing 1). Second, the ksoap approach, based on ksoap2 [9], created and exchanged SOAP messages encoding the list of readings in XML (Listing 2). Third, the ws4d approach, based on the DPWS implementation ws4d Java ME [10], used the same XML structure as the ksoap approach to encode the list of readings. In addition to SOAP the ws4d approach included HTTP information into the message. The transfer of SOAP messages is not bound to a specific protocol; however, the advantage of embedding a SOAP message into HTTP is that firewalls usually do not block the required port.

As performance metrics we evaluated disk space, message size, average response time, and average energy consumption. Each average was calculated based on 30 samples and indicated only marginal statistical spread. For energy consumption we measured the available battery capacity with a method provided by the Sun SPOT engine and calculated the difference. In our investigation we focused on the service invocation phase. The time and energy spent on the acquisition of the service description and endpoint address was not included in our measurements.

Listing 1. Conventional format for list of sensor readings

```
value#timestamp#samplerid;..#..#..;
```

Listing 2. XML format for list of sensor readings

```
<readings>
  <reading>
    <value> ... </value>
    <timestamp> ... </timestamp>
    <samplerId> ... </samplerId>
  </reading>
  <reading> ... </reading>
</readings>
```

IV. Results

In the following we present the results of the performance study that show the effect of Web services on the disk space, message size, response time, and energy consumption.

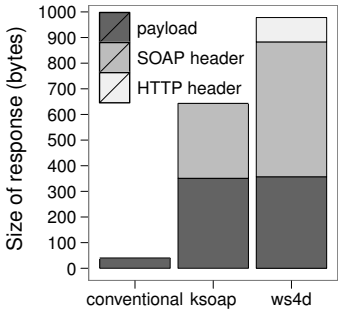
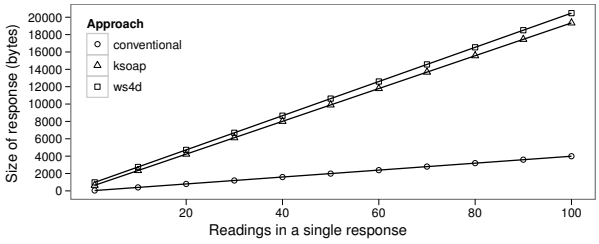
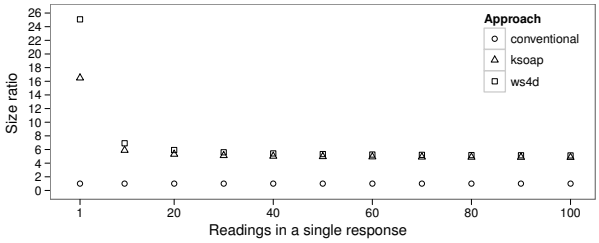


Figure 1. Structure of response message containing one reading



(a)



(b)

Figure 2. Effect on response message size (a) absolute and (b) relative to conventional approach

A. Effect on disk space

Apart from the application code, we used two open-source libraries to enable Web services on the wireless sensor platform directly. Ksoap2 including the kxml2 parser requires 61 Kbytes of disk space. For the ws4d stack 478 Kbytes of disk space are allocated, after unnecessary modules have been removed.

B. Effect on message structure and size

Figure 1 depicts the structure of the response message that contains one reading. The Web service based messages are 25 (ws4d) and 16 (ksoap) times larger than a conventional message. In particular, the actual payload in the Web service based responses is nine times larger than the payload of the conventional approach. This illustrates the overhead that is associated with using a structured versus a custom data format. In addition, the size of the required header information equals (ksoap) or is twice (ws4d) the size of the actual payload in the Web service based response. This shows the overhead of SOAP as an application layer protocol. For transmission over the air, IEEE 802.15.4 allows 120-102 Bytes for packet data depending on how long the address information on MAC layer is. If 120 Bytes per radio packet are available, seven (ksoap) and ten (ws4d) radio packets are required to transmit a single reading while a conventionally encoded reading fits in a single radio packet.

Figure 2a shows the development of the response message size as the number of readings, enclosed in the single response, increases. With each additional reading in the response, the message grows by about 40 (conventional), 189 (ksoap), and 197 (ws4d) bytes. The increase for the ws4d approach is slightly higher than for ksoap because ws4d formats the payload differently.

Figure 2b illustrates the overhead of Web service based messages relatively to conventional messages. The ratio approaches a limit of about 5 for both Web service approaches as the readings included in a single response increase. The decrease of the Web service overhead on the message size has two reasons: First, the internal XML structure of the list of readings plays off its efficiency if multiple readings are enclosed. Second, independent from how many readings a response contains, the header remains constant (apart from occasionally one byte difference in the HTTP header for encoding the content length).

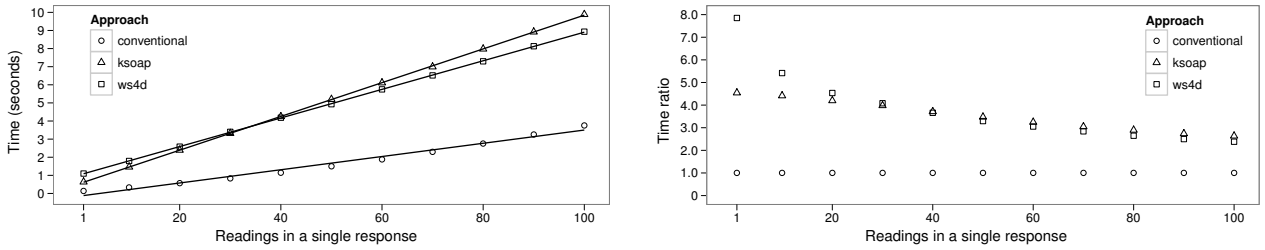
C. Effect on response time

The response time, captured on the client device, is the time from serialising a request to the completion of deserialising its response.

Figure 3a presents the development of the average response time as the number of readings per response increases. The average response time for the ws4d approach does not increase as fast as for the ksoap approach because ws4d starts to process a message with the arrival of the first bytes. The ksoap approach, instead, blocks processing until the entire message is received.

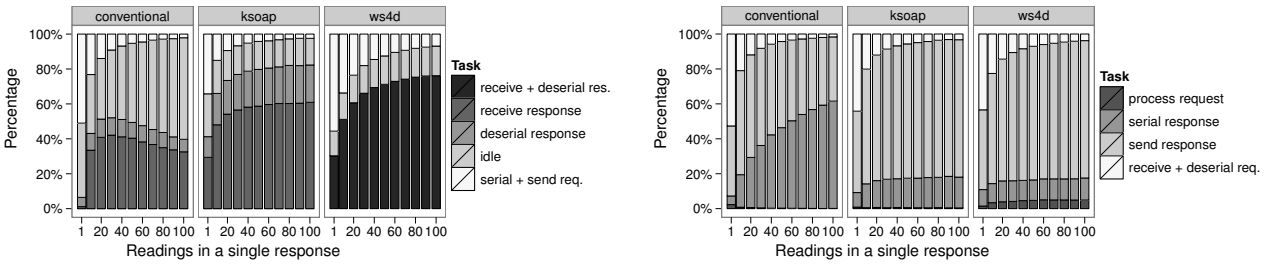
Figure 3b depicts the overhead of Web service on the response time relatively to the conventional approach. The ratio approaches a limit of 2.5 (ksoap/conventional) and 2 (ws4d/conventional) with increasing readings per response.

Regarding the structure of the response time, Figure 4 shows the proportional share, a task has on the average response time. On the client device, in Figure 4a, a substantial part of the time is required for receiving the response. For the ws4d approach, reception and deserialisation interleave and cannot be measured individually. For ws4d we assume that reception takes a greater share than deserialisation. Another interesting observation is the following: In the conventional approach, the share of the receive task decreases in favour of the share of the idle task. In contrast, in the Web service approaches the share of the receive task increases with the increase of readings in a single response. The reason for this reveals the time structure on the service device (Figure 4b). In the conventional approach, the amount of time for



(a) (b)

Figure 3. Effect on average response time (a) absolute and (b) relative to conventional approach



(a) (b)

Figure 4. Time structure on client device (a) and service device (b)

serialisation grows faster than for sending because up to three readings can be transmitted with a single radio packet. While the amount of time for serialising is the same for each reading, the amount of time for sending a reading depends on whether the reading has to be send in a new radio packet or not. The increasing share of serialisation on the server side has a direct effect on the client device whose share for being idle increases. The advantage of this is that sensor platforms can be configured to consume fewer or no resources when being idle. For the Web service approaches, the time for serialisation and transmission grows almost proportionally because no two readings can be transmitted in a single radio packet and the time for sending a reading is the same for all readings.

D. Effect on energy consumption

Figure 5 presents the overhead on energy consumption of Web service based messages relatively to conventionally encoded messages. On the client device the ratio approaches a limit of about 3 (ksoap/conventional) and 2.5 (ws4d/conventional) as the numbers of readings per response increases. On the service device the ratio approaches the limit of about 2 for both Web service approaches. This means the client consumes marginally more energy than the service device.

Figure 6 shows the proportional share a task has on the average energy consumption and gives an indication on which task is most energy-intensive. For the Web service approaches most energy is consumed by sending and receiving a response over the network. For the conventional approach the case seems less apparent. On the client device, in Figure 6a, the share for serialising a request and being idle increases. As the time structure reveals, being idle is the major driver since its time share increases. The energy structure of the conventional approach suggests that energy is consumed while being idle. This is caused by the transceiver unit which was active the entire time ready to receive radio packets. For the Web service approaches the influence of the transceiver unit is overshadowed by the faster growing energy consumption for actually receiving a response. On the service device in the conventional approach (Figure 6b), the share for processing and serialising a response increases while the share for sending a response decreases. The conventional approach benefits from the fact that up to three readings fit into a radio packet. Therefore, the amount of energy for sending readings does not increase as fast as the amount of energy to serialise each reading. The message size of Web service approaches exceeds the possible data volume of one radio packet. With the increasing size of response messages, the amount of energy for serialisation increases almost as fast as the amount of energy for sending the message.

V. Discussion

The result of our performance study is empirical data that quantifies the overhead of Web services on embedded devices. In line with the general perception, the study demonstrates that a Web service-based approach is more resource-intensive than a conventional approach. The study

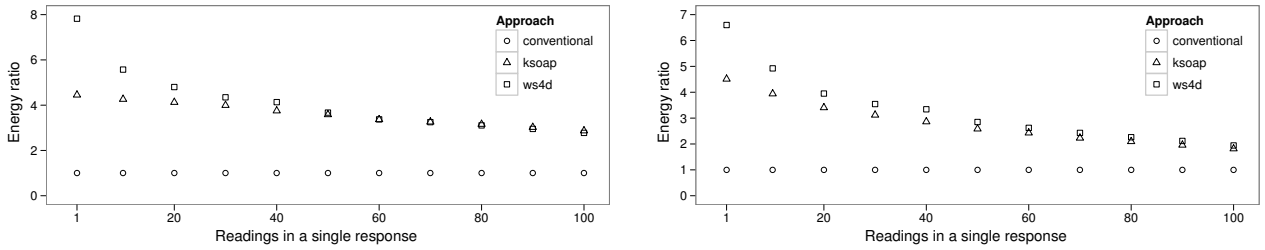


Figure 5. Effect on average energy consumption on (a) client device and (b) service device relatively to conventional approach

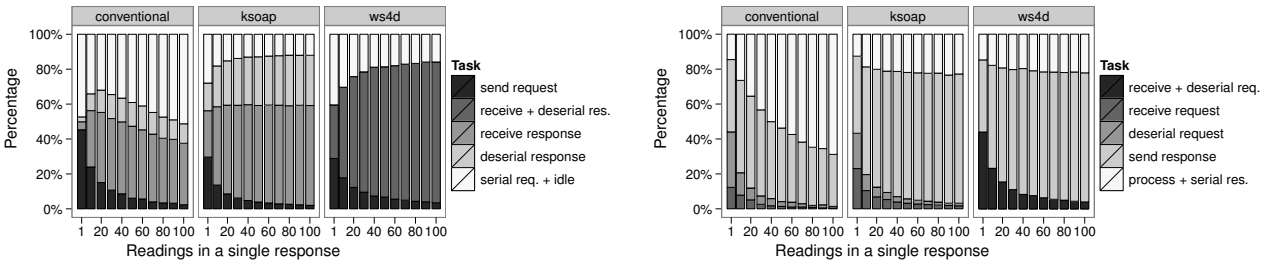


Figure 6. Structure of energy consumption on client device (a) and service device (b)

focused on the Web service overhead relative to a conventional approach and its main results can be summarised as follows:

- The required disk space for Web service enabling libraries varies between 61 KByte (ksoap) and 478 KByte (ws4d).
- The ratio for message size approaches a limit of about 5, i.e., Web service messages were at least 5 times larger than conventional messages.
- The ratio for response time approaches a limit of about 2.5 to 2, i.e., Web service messages took at least 2.5 to 2 times longer than conventional messages.
- The ratio for energy consumption on the client device approaches a limit of 3 to 2.5, i.e., on the client device Web service message consumed at least 3 to 2.5 times more energy than conventional messages.
- The ratio for energy consumption on the service device approaches a limit of about 2, i.e., on the service device Web services consumed at least 2 times more energy than conventional messages.

In the scenario two sensors exchanged SOAP-based Web service messages that included a list of temperature readings. The results are partly application-specific because the message size depends on the choice of parameter encoding and on the length of the chosen tags, namespaces, and service endpoint names. On the other hand, as the study builds on two main state-of-the art Web service libraries for resource-constraint systems and IEEE 802.15.4 as a standard protocol for radio communication, it is likely that the trend of the results are similar for different applications in resource-constraint settings.

It depends on the individual application requirements whether the outlined overhead of Web services is acceptable. In wireless sensor networks, for example, the strict hardware constraints of small-size sensor platforms would prevent the use of ksoap2 and ws4d Java ME because the required disk space is not available. However, alternative libraries, such as gSOAP [5] and ws4d-gSOAP [10], allow compile optimisations for a specific target platform and thus may provide smaller binaries. Researchers also explore restricting the message parser to fit the requirements of low-cost and deeply embedded devices [11], [12]. As a result, the overall code size is reduced and less disk space has to be allocated.

Although the overhead can be quantified by the limit it approaches, the study shows that such a limit may only be reached if multiple pieces of structured data are included in a message. For example, in Figure 2b, if a Web service message includes a single sensor reading, it is 25 (ws4d) times larger than a conventional message. However, as soon as ten readings are included, the overhead drops rapidly and the Web service message is only seven times larger. Notice a similar behaviour for response time and energy consumption where the overhead for a one reading is significantly higher than the calculated lower limit. For embedded systems that periodically exchange small pieces of data (e.g., a single reading) the overhead will outperform the benefit of exchanging structured data with SOAP.

Most expensive, in terms of time and energy, is the transmission and reception of Web service messages. These two tasks incur the most proportional cost irrespective of whether one or multiple pieces of structured data are exchanged. This is different for the conventional approach: The more data is exchanged, the more resources will be spent on processing data rather than transmitting it. For Web services to achieve the same behaviour, the overhead causing the expensive data transmission and reception has to be reduced. Empirical data, presented in this paper, provides scientific evidence that the size of Web service messages has a negative effect on the response time and energy consumption of hardware-constrained embedded devices. For a single piece of Web service encoded structured data, several packets have to be allocated on the physical link layer. Data transmission and reception take longer and more battery is discharged. For wireless sensor platforms, higher energy consumption is critical because it reduces the platform's lifetime. Once a sensor node is deployed in the field, it rarely has the means to recharge and energy conservation is a main concern.

Emerging research investigates different possibilities to reduce the size of Web service messages. One approach is to minimise the XML overhead by applying suitable compression and tag compacting techniques [13]. Another approach focuses on omitting SOAP and solely using HTTP [14]. A message is then either encoded into the URL or, if more structured, encoded into XML and carried in the HTTP body. In our own work, we are looking at streamlining the overhead of using Web service specifications.

ACKNOWLEDGMENT

This work was supported, in part, by Science Foundation Ireland grant 03/CE2/I303_1 to Lero - the Irish Software Engineering Research Centre (www.lero.ie)

REFERENCES

[1] S. N. Srirama, M. Jarke, and W. Prinz, "Mobile web service provisioning," in *Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW)*. IEEE, 2006, pp. 120–125.

[2] D. Savio and S. Karnouskos, "Web-service enabled wireless sensors in SOA environments," in *International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2008, pp. 952–958.

[3] M. Asif, S. Majumdar, and R. Dragnea, "Hosting web services on resource constrained devices," in *International Conference on Web Services (ICWS)*. IEEE Computer Society, 2007, pp. 583–590.

[4] H. Schmidt, A. Köhrer, and F. J. Hauck, "SoapME: A lightweight java ME web service container," in *Workshop on Middleware for Service Oriented Computing (MW4SOC)*. ACM, 2008, pp. 13–18.

[5] R. van Engelen, G. Gupta, and S. Pant, "Developing web services for C and C++," *Internet Computing*, vol. 7, no. 2, pp. 53–61, 2003.

[6] DPWS, "Devices profile for web services (DPWS)," 2006, http://schemas.xmlsoap.org/ws/2006/02/devprof/.

[7] G. Moritz, E. Zeeb, S. Prüter, F. Golatowski, D. Timmermann, and R. Stoll, "Devices profile for web services in wireless sensor networks: Adaptations and enhancements," in *International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2009.

[8] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, "Tiny web services: Design and implementation of interoperable and evolvable sensor networks," in *International Conference on Embedded Networked Sensor Systems (SenSys)*. ACM, 2008, pp. 253–266.

[9] ksoap2, http://ksoap2.sourceforge.net/.

[10] "Web services for devices (ws4d)," http://ws4d.e-technik.uni-rostock.de.

[11] G. Moritz, S. Prüter, D. Timmermann, and F. Golatowski, "Web services on deeply embedded devices with real-time processing," in *International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2008, pp. 432–435.

[12] D. Villa, F. J. Villanueva, F. Moya, F. Rincón, J. Barba, and J. C. López, "Web services for deeply embedded extra low-cost devices," in *International Conference on Advances in Grid and Pervasive Computing*. Springer, 2009, pp. 400–409.

[13] M. Ericsson, "The effects of xml compression on soap performance," *World Wide Web*, vol. 10, no. 3, pp. 279–307, 2007.

[14] D. Guinard and V. Trifa, "Towards the web of things: Web mashups for embedded devices," in *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences)*, 2009.