International Journal of Computer Applications Technology and Research
Volume 2– Issue 2, 180 - 184, 2013, ISSN: 2319–8656

Survey on Service Oriented Architecture to Support the Deployment of Web Services on Sensor Nodes

Vinodhini.J
Department of CSE,
Agni College of Technology,
Chennai, India

Vasantha.R
Department of CSE,
Agni College of Technology,
Chennai, India

Dhivya.M
Department of CSE,
Agni College of Technology,
Chennai, India

W.Mercy
Department of CSE,
Agni College of Technology,
Chennai, India.

Abstract: Service Oriented Architecture (SOA) seamlessly interconnects sensors (embedded devices) inside and between four distinct domains - the business, telecommunication, automotive and home automation domain. In this paper, Simple Object Access Protocol (SOAP)-based web services are directly deployed on the sensor nodes without using any gateways in order to ensure interoperability. This approach provides easy integration with bequest IT systems and supports heterogeneity at the least level.

Keywords: Service-oriented architecture (SOA), simple object access protocol (SOAP).web services, embedded devices (sensors).

---

1. INTRODUCTION

Embedded systems are small, fast, and very great tools, gadgets and equipment which have become part of our everyday life. They monitor and control various physical parameters of the environment as well as communicate the information over the internet. Though the benefits of integrating these devices with business applications are evident, this raises a few technical issues regarding,

(i) interoperability between sensor nodes and business applications,
(ii) heterogeneity of acquired sensor data,
(iii) confidentiality

are presented in the work of Laurent *et al.* [5].

Dealing with the heterogeneity of devices and software systems requires a flexible solution that can lower the complexity and reduce the development, deployment and system maintenance expenses. Service- Oriented Architecture (SOA) has proven successful in controlling these expenses [3]. Also, research study has shown its applicability for embedded systems development [4].

Service oriented architecture (SOA) is computing paradigm that aims to build information system with services as basic units or building blocks. The architectural challenge of SOA can be described as follows.

- ability to access heterogeneous resources(data and others) and services on the web over http protocol
- ability to publish services globally
- ability to make services to discover each other and consume automatically

However, the majority of research studies have been directed towards using middleware software running on more competent devices or gateways as suggested by Wolff *et al*. in [8] or in the work of Bosman*et al*.[9]. Devices Profile for Web Services (DPWS) is implemented on these middleware software systems in order to perform following tasks

(i) Sending secure messages to and from a Web service.
(ii) Dynamically discovering a Web service.
(iii) Describing a Web service.
(iv) Subscribing to, and receiving events from, a Web service.

This approach has the advantage of leaving the resource-intensive tasks to the gateway as described in [2], but also has few drawbacks such as single point of failure, an inability to support heterogeneity on the node level [6], etc. Even though the node-level service implementations have previously 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. In this paper, we present few techniques to improve efficiency that allow us to deploy standard SOAP web services on resource constrained sensor nodes.

1.1 Problem analysis

Problem that an SOA to deeply constrained device such as sensor nodes is still an open research problem since unacceptable overhead is caused by use of Devices Profile for Web Services (DPWS) on the middleware is described by Rumen *et al.* [1]. The unacceptable overhead is due to power consumption, latency, RAM and CPU usage. The Security outline 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, the presented approach is only appropriate for noncritical applications where sensor nodes are behind enterprise firewall.

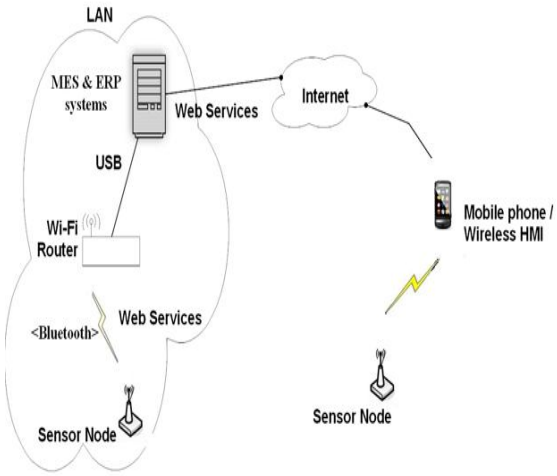

Figure.1 Sensor nodes are incorporated with enterprise systems using standard SOAP-based web services.

The remainder of this paper is organized as follows: Section II describes about the terms and technologies used. Section III summarizes about the related work. Section IV describes about the architecture that connects sensor nodes to an enterprise application. In Section V, the possible improvements and extensions to our work are discussed, and Section VII concludes this paper.

2. RELATED WORK

The analysis performed in research projects such as SOCRADES and within ITEA gives priority to SOAP based web services in which devices are integrated with the IT systems using DPWS.

The work presented by Leguay*et al*. [7] provides 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.

The work presented by Rumen et al [1], usesgSOAP runtime since sensors supports embedded platforms and it includes a highly efficient runtime environment to process SOAP messages.

Another approach more closely related to our paper is that by Priyantha*et al*, at Microsoft Research [11].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.

3. SOCRADES CROSS LAYER APPROACH

SOCRADES (Service-Oriented Cross-layer Infrastructure for Distributed smart Embedded devices) is a European research and innovative development project [10]. A diagram from the SOCRADES Roadmap shown in Fig.2 represents the concept of applying SOA approach for vertical inter- enterprise integration.The goal of the SOCRADES project is to create new procedures, technologies and tools for the modeling, design, execution and operation of networked hardware/software systems embedded in smart physical objects. The use of the SOA paradigm at the device level enables the adoption of a unifying technology for all levels of the enterprise, from sensors and actuators up to enterprise processes. This will lead to information being available "on demand" and allow business applications to use high-level information for such purposes as diagnostics, traceability and performance indicators resulting in increased overall equipment effectiveness and business agility.

3.1 Service-Oriented Architecture

The SOA denotes the usage of exact and self-contained function calls between distributed nodes independent of the locality and platform of the parties involved.Service Oriented Architecture is specific reference architecture that helps solve the data and functionality duplication, thus making the companies that apply this moreflexible, and operate more efficiently.

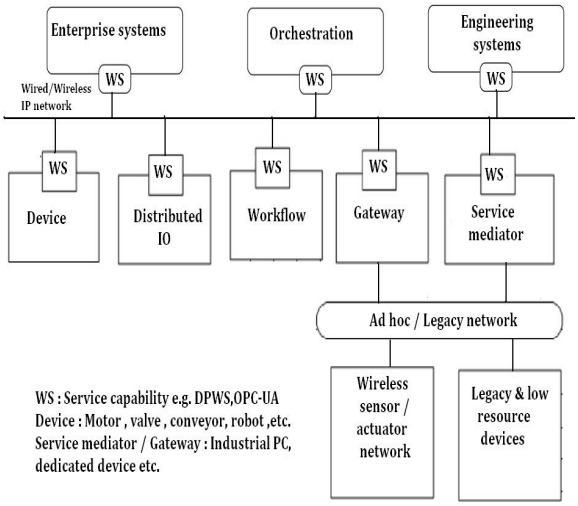

Figure. 2 The SOCRADES cross-layer approach

There are many different concrete implementations of the SOA approach: web services, CORBA, UPnP, OPC-UA, Jini etc. This paper implements SOA using SOAP-Web Services. Based on the characteristic 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 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.

3.2 Web Service

A *web service* is a network accessible interface to application functionality, built using standard Internet technologies. Web services are a messaging framework. The only requirement placed on a web service is that it must be capable of sending and receiving messages using some combination of standard Internet protocols. In general, there are two types of web services: SOAP-based web services and RESTful web services. Simple Object Access Protocol (SOAP) is a XML based messaging protocol that wraps business logic. REpresentational State Transfer (REST) is an application-specific architecture that accesses the resources or data. The web service presented in this paper is built upon SOAP.

3.3 SOAP web service

SOAP web services are designed with a common XML-based protocol. It provides a flexible way for applications to communicate, and forms the basis of SOAP. Because XML is not tied to a particular application, operating system, or programming language, XML messages can be used in all environments. The important drawback of using SOAP-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) which is an alternative mean to represent the XML Information set that provides one-to-one translation to text-based XML representation. The work presented in this paper shows that even verbose XML can be used as a service message protocol for sensor nodes. Introducing the EXI coding to embedded service implementations however, will require the ability to change the XML parser and serializer with EXI ones.

3.4 JAX-WS

Java API for XML Web Services (JAX-WS) is a technology for developing SOAP based and Restful Java Web services. It provides a complete web services stack that eases the task of developing and deploying web services. JAX-WS supports the WS-I Basic Profile 1.1, which ensures that the web services developed using the JAX-WS stack can be consumed by any clients developed in any programming language that adheres to the WS-I Basic Profile standard. The JAX-WS API provides a great environment for writing interoperable SOAP-based Web Services and Web Service Clients. 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).SOAP and XML messaging is a complex domain, but JAX-WS aims to hide the complexity of that domain. JAX-WS 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 tradeoff inherited from this abstraction is a high resource consumption, which makes it not suitable for highly constrained sensor nodes. So, to use the event based "raw" lwIP API, the network layer of JAX-WS runtime was rewritten and additional lwIP wrapper was introduced. This includes the splitting of the sequential execution blocks that contain blocking network operations into smaller nonblocking 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 nonblockingIwIP 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.

4. PROOF-OF-CONCEPT

The technologies discussed in Section II are implemented in a proof-of-concept application that connects sensor nodes to a business application. The overall architecture is depicted in Fig. 3.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.

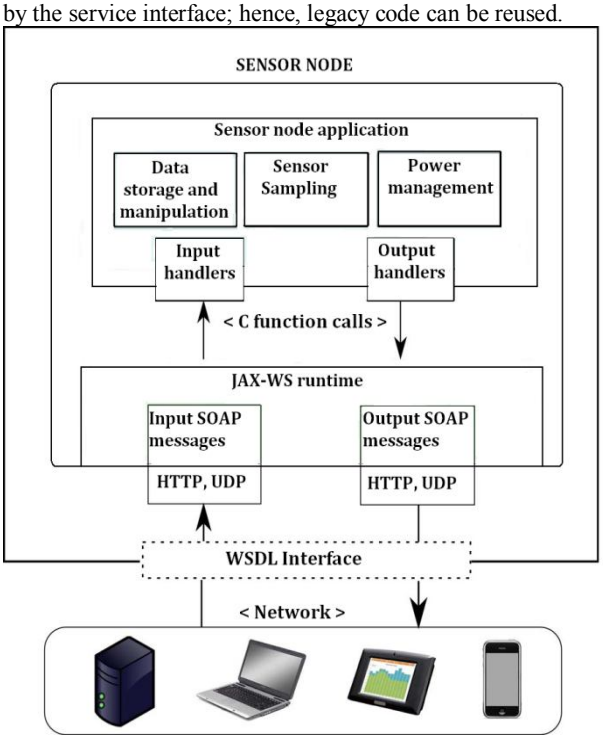

Figure.3 System architecture

Sensor data aggregation is applied for reducing the transmission time and battery life. In [7], Lee et al. used similar approach for industrial monitoring application. 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 JAX-WS runtime using handlers. In [1], Rumen et al. used similar approach using gSOAP runtime. The runtime serializes the output data to a SOAP message, and then uses lwIP to send it over a network. 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- and then are used to generate the SOAP runtime. At the end, the developer connects the provided interface with sensor application.

4.1Mulle Sensor Platform

The Mulle is a miniature wireless Embedded Internet System (EIS) suitable for wireless sensor networking and rapid prototyping. The Mulle platform has ultra-low power consumption and its large number of I/Os makes it ideal as a building block for wireless sensors. The Bluetooth version is capable of communicating with most Bluetooth-enabled devices, e.g. computers, PDAs and mobile phones. The use of TCP/IP enables the Mulle to transmit sensor data directly to the Internet and the small form factor allows it to be easily embedded in any product.

The nodes in the Mulle sensor 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 Fig. 4.

```
        <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>
```

Figure.4 Segment of the service request initiated by the Mulle sensor node. It contains an aggregation of the sensor node for the period of interest.

Proof of concept Experiment

To test the applicability and performance of our solution, several services were implemented. In all cases the interactions between the sensor node and the PC proceeded without any compatibility problems. In this paper, we present two possible scenarios to experiment the project.

1) District Heating Scenario: In today's district heating substations, different sensors and actuators are hard-wired together. This restricts the information interchange between the communicating devices. With wireless sensor platforms integrated in such district heating devices, greater opportunities for system optimization are achieved as information can be interchanged without limitations.

With a Service-oriented architecture integrated in the sensor 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 PC is plugged with a Bluetooth Dongle for communication with sensor. The Swing Application developed on the PC displays the sensor data (temperature, humidity, etc) by making the sensor nodes to invoke the JAX-WS deployed on the server. The nodes are in sleep mode most of the time with short active intervals for sensor sampling and data aggregation.

The implementation started with modeling the desired interactions between the sensors and the business system using Web Service Description Language. The abstract WSDL

service definitions were then fed into Apache Tomcat framework to generate the serialization and parsing code. The same WSDL interface was used by the JAX-WS code generation tools. The code produced was then combined with 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.

2) Mobility Scenario: In this scenario, the sensor node is being carried by a person with Bluetooth-enabled Android mobile phone. The phone provides access to a 3G network that enables connectivity of the sensor node and the Java server on a TCP/IP layer. The Android application installed on the mobile displays the sensor data from the sensor by making it to invoke the web service deployed on the server.

5. CONCLUSION

Integration of high-end systems with deeply embedded sensor nodes enables standard based and direct application-layer integration between web service enabled IT systems and resource constrained sensor nodes. Our future work to apply the same SOA approach to sensor nodes for critical applications where security mechanisms are essential is on research analysis.

6. ACKNOWLEGEMENT

This paper has benefited from conversations with many different people – far more than can be acknowledged completely here. Still we would like to particularly thank Dr.P.S.K.PATRA, HOD, CSE for his guidance and support.

7. REFERENCES

[1] Rumen Kyusakov and Jens Eliasson, "Integration of Wireless Sensor and Actuator Nodes with IT infrastructure using Service-oriented Architecture", IEEE Trans.Industrialinformatics, vol.9, no.1, Feb 2012.

[2] Devices Profile for Web Services Version 1.1, OASIS Std., 2009.[Online] .Available:http://docs.oasis-open.org/wsdd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.pdf

[3] L. D. Xu, "Enterprise systems: State-of-the-art and future trends,"IEEE Trans. Ind. Informat., 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," IEEE Pervasive Comput., vol. 5,no. 3, pp. 94–96, Jul.-Sep. 2006.

[5] Laurent Gomez, Annett Laube and Alessandro Sorniotti,"Design Guidelines for Integration of Wireless Sensor Networks with Enterprise Systems ",IEEETrans,Computer Society,Jun.2007.

[6] G.Moritz, E. Zeeb, F. Golatowski, D. Timmermann, and R. Stoll, "Webservices to improve interoperability of home healthcare devices," inProc. 2rd Int. Conf. Pervasive Comput. Technol. Healthcare, Pervasive Healthcare, 2009, pp. 1–4.

[7] J. Leguay, M. Lopez-Ramos, K. Jean-Marie, and V. Conan, "An efficientservice oriented architecture for heterogeneous and dynamicwireless sensor networks," in Proc.33rd IEEE Conf. Local Comput.Networks, LCN'08, 2008, pp. 740–747.

[8] A.Wolff, S. Michaelis, J. Schmutzler, and C.Wietfeld, "Network-centric middleware for service oriented architectures across heterogeneous embedded systems," in Proc. IEEE 11th Int. EDOC Conf. Workshop,EDOC'07, 15–16, 2007, pp. 105–108.

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

[10] A. Cannata, M. Gerosa, and M. Taisch, "Socrades: A framework for developing intelligent systems in manufacturing," in Proc. Int. Conf.Ind. Eng. Eng. Manage., IEEM'08, 8–11, 2008, pp. 1904–1908.

[11] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, "Tiny web services: Design and implementation of interoperable and evolvablesensor networks," in *Proc. 6th ACM Conf. Embedded Network Sensor Syst., SenSys'08*, New York, NY, USA, 2008, pp. 253–266.