# Integration of Wireless Sensor and Actuator Nodes with IT Infrastructure Using Service-Oriented Architecture

Rumen Kyusakov, *Student Member, IEEE,* Jens Eliasson, Jerker Delsing, *Member, IEEE,* Jan van Deventer, and Jonas Gustafsson

***Abstract*—A large number of potential applications for Wireless Sensor and Actuator Networks (WSAN) have yet to be embraced by industry despite high interest amongst academic researchers. This is due to various factors such as unpredictable costs related to development, deployment and maintenance of WSAN, especially when integration with existing IT infrastructure and legacy systems is needed. Service-Oriented Architecture (SOA) is seen as a promising technique to bridge the gap between sensor nodes and enterprise applications such as factory monitoring, control and tracking systems where sensor data is used. To date, research efforts have focused on middleware software systems located in gateway devices that implement standard service technology, such as Devices Profile for Web Services (DPWS), for interacting with the sensor network. This paper takes a different approach - deploying interoperable Simple Object Access Protocol (SOAP)-based web services directly on the nodes and not using gateways. This strategy provides for easy integration with legacy IT systems and supports heterogeneity at the lowest level. Two-fold analysis of the related overhead, which is the main challenge of this solution, is performed; Quantification of resource consumption as well as techniques to mitigate it are presented, along with latency measurements showing the impact of different parts of the system on system performance. A proof-of-concept application using Mulle - a resource-constrained sensor platform - is also presented.**

***Index Terms*—Sensor networks, WSN, SOA, Web services, SOAP**

## I. Introduction

THE ability of networked, embedded devices to monitor and control various physical parameters of the environment as well as communicate the data over the Internet makes them a foreseeable source of innovation in many fields: from factory automation to use in smart homes and healthcare. While the benefits of integrating these devices with enterprise systems and services are evident from the perspective of business process synergy, many challenges still prevent widespread integration of sensor nodes into Manufacturing Execution Systems (MES), Enterprise Resource Planning (ERP), accounting and distribution systems. More details regarding the opportunities and challenges of applying WSANs in industrial environment are presented in [1] as well as in the work of Andreas Willig on wireless industrial communications [2].

Manuscript received September 10, 2010; revised January 21, 2011, August 17, 2011 and February 5, 2012. Accepted for publication March 6, 2012



R. Kyusakov, J. Eliasson, J. Delsing, J. van Deventer and J. Gustafsson are with the Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology S-97187 Luleå, Sweden

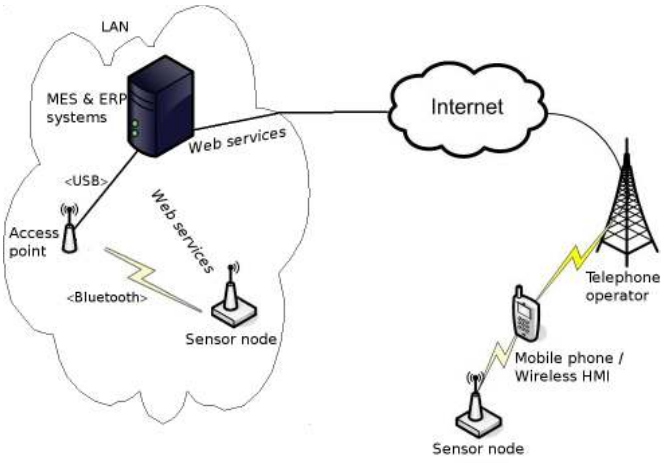


Fig. 1. Sensor nodes are integrated with enterprise systems using standard SOAP-based web services

Dealing with the heterogeneity of devices and software systems requires a flexible solution that can lower complexity and decrease development, deployment and system maintenance costs. Service-Oriented Architecture (SOA) has proven successful in leveraging these costs and it is seen as enabling technology for the development of enterprise systems in industrial domain [3]. Moreover, research analysis has shown its applicability for embedded systems development [4]. The prototype systems implemented within the European project SIRENA [5] as well as some commercial products that provide support for Devices Profile for Web Services standard [6] further prove the applicability of the SOA concept in the embedded domain. However, applying SOA to deeply constrained devices such as sensor nodes is still an open research problem due to unacceptable overhead. Some of the proposed solutions are based on modifications of the SOA protocols that simplify the implementation and lower the resource requirements [7]. However, the majority of research efforts have been directed towards using middleware software running on more capable devices or gateways as suggested by Wolff et al. in [8] or in the work of R. Bosman et al. [9]. This middleware is responsible for exposing the functionality of the whole sensor network as services using standard SOA

technology. In this way, communication within the network is still based on specialized, proprietary protocols. This approach has the benefit of leaving the resource-intensive tasks related to standard service implementation to the gateway, but also has some drawbacks such as a single point of failure, an inability to support heterogeneity on the node level [10], etc.

Although the node-level service implementations have already been proposed, there are no studies investigating the applicability of deploying fully interoperable and compliant services, such as those described in WS-I Basic Profile 1.0, directly on the sensor nodes. This is due to the general perception that the use of XML-based services on highly constrained sensor nodes is inapplicable or even impossible, as stated by Leguay et al. [11]. The higher overhead, in terms of power consumption, latency, RAM and CPU usage, related to serialization, transmitting and parsing of verbose XML messages is undisputed, and has in fact been well studied by Groba et al. [12] where empirical data that quantifies the overhead of web services on embedded devices is presented. Especially challenging are the high memory requirements resulting from the need for large buffers used to accommodate the XML documents.

In this paper, we present few techniques for improving efficiency that allow us to deploy standard SOAP web services on resource-constrained sensor nodes. These techniques are implemented in a proof-of-concept application that connects sensor nodes to an enterprise application. The architecture of the experimental setup is shown in Figure 1. Among others, we applied sensor data aggregation for reducing the transmission time and active mode intervals of the nodes, and hence increasing battery life. As this technique is not applicable in general case a real-world scenario which allows for such aggregation is also presented. In [13], Lee et al. used similar approach for industrial monitoring application.

The remainder of this paper is structured as follows - we provide a motivation for our work in Section II. Section III summarizes the related work in the area and presents some of the technologies used. Section IV goes into details about the problems related to the use of SOA in WSAN. Section V presents our sample application together with performance measurements. In Section VI, we give the possible improvements and extensions to our work, and Section VII concludes the paper.

## II. Motivation

As pointed out by Jammes et al. [14], the manufacturing enterprises, pushed by the global competition, are seeking ways to increase their responsiveness to the market demands on a real-time scale. At the same time, the costs for maintaining the process flows evolution or modification are substantial due to the semi-automatic, or even error prone manual, configurations involved. A recent study by Candido et al. proposed an architecture that supports the device and process lifecycle evolution based on SOA and Evolvable Production Systems (EPS) [15]. As part of this architecture, the devices have SOA interfaces that allow high level business applications to interact with them without any intermediary protocol gateways - a concept

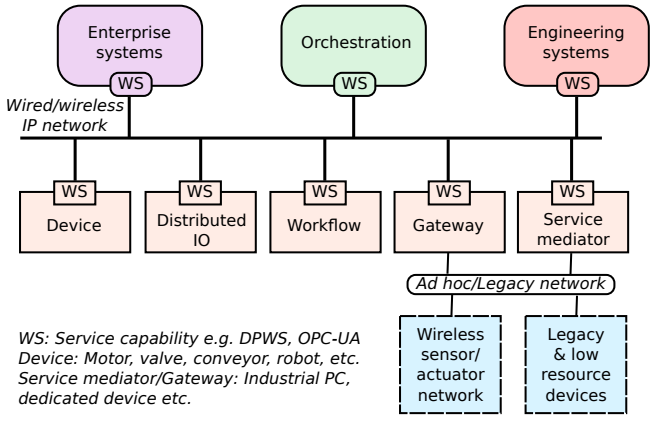


Fig. 2. The SOCRADES cross-layer approach

also suggested in [16]. The support for cross-layer integration between the shop floor and enterprise systems was also a main objective for the SOCRADES project [17]. As an outcome of this project, an architecture for vertical integration based on the SOA approach was proposed, where the ERP and MES systems together with shop floor devices are integrated using web services. Kalogeras et al. presented similar architecture with emphasis on the use of web services, workflows and ontologies [18]. A diagram from the SOCRADES Roadmap shown in Figure 2, represents the concept of applying SOA approach for vertical inter-enterprise integration. As depicted, the resource constrained devices, including wireless sensors and actuators, are exposed to the SOA interface through service gateways or mediators. The work presented in this article is an extension to the aforementioned SOA architecture aiming at deploying the service interface provided by the gateways directly on the wireless nodes. This is made possible due to the advancement in embedded systems hardware but also the application of resource-aware implementation techniques.

## III. Background and related work

Service-Oriented Architecture denotes the usage of well-defined and self-contained function calls between distributed nodes independent of the location and platform of the parties involved. It also implies that interoperable network protocols for communicating service requests and responses are available. Although many challenges still remain, there are different approaches for providing low layer (physical and data-link) integration of wireless networks in an industrial environment [19], [20]. In this work we focus on providing application layer integration with the use of an access point for the data-link layer integration and TCP/IP stack on the network and transport layers.

There are many service technologies that are built upon the SOA approach: CORBA, UPnP, OPC-UA, Jini and different flavors of web service technology (SOAP-based, RESTful [21], etc). The web services conforming to the Web Services Description Language (WSDL) specification are designed to be application- and transport protocol-agnostic, which leads to compatibility issues. For that reason, different web service profiles are specified to leverage the diversity of network

protocols used and to adapt the specifications to a particular application domain. Services in enterprise systems mostly conform to WS-I Basic Profile 1.0, while Devices Profile for Web Services is used to define a set of protocols to enable plug-and-play behavior for embedded networked devices. Both profiles rely on SOAP as an application layer protocol for serialization of service requests and responses.

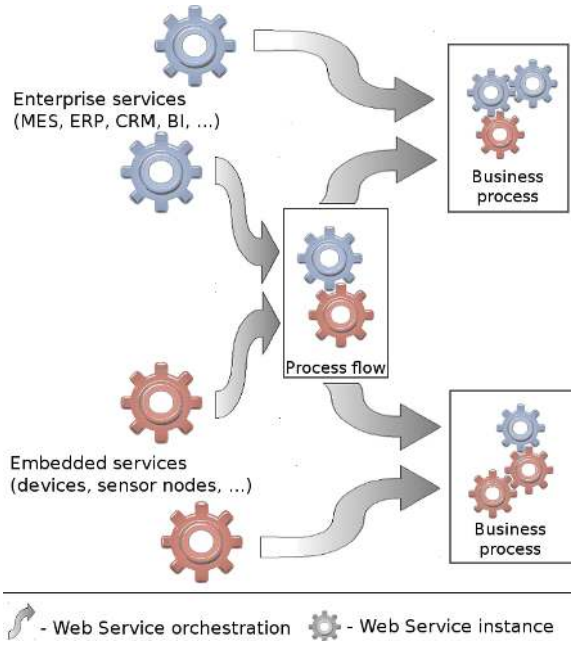


Fig. 3. Direct orchestration of sensor node and enterprise web services is made possible due to their compatibility

Advantages and disadvantages of the aforementioned service technologies applied to different applications are already being studied by researchers (e.g. [5]); thus, a comparison of them is not included in this paper. The analysis performed in research projects such as SIRENA, SOCRADES and within the research program ITEA gives priority to SOAP-based web services in which the devices are integrated with the IT systems using DPWS. The main argument in support of this architecture is the possibility to apply service orchestration of embedded and system services directly without the need for adapters, as shown in Figure 3. A white paper by Boyd et al. [22] provides further reading on service orchestration and other SOA concepts along with case studies of applying SOA to manufacturing infrastructure.

The use of proprietary or nonstandard SOA implementations requires translation middleware when working with standardized service orchestration, such as Business Process Execution Language (BPEL). As our approach aims at limiting the external dependencies of the SOA implementation for devices the work presented in this paper considers standard SOAP-based web services.

To ensure interoperability, SOAP web services are entirely based on open standards and rely heavily on the usage of XML and XML Schema Definition Language (XSD). Thus, each SOAP message is a XML document that must first be serialized, transmitted, received and then parsed. To avoid these resource-intensive operations being performed on the sensor nodes, researchers are investigating the use of middleware software deployed on gateway devices that first communicate with the nodes in an ad-hoc manner and then translate their functionality as web services to external systems. An example of such a design was proposed by Avilés-López et al. [23]; In their system, the middleware included an advanced registry mechanism. A similar solution that also incorporated a lightweight, ad-hoc service protocol within the sensor network was presented by Leguay et al. [11]. In that work, the translation between internal and external DPWS-compatible services was done on the gateway. The architecture supports one-to-one, but also many-to-one, relations between the services with a highly flexible eventing mechanism built upon hierarchical subscriptions. Another approach more closely related to the work presented in our paper is that by Priyantha et al. at Microsoft Research [24]. Instead of using specialized, ad-hoc services for node-to-node communications, they proposed to use web services described by WSDL. To keep the overhead low, these services were implemented using HTTP binding and not SOAP. This provides for shorter and easier to parse and serialize messages but also implied constraints on the structure of the data transmitted and impaired the compatibility with enterprise systems. To address these issues, Priyantha et al. proposed an external server called *Controller* that more or less fulfilled the role of the gateway middleware presented in the previous papers. The controller served as a proxy that translated the internal web services to SOAP-based services and provided eventing through the use of WS-Eventing. In this way, the client applications communicated with the sensor nodes indirectly through the controller. Also presented in that paper are different techniques to lower the XML-related impact on performance as well as an analysis of possible application scenarios where this approach will lead to cost savings.

In contrast with the aforementioned approaches, the solution presented in the present paper deploys fully compatible SOAP-based web services directly on a highly constrained sensor platform and hence eliminates the need for additional middleware. In this way, its main contribution is an efficient implementation that combines a lightweight TCP/IP stack - lwIP [25] and a gSOAP [26] web service toolkit. The lwIP provides two different APIs to access the network services: a low-level ”raw” API relies on callbacks, and a higher-level ”sequential” API is easier to work with but also implies higher resource consumption. We used the ”raw” API to minimize the footprint of our solution.

The gSOAP toolkit includes a highly efficient runtime environment to process SOAP messages that uses either a general-purpose XML parser or an application-specific one that can be generated from the service description (WSDL) file. The use of an application-specific parser and serializer provides for lowering the RAM and ROM utilization, as the processing logic for the input and output generators is optimized for the specific usage, and there are no execution paths left unused by the application.

SOAP-based service implementation with a general-purpose XML parser on Tmote Sky was reported by Yazar et al.

[27]. Their solution differs from our approach in that it does not provide tool support for developing SOAP-based services but rather is only used to evaluate the performance of their RESTful implementation, as stated in their paper: "The SOAP-based implementation is used as a reference point in performance evaluation and is not intended for general use." Also not included in the work of Yazar et al. is a mechanism to dynamically discover the network location of a service, neither by their RESTful nor their SOAP implementation.

A work that is aiming at deploying standard-based embedded web services directly on resource constrained sensor nodes is available from open source project WS4D-uDPWS [28]. Although, the intended outcomes of WS4D-uDPWS and our approach are very similar the implementation techniques differ in many aspects. First, the network layer used in WS4D-uDPWS is uIP TCP/IP stack. It has smaller footprint than lwIP but provides lower throughput. Second, while we built our solution on an existing service implementation (i.e. stripped version of gSOAP), uDPWS provides its own web service runtime which is highly optimized and has smaller RAM and ROM footprint than our runtime. However, WS4D-uDPWS does not provide tool support for generating the application-specific parts of its runtime based on the WSDL service descriptions. Code-generation is provided, but it is based on text files with formatting and naming conventions specific to WS4D-uDPWS. Moreover, while the processing of the SOAP headers is automated, the parsing and serialization of the SOAP body is left to the web service developers.

## IV. SOA ON SENSOR NODES

Due to the number of nodes in WSN, it is very important to deploy and configure sensor nodes with the least manual work possible. The initialization and configuration parameters can depend on various conditions, most probably originating outside of the sensor network. Looking at factory automation as an example, these conditions are connected to MES systems but also to strategic decisions in ERP systems, historical data from databases, etc. Any changes done in these systems that affect the behavior of the sensor nodes must be propagated down while sensor data, after undergoing filtering and aggregation, must be propagated up. Using SOA all the way down to the smallest devices results in increased compatibility, where auto-configuration and plug-and-play capabilities can be modeled as services. In such way, higher flexibility for tuning or even changing the manufacturing processes is achieved stemming from direct interactions between all system components. However, this flexibility also leads to complex systems integration and difficulties when defining or verifying the required functionality of particular module or the system as a whole. Handling this complexity requires data models that constrain the possible interactions and formats for data exchange. Examples of such data models are OPC-UA information model or Business to Manufacturing Markup Language (B2MML) used to link business systems such as ERP and supply chain management systems with manufacturing systems such as control systems and MES. Similar models used for interfacing sensing devices are described by Sensor Web Enablement Framework of Open Geospatial Consortium.

### A. Web services

The main drawback to using SOAP-based web services is the need to parse verbose XML documents. However, there are already a number of compression techniques that require a factor of ten less RAM, CPU and bandwidth as compared to text-based XML. The most promising of these is Efficient XML Interchange (EXI) [29], a W3C recommendation as of March 10, 2011. EXI is defined as an alternative mean to represent the XML Information Set [30] that provides one-to-one translation to text-based XML representation. Depending on the document properties and processing options specified, EXI provides between 50% and 99% reduction in size and up to 15 times faster processing [31]. The work presented in this paper shows that even verbose XML can be used as a service message protocol for sensor nodes; future binary XML representations will only extend the applicability of the presented solution. Introducing the EXI encoding to embedded web service implementations however, will require the ability to change the XML parser and serializer with EXI ones. Our first attempt in this direction is the creation of EXIP open source project[1]. Another question arising from the use of binary encoding is how to connect embedded EXI encoded web services with text-based enterprise services. One such techniques that is already available as a commercial product is to introduce a transparent HTTP proxy in between. The role of the proxy is to translate binary EXI encoding to text-based XML and vise versa. More details on the opportunities and challenges of using EXI in industrial environment are presented in [32].

### B. Tools

The development of web service applications depends upon a runtime system responsible for the network communications, parsing, validation and serialization of service requests and responses. Besides the runtime system, software tools are used to map data structures in the XML to programming language constructs - also known as XML data binding. Based on the characteristics of our target domain, the required properties of the SOA runtime system and supporting tools are as follows:

- written in programming language that is used for sensor and actuator nodes development - currently most widely used are C and its dialect nesC.
- easily portable on different embedded platforms.
- featuring small footprint implementation.
- highly configurable - it should be possible to remove features that are not used or needed.

For C language, there are two web service toolkits, namely, gSOAP and Apache Axis2/C. While gSOAP supports and has been ported on several embedded platforms, Axis2/C is mostly used on Windows and Linux machines. Moreover, gSOAP runtime has a wide range of features that can be selectively included or excluded from it. The version of gSOAP used in our solution has the following components removed: XML DOM parser, HTTPS and SSL support, compression, logging

[1]Efficient XML Interchange Processor - http://exip.sourceforge.net/

module, all support for attachments including SOAP with Attachments, MIME, DIME or MTOM, HTTP chunked transfer mode.

## V. Proof of Concept

WS-I Basic Profile 1.0 defines a bare minimum of constraints on the WSDL specification that make different web service implementations compliant. Examples of such constraints are the use of SOAP version 1.1 binding, HTTP 1.0 or HTTP 1.1 as a transport protocol. The applications developed using our solution are compliant with WS-I Basic Profile 1.0 provided that the "-1" command-line option is used when executing gSOAP *soapcpp2* code generator. The current enterprise systems are mostly conforming to this profile which enables interoperability with our SOA approach for sensor nodes.

DPWS, in contrast, poses many more requirements aimed at providing plug-and-play capabilities as well as automatic deployment and configuration. It also denotes the usage of SOAP version 1.2 as well as the addressing fields in the SOAP header defined in WS-Addressing specification. Moreover, a set of predefined services must be available on the devices willing to comply with the DPWS standard. As an example, a manifest service called *device* is responsible for hosting and advertising the other services that represent the functionality provided by the device. Another predefined service, with an interface consisting of six operations, is specified in WS-Discovery - a protocol that enables dynamic discovery of available services on the network without the use of centralized registry such as UDDI. All six operations use SOAP-over-UDP to minimize network traffic. Figure 4 shows all of the protocols included in the DPWS specification, with those not covered by our solution illustrated with hatching.

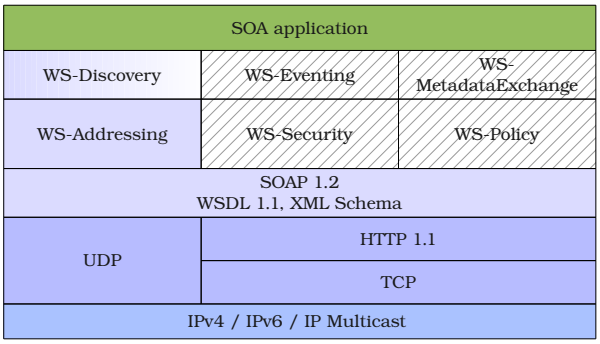


Fig. 4. DPWS protocol stack. Parts not covered in this work are illustrated with hatching

When the required settings for using SOAP v1.2 and SOAP-over-UDP are specified as described by the gSOAP documentation, our current solution supports SOAP-over-UDP, SOAP 1.2, WSDL 1.1, WS-Addressing and certain parts of WS-Discovery. Two WS-Discovery operations are included in our implementation: Probe, which is a query multicasted to specific IP multicast address and port, and ProbeMatch, which is the response of the queried nodes to the Probe message. The use of discovery proxies, as defined by the specification is not supported. Nevertheless, this limited implementation is sufficient to locate a service advertised by a WS-Discovery-compliant device.

The security scheme defined by DPWS enables protection of the service executions in three directions: authentication of the parties involved, message integrity protection and confidentiality. While the majority of the target applications will not require confidentiality for sensor data and/or actuator control data, authenticity and integrity are crucial especially for wireless communications. However, the resources available on current sensor platforms are not sufficient for supporting standard based authentication mechanisms based on digital certificates and asymmetric cryptography. For that reason, the presented approach is only appropriate for non-critical applications where the sensor nodes are behind enterprise firewall.

### A. Architecture

As was already stated, the web service implementation presented in this paper is built upon the gSOAP toolkit. The gSOAP design supports different network layers with BSD-socket API supported out of the box. However, its runtime is written with the perception that the network interface it uses supports sequential execution, which requires the use of threading. Thread-based network APIs provide abstraction of the complex event-driven nature of network communications. The trade-off inherited from this abstraction is a higher resource consumption, which makes it not suitable for highly constrained sensor nodes [25]. So, to use the event-based "raw" lwIP API, the network layer of gSOAP runtime was rewritten and an additional lwIP wrapper was introduced. This includes splitting of the sequential execution blocks that contain blocking network operations into smaller non-blocking programming sequences connected with callback functions. As an example, consider the following simplified programming fragment that uses threaded network layer:

```
Block_1() {
   blocking_connect();
   /* The TCP connection is established */
   serialize_http_header();
   blocking_send();
   /* The http header is sent */
   serialize_soap();
   blocking_send();
   /* The soap message is sent */
   cleanup();
}
```

The equivalent functionality based on non-blocking lwIP network operations and callbacks is coded as follows:

```
Block_1() {
   store_soap_state();
   lwip_connect(); /* calls Block_2() when connected*/
}
Block_2() {
   serialize_http_header();
   lwip_send(); /* calls Block_3() when the header is sent*/
}
Block_3() {
   serialize_soap();
   lwip_send(); /* calls Block_4() when the soap is sent*/
}
Block_4() {
   cleanup();
}
```

The listings also present the concept of transmission on the fly - when the HTTP header is serialized, it is sent over the network. Then the sending buffer is released and used for storing the SOAP message before its transmission. The same technique is used on the receiving side: when the HTTP header is received it is parsed and then the receiving buffer is released. In this way, the size of the buffers, and hence the RAM usage, can be restricted.

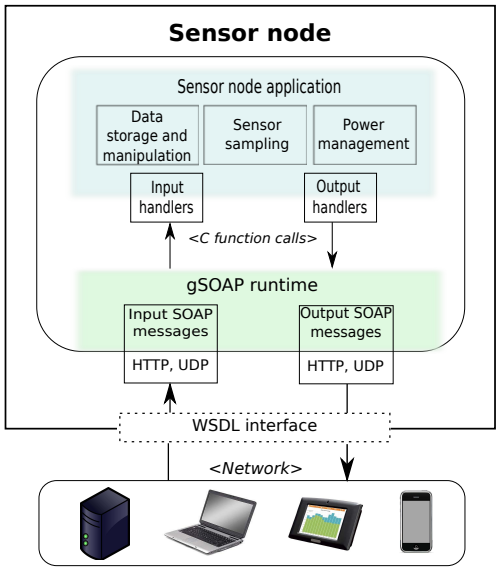


Fig. 5. System architecture

The overall architecture is depicted in Figure 5. The modules responsible for power management, sampling the sensors and aggregating the data are not affected by the service interface; hence, legacy code can be reused. Instead of connecting the input and output of the sensor application to a network API implementing proprietary, specialized protocols, the data are passed to the gSOAP runtime using handlers. The runtime serializes the output data to a SOAP message, and then uses lwIP to send it over the network. The opposite is true for input data: it is first parsed and then forwarded to the sensor application. The interface describing the services provided by and consumed by the nodes is available through the use of standardized Web Service Description Language. This allows for so-called top-down SOA development, where the WSDL interfaces for the nodes are defined first - usually using graphical tools[2]- and then are used to generate the SOAP runtime. At the end the developer connects the provided interface with the sensor application. This is the approach used in the development of our testbed, described in the subsequent subsections.

### B. Mulle sensor platform

The Mulle sensor platform [33] used in our experimental setup is equipped with a Renesas M16C/62 microcontroller running at 10 MHz with 31kB RAM and 384 kB programming memory. A Mitsumi Bluetooth radio transceiver, operating at 57 kbits/s, was used in our testbed to enable mobility through the use of a mobile phone as an access point. The Mulle sensor platform is also available with an IEEE 802.15.4 radio transceiver, which also can be employed instead of the Bluetooth one, provided that the lwIP stack is configured for using it.

[2]In our use case we used Eclipse WSDL Editor

### C. Proof of concept experiment

To test the applicability and performance of our solution, several services were implemented. The first was a very simple, light service with operations for switching a LED on and off and for checking the status of the LED. Tests were performed under different scenarios with the service being hosted on a sensor node using our solution, on a stationary PC or on both. To check compatibility, two different implementations of the light service were used on the PC. The first was C-based, using a gSOAP port for Linux. The other was Java-based, using JAX-WS API running on a GlassFish server. In both cases the interactions between the sensor node and the PC proceeded without any compatibility problems.

For the second test, it was decided to replicate a real-world scenario where, despite the overhead, the SOA implementation would still be beneficial to use [34]. In such an application, the system must lack any real-time properties. Also, it should be possible to aggregate the sensor data before its dissemination that should happen at long intervals. The source of inspiration was a district heating project [35] aimed at increasing the efficiency of energy distribution.

*District heating scenario:* In today's district heating substations, different sensors and actuators are hard-wired together. This limits the possibilities for system optimization as communication barriers limit the information interchange. With wireless sensor platforms integrated in such district heating devices as a circulation-pump, heat meter and temperature sensors, greater opportunities for system optimization are achieved as information can be interchanged without limitations.

With a service-oriented architecture integrated in the end nodes, there is no direct need for a central control unit, as the sensor nodes are powerful enough to control the relatively slow heating process. The slow process makes the use of SOA over WSAN particularly suitable as there is no need for frequent data transmission, which would decrease the expected life-length of the sensor platforms. Thus, the nodes are in sleep mode most of the time with short active intervals for sensor sampling and data aggregation. The transceiver is infrequently turned on only when the highly aggregated data are sent directly to the enterprise systems responsible for heating process management.

In our testbed, nodes were equipped with temperature and humidity sensors, and the data sent to the server consisted of multiple metrics, such as current sensor readings as well as the average, minimum, maximum and standard deviation of the temperature and humidity for a given period, as shown in Figure 6. The intervals in which the sensor nodes communicate the data were controlled by the management system. For the implementation of the heating process management system we chose the SOA Swordfish toolkit that supports deployment

```
<hts:GetSummary>
  <Temp>
    <Current>19.3</Current>
    <Average>18.2</Average>
    <Min>17.4</Min>
    <Max>21.0</Max>
  </Temp>
  <Humidity>
    <Average>65</Average>
    <StdDeviation>5.0</StdDeviation>
  </Humidity>
</hts:GetSummary>
```

Fig. 6. Segment of the service request initiated by the Mulle sensor node. It contains an aggregation of the sensor data for the period of interest

on a Java EE application server. Also implemented on the server was a Java version of WS-Discovery, which was used to advertise the heating service on the network.

The implementation started with modeling the desired interactions between the sensors and the management system using Web Service Description Language. The abstract WSDL service definitions were then fed into Swordfish framework to generate the serialization and parsing code. The same WSDL interface was used by the gSOAP code generation tools. The code produced was then combined with our modified gSOAP runtime, lwIP and our network layer wrapper, which were deployed on the Mulle sensor platform. To avoid manual configuration of the server address for each sensor node, two operations of the WS-Discovery were also implemented and deployed on the sensor platform to dynamically locate the heating service.

*Mobility scenario:* The heating management service was also used as a testbed for a mobility scenario where a sensor node is being carried by a person with a Bluetooth-enabled mobile phone. This can be useful for assisting and documenting manual inspections and diagnostics of industrial equipment by technicians for example. The phone provides access to a 3G network that enables connectivity of the sensor node and the Java server on a TCP/IP layer. With this infrastructure setup, the sensor node seamlessly communicates the aggregated sensor data to the enterprise application using web services. However, an important requirement in this scenario is the presence of a secured VPN connection between the mobile phone/wireless HMI and the enterprise network as our solution does not support the security mechanisms defined in the DPWS specification and the connection is established from outside the enterprise firewall.

### D. Performance measurements

A gSOAP runtime with no network layer or deployed services requires around 5.5 kB of RAM and 123 kB of programmable memory. For each service (client or server) added, an additional 13 kB of ROM is required, on average. During service invocation 3 kB of RAM are allocated and hence need to be available on the system. If only one service is executed at a time, the overall RAM consumption is 8.5 kB independent of the number of services added. However, allowing different service executions to be interleaved requires an additional 3 kB of RAM for each service deployed. The time needed to parse and serialize a particular request or response is highly dependent on its size, structure and the number of namespaces used in the XML document. Table I shows the processing time for messages used in the LED and heating service examples.

| SOAP Messages | Parsing time (ms) | Serialization time (ms) |
|---|---|---|
| Heating service request 654 bytes | – | 14.5 |
| Heating service response 479 bytes | 14.5 | – |
| LED check status service request 386 bytes | 24 | 7 |
| LED check status service response 414 bytes | 26 | 16 |
| LED switch service request 415 bytes | 25 | 7.5 |

TABLE I
TIME NEEDED BY THE MULLE SENSOR PLATFORM TO PROCESS SOAP MESSAGES

To evaluate the latency overhead, we used the *GetStatus* operation of the LED web service hosted on a PC running Linux with a Bluetooth v1.2 dongle. A Mulle sensor node, with LED service client implemented using our solution, was also set up within transmission range. All communication were performed using the Bluetooth Personal Area Networking (PAN) profile where the PC was hosting the Network Access Point (NAP) service and the Mulle acting as user (PAN-U). The deployment of the LED service client on the Mulle node allowed it to use sleep mode such that it periodically waked up and sent *GetStatus* SOAP request, then waited for the response, parsed it and went back to sleep mode. Having the node as a LED server would increase the power consumption substantially as it would require the Bluetooth module to be powered on at all times. The current approach allows the Bluetooth module to be duty-cycled.

The same interactions between the PC and the Mulle node were implemented using a bare TCP approach with one-byte payload. In such way the type of operation (*GetStatus* or *Switch*) is encoded using a single bit and another bit is used to indicate the status (*on* or *off*) of the LED. Although it cannot be applied in practice, the ad-hoc one-byte TCP implementation represents the shortest possible encoding of the LED operations over TCP/IP thereby enabling the overhead of our solution to be measured.

Figure 7 shows the completion time for our SOAP-based solution (514 ms) compared to the bare TCP approach (129 ms). The measured time, averaged over 50 transmissions, is given for the three phases of the service execution i.e. TCP connection establishment, SOAP message transmission and XML processing. The results show that the time to parse and serialize SOAP messages by the Mulle sensor node denotes just a small part of the latency related to web service invocation - 33 ms for the *GetStatus* LED service or about 6.5 % of the total service execution time. The larger part is due to the actual transmission. This observation proves that the use of more compact representation of the service messages, even compression and other techniques which affect the processing

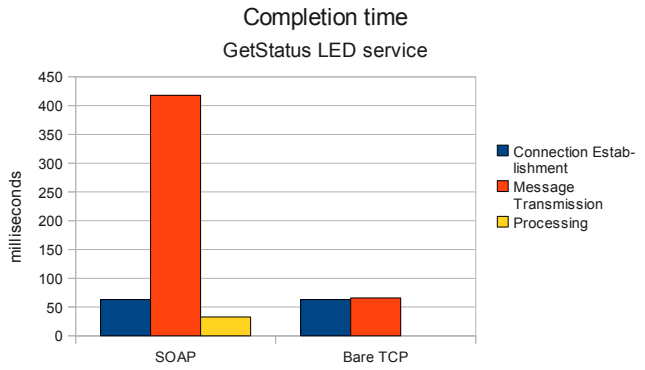


Fig. 7. Completion time in milliseconds for service execution

speed, will improve the real-time properties of the system. Moreover, it takes almost four times longer to complete the SOAP service compared to a one-byte TCP payload ad-hoc representation of the LED service operations.

## VI. Future work

The work presented in this paper shows that even highly resource-constrained networked sensor nodes can be integrated within an IT infrastructure using standard SOA technology. However, even efficient implementation poses significant performance overhead, which makes the solution only suitable for applications where the sensor data can be heavily aggregated and transmitted over relatively long periods. One such example from energy management domain was presented in the paper, but other applications for home and factory automation networks would also meet this criterion. The level of data aggregation and the length of non-transmission intervals needed depend on many parameters such as power consumption, sleep schedule, real-time requirements, etc. Therefore, a precise analysis showing their exact threshold that would make this solution beneficial is an important topic for future work. This analysis must take into account all parameters, and their interdependence, that play a role in the applicability of the SOAP-based web services. This analysis must also provide a comparison with emerging standards for embedded web services such as those described by Shelby in [36].

Applying the same SOA approach to full-scale sensor networks, where most communications are multihop and the nodes use IEEE 802.15.4 radio, is another area for future exploration. In addition, different ways to lower the related overhead should be investigated. The most important in this respect is the use of binary encoding for the SOAP messages.

## VII. Conclusion

Integration of high-end systems with deeply embedded wireless sensor nodes is an important area of research that aims to provide new possibilities for control and monitoring applications. The solution presented in this paper enables standard-based and direct application-layer integration between web service-enabled IT systems and resource-constrained sensor nodes. Its main contribution is the efficiency of the provided implementation, which combines light-weight TCP/IP stack implementation and SOAP-based web service implementation. In addition, we included performance measurements on the impact of this method on latency. One important observation was that the overhead related to SOAP message processing is very small compared to message transmission. We also showed an example application that can benefit from the SOA approach, despite the related overhead.

## Acknowledgment

The authors would like to thank the European Commission and the partners of IMC-AESOP project for their support. We gratefully acknowledge the valuable comments and suggestions from the anonymous reviewers and associate editor.

## References

[1] V. Gungor and G. Hancke, "Industrial wireless sensor networks: Challenges, design principles, and technical approaches," *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 10, pp. 4258 –4265, oct. 2009.

[2] A. Willig, "Recent and emerging topics in wireless industrial communications: A selection," *Industrial Informatics, IEEE Transactions on*, vol. 4, no. 2, pp. 102 –124, may 2008.

[3] L. D. Xu, "Enterprise systems: State-of-the-art and future trends," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 4, pp. 630 –640, nov. 2011.

[4] S. de Deugd, R. Carroll, K. E. Kelly, B. Millett, and J. Ricker, "SODA: Service Oriented Device Architecture," *Pervasive Computing, IEEE*, vol. 5, no. 3, pp. 94 –96, july-sept. 2006.

[5] H. Bohn, A. Bobek, and F. Golatowski, "SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains," in *International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006*, 2006.

[6] *Devices Profile for Web Services Version 1.1*, OASIS Std. [Online]. Available: http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.pdf

[7] I. K. Samaras, J. V. Gialelis, and G. D. Hassapis, "Integrating wireless sensor networks into enterprise information systems by using web services," in *SENSORCOMM*, 2009.

[8] A. Wolff, S. Michaelis, J. Schmutzler, and C. Wietfeld, "Network-centric middleware for service oriented architectures across heterogeneous embedded systems," in *EDOC Conference Workshop, 2007. EDOC '07. Eleventh International IEEE*, 15-16 2007, pp. 105 –108.

[9] R. Bosman, J. Lukkien, and R. Verhoeven, "Gateway architectures for service oriented application-level gateways," *Consumer Electronics, IEEE Transactions on*, vol. 57, no. 2, pp. 453 –461, may 2011.

[10] G. Moritz, E. Zeeb, F. Golatowski, D. Timmermann, and R. Stoll, "Web services to improve interoperability of home healthcare devices," in *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*, 2009.

[11] J. Leguay, M. Lopez-Ramos, K. Jean-Marie, and V. Conan, "An efficient service oriented architecture for heterogeneous and dynamic wireless sensor networks," in *Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on*, 2008, pp. 740 –747.

[12] C. Groba and S. Clarke, "Web services on embedded systems - a performance study," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, march 2010, pp. 726 –731.

[13] A. Lee and J. Lastra, "Data aggregation at field device level for industrial ambient monitoring using web services," in *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, july 2011, pp. 491 –496.

[14] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation," *Industrial Informatics, IEEE Transactions on*, vol. 1, no. 1, pp. 62 – 70, 2005.

[15] G. Candido, A. Colombo, J. Barata, and F. Jammes, "Service-oriented infrastructure to support the deployment of evolvable production systems," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 4, pp. 759 –767, nov. 2011.

[16] A. Ramos, I. Delamer, and J. Lastra, "Embedded service oriented monitoring, diagnostics and control: Towards the asset-aware and self-recovery factory," in *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, july 2011, pp. 497 –502.
[17] A. Cannata, M. Gerosa, and M. Taisch, "SOCRADES: A framework for developing intelligent systems in manufacturing," in *Industrial Engineering and Engineering Management, 2008. IEEM 2008. IEEE International Conference on*, 8-11 2008, pp. 1904 –1908.
[18] A. Kalogeras, J. Gialelis, C. Alexakos, M. Georgoudakis, and S. Koubias, "Vertical integration of enterprise industrial systems utilizing web services," *Industrial Informatics, IEEE Transactions on*, vol. 2, no. 2, pp. 120 – 128, may 2006.
[19] T. Sauter, "The three generations of field-level networks - evolution and compatibility issues," *Industrial Electronics, IEEE Transactions on*, vol. 57, no. 11, pp. 3585 –3595, nov. 2010.
[20] G. Cena, A. Valenzano, and S. Vitturi, "Hybrid wired/wireless networks for real-time communications," *Industrial Electronics Magazine, IEEE*, vol. 2, no. 1, pp. 8 –20, 2008.
[21] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Transactions on Internet Technology*, vol. 2, no. 2, pp. 115–150, 2002.
[22] A. Boyd, D. Noller, P. Peters, D. Salkeld, T. Thomasma, C. Gifford, S. Pike, and A. Smith, "SOA in Manufacturing - Guidebook," IBM Corporation, MESA International and Capgemini, Tech. Rep., 2008. [Online]. Available: ftp://public.dhe.ibm.com/software/plm/pdif/MESA_SOAinManufacturingGuidebook.pdf
[23] E. Avilés-López and J. A. García-Macías, "TinySOA: a service-oriented architecture for wireless sensor networks," *Service Oriented Computing and Applications*, vol. SOCA (2009) 3:99-108, pp. 99–108, 2009.
[24] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, "Tiny web services: design and implementation of interoperable and evolvable sensor networks," in *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*. New York, NY, USA: ACM, 2008, pp. 253–266.
[25] A. Dunkels, "Full TCP/IP for 8-bit architectures," in *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*. New York, NY, USA: ACM, 2003, pp. 85–98.
[26] R. A. van Engelen and K. A. Gallivany, "The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks," in *2nd IEEE International Symposium on Cluster Computing and the Grid*, 2002.
[27] D. Yazar and A. Dunkels, "Efficient Application Integration in IP-based Sensor Networks," in *Proceedings of ACM BuildSys 2009, the First ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings*, Berkeley, CA, USA, Nov. 2009.
[28] C. Lerche, N. Laum, G. Moritz, E. Zeeb, F. Golatowski, and D. Timmermann, "Implementing powerful web services for highly resource-constrained devices," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, march 2011, pp. 332 –335.
[29] *Efficient XML Interchange (EXI) Format 1.0*, W3C Std., March 2011. [Online]. Available: http://www.w3.org/TR/2011/REC-exi-20110310/
[30] J. Cowan and R. Tobin. XML Information Set (Second Edition). W3C. [Online]. Available: http://www.w3.org/TR/xml-infoset/
[31] G. White, J. Kangasharju, D. Brutzman, and S. Williams, "Efficient XML Interchange Measurements Note," W3C, Tech. Rep., 2007. [Online]. Available: http://www.w3.org/TR/exi-measurements/
[32] R. Kyusakov, H. Mäkitaavola, J. Delsing, and J. Eliasson, "Efficient XML Interchange in factory automation systems," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, nov. 2011, pp. 4478 –4483.
[33] J. Johansson, M. Völker, J. Eliasson, Å. Östmark, P. Lindgren, and J. Delsing, "Mulle: A minimal sensor networking device - implementation and manufacturing challenges," in *Proceedings IMAPS Nordic*, 2004, pp. 265–271.
[34] J. Delsing, J. Gustafsson, and J. van Deventer, "A service oriented architecture to enable a holistic system approach to large system maintenance information," in *Proceedings CM-MPFT*, 2010.
[35] J. van Deventer, J. Gustafsson, J. Eliasson, J. Delsing, and H. Mäkitaavola, "Independence and interdependence of systems in district heating," in *Systems Conference, 2010 4th Annual IEEE*, 5-8 2010, pp. 267 –271.
[36] Z. Shelby, "Embedded web services," *Wireless Communications, IEEE*, vol. 17, no. 6, pp. 52 –57, 2010.

**Rumen Kyusakov** received a Licentiate degree in Industrial Electronics from Luleå University of Technology, Sweden in 2012 and is currently pursuing a PhD degree in the same university. He is the founder and principal developer of EXIP open source project.

Before joining EISLAB research group as a PhD student in January 2010, he had been working as a software developer in TechnoLogica Ltd., Bulgaria for 3 years. As part of this position, he was developing database and enterprise information systems in several projects for government institutions and banks. He graduated from Sofia University, Bulgaria in 2006 with BSc in Informatics and then in 2008 with MSc in Mechatronics and Robotics.

**Jens Eliasson** is an assistant professor in Computer Science at Luleå University of Technology and holds a Ph.D. in Industrial Electronics from the same university. His main research areas are low-power design of Embedded Internet Systems (EIS) and Internet of Things, wireless sensor networks and Service Oriented Architecture. He is currently involved in several European projects aiming at enabling SOA even on very resource constrained devices.

**Prof. Delsing** received the M.Sc. in Engineering Physics at Lund Institute of Technology, Sweden 1982. In 1988 he received the PhD. degree in Electrical Measurement at the Lund University. During 1985 - 1988 he worked part time at Alfa-Laval - SattControl with development of sensors and measurement technology. In 1994 he got the docent degree (associate prof) in Heat and Power Engineering. Early 1995 he was appointed full professor in Industrial Electronics at Luleå University of Technology where he currently is working as the scientific head of EISLAB, http://www.ltu.se/eislab. For the period 2004-2006 he also served as Dean of the engineering faculty at Luleå University of Technology. Since 1999 he is chairman of ITF (Instrument Tekniska Foreningen/Instrument Society of Sweden).

**Jan van Deventer** was born on September 15, 1965. He received his B.S. in physics from Hope College, Holland, MI, in 1987 and his M.S. in mechanical engineering from the University of Michigan, Ann Arbor, MI in 1989. He received his Ph.D. in Electrical Engineering in 2001 from Lule University of Technology, Luleå, Sweden. From 1991 to 1996, he has worked worldwide with Kelsey-Hayes in the development of anti-locking braking systems. Since 2001, he has been an assistant professor at EISLAB, Luleå University of Technology. His research lies in automotive systems and energy systems.

**Jonas Gustafsson** holds a position as Assistant Professor at Luleå University of Technology. He received his Ph.D. in Industrial Engineering at Luleå University of Technology in May 2011. His research has mainly focused around how district heating systems can be made more efficient by using wireless sensor network technology in a system of system approach.

After receiving his M.Sc. in Electrical Engineering in 2005, Jonas was working with technology for ground penetrating radars in the industry until starting his Ph.D.-studies in 2007.