A Web Services based Approach for Resource-Constrained Wireless Sensor Networks

Sana Baccar[1], Mohsen Rouached[2]

[1] CES Research Unit, National school of Engineers of Sfax
Sfax, Tunisia

[2] College of Computers and Information Technology, Taif University
Taif, Saudi Arabia

Abstract

The large diffusion of Wireless Sensor Networks (WSNs) in our contemporary life with their numerous applications has led to a huge heterogeneity. This heterogeneity makes the possibility of discovering and collecting data from the wireless sensors more and more difficult. Indeed, WSNs are currently developed around different communities of sensor and user types, with each community typically relying on its own system, metadata semantics, data format and software. Therefore, the ability to discover and utilize a new sensor asset is typically hindered by incompatible services and encodings which can cause interoperability between different sensor nodes within the same WSN. Service-Oriented-Architecture (SOA) is one of the key paradigms that enables the deployment of services at large-scale over the Internet domain and its integration with WSNs could open new pathways for novel applications and research. Despite the need to integrate SOA with WSNs, only handful efforts are underway to achieve the goal. In this paper, we tackle integration of SOA with WSNs by proposing a Lightweight Representational State Transfer (REST)-based Web Services approach to treat sensors in an interoperable, platform-independent and uniform way.

Keywords: *Wireless Sensor Networks, Service-Oriented Architecture, REST.*

1. Introduction

Sensor technology is continuously improving as the devices become smaller, cheaper, more intelligent, and more power efficient. In consequence, more and more application fields are making use of these technologies [1, 2]. The increasing complexity of device networks consisting of up to thousands of devices is demanding new technologies for simple device interaction and interoperability.

Indeed, sensor networks are currently developed around different communities of sensor types and user types, with each community typically relying on its own stovepipe system for discovery, accessing observations, receiving alerts, and tasking sensor systems and models. Even within fairly coherent communities, each type of sensor tends to be accompanied by its on metadata semantics, its own data formats, and its own software. Thus the ability to discover and utilize a new sensor asset is typically hindered by incompatible encodings and services. There are a number of scenarios which can be presented as test cases for the need of interoperability, for example, a smart home with a set of services like security, energy management, assisted living, etc. A home in this case would have intrusion sensors on doors and windows, smoke sensors in rooms, temperature and light sensors for temperature control and may be fire sensors connected to fire station. Traditionally, each sensor shall be running only one application restricting the generic extensibility of the infrastructure. If we could access all these sensors (and applications) through a common interface, not only we can continue to run the existing applications, but we can also create and run more applications using the same resources.

The necessity, therefore, arises to espouse an interoperability architecture that is open and extensible, and allows for dynamic integration of services. The enabling of an open and extensible architecture requires interoperability at network as well as at application levels. The application layer interoperability poses bigger challenges. Different types of sensors are available, which generate sensor-specific data. The application developer must understand and analyze the message types and parameters used in the sensor nodes. One solution is to adopt a common specification, for all the sensing devices. This approach may work for a small set of devices, but is highly impractical. An alternative approach is to tailor, trim and use existing standard services in a light-weight fashion. SOA is a promising candidate middleware platform that closes this interoperability gap and mediates data exchange between heterogonous sensor platforms and Web applications and services in a unified way. SOA possesses an architectural style encompassing a set of services for building complex systems from existing

components. As an architectural evolution and a paradigm shift in systems integration, SOA enables the discovery, access and sharing of the services, data, computational and communication resources in the network for multiple users. It also allows rapid and cost-effective composition of interoperable and scalable systems based on reusable services exposed by these systems. SOA inherently supports two major requirements: heterogeneous infrastructures and runtime adaptability, which are essential for large-scale cyber-physical systems in which multiple applications run over diverse platforms and adopt different technologies.

The SOA, however, brings with it numerous research and development challenges for use on low power sensor nodes. These challenges range from resource constraints of sensor nodes to the application space of such networks. More details about these challenges are discussed in our previous work presented in [3].

Though SOA has become a cornerstone in many recent research efforts, many of its elegant potentials have not been sufficiently explored in Low power Wireless Personal Area Networks (LoWPANs). In this paper, we tackle integration of SOA with WSNs. More specifically, we propose a Lightweight RESTful Web Services-based approach for Resource-Constrained Wireless Sensor Networks.

The key features of our approach are:

1. Design and implementation of a RESTful API on each sensor node within WSNs in order to guarantee an easy communication between the sensors.
2. Adaptation of the Sensor Web Enablement (SWE) standards [4] to ensure a seamless communication between WSNs and Web-Users via SWE services. This adaptation addresses the challenges related to the data format exchange and the implementation architectural style

The remainder of the paper is structured as follows. Section 2 describes the main concepts needed to understand our contribution. In section 3, we present our approach that aims to design and realize Lightweight RESTful Web Services for Resource-Constrained Wireless Sensor Networks. Section 4 is dedicated to the experimentation and the performance evaluation of the proposed approach. The related work is reviewed in Section 5. Finally, Section 6 concludes the paper and outlines some future directions.

2. Lightweight RESTful Web Services for Resource-Constrained WSNs

2.1 Model ingredients

This section introduces the main ingredients and components of our approach. More precisely, we introduce the SWE standard and the REST technology, which represent the key features of our approach.

2.1.1 Overview of the SWE

The SWE framework consists of a set of standards that define data formats for sensor data and metadata and web service interfaces for providing sensor related functionality. Figure 1 shows an overview of the components forming the SWE architecture.

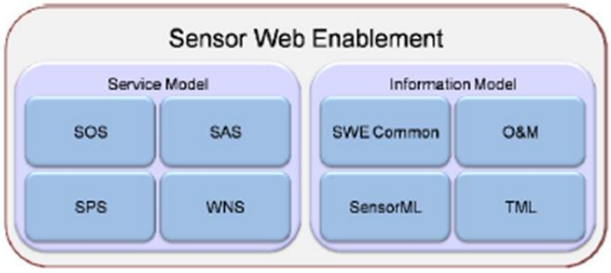

Fig. 1 Overview of the OGC SWE framework

As figure 1 illustrates the SWE framework can be divided into two parts: the interface model and the information model [5].

- The interface model defines the interfaces of sensor related web service types. The SWE information model comprises a set of standards which define data models primarily for the encoding of sensor observations as well as sensor metadata. For this purpose, the first generation of SWE contained three specifications: Observations & Measurements (O&M), the Sensor Model Language (SensorML), and the Transducer Markup Language (TML), which define XML schemas for encoding sensor observations and measurements, describing sensors and processes they can participate in, and describing transducers respectively.

  TML supports the encoding of sensor data as well as metadata by focusing on data streaming. TML has only been rarely used in practice and has not been further evolved so far. In the new generation of SWE specifications, TML is not referenced anymore and recent conversations in OGC's SWE working group showed that there is no urgent

demand in TML and a retirement of the standard is in discussion [5]. Hence, we do not see TML as part of the new generation SWE. The Observations & Measurements standard defines a domain independent, conceptual model for the representation of (spatiotemporal) measurement data. It comprises an implementation of this conceptual model as an XML based GML application schema. The Sensor Model Language specifies a model and XML encoding for the description of all kinds of sensor related processes.

- The information model comprises the standards which address the specification of data formats. The SWE interface model comprises standards that specify the interfaces of the different Sensor web services. Four service interfaces were defined for the first generation of SWE: The Sensor Observation Service (SOS) offers pull-based access to sensor measurements as well as metadata. The Sensor Alert Service (SAS) allows subscribing to alerts in case of a sensor measurement event that fulfills certain criteria. The Sensor Planning Service (SPS) can be used for tasking sensors and setting their parameters. The Web Notification Service (WNS) is, unlike the other three services, not directly sensor related. It is a supportive service which provides asynchronous notification mechanisms between SWE services and clients or other SWE services (e.g., delivery of notifications) including protocol transducing capabilities. The service specifications of the first generation SWE primarily concentrated on the definition of an XML schema which reflected the service functionality. In the new generation SWE, first a conceptual service model is defined (using UML notation), before an XML implementation of that model is specified. This facilitates the adoption of other forms of implementations of the conceptual model such as the JavaScript Object Notation (JSON) implementation.

In summary, the SWE framework provides a comprehensive and stable framework that allows building applications that are based on sensors and sensor data.

2.1.2 REST architectural style

Representational State Transfer (REST) [6, 7] is not just a lightweight instantiation of the web services concept, but it is also an architectural model that is particularly well suited to the properties of smart objects. Systems built around the REST architecture are said to be RESTful.

This powerful architectural model builds on three concepts which are representation, state and transfer:

- Representation: Data or resources are encoded as representations of the data or the resource. These representations are transferred between clients and servers. One example of a representation of a resource is a temperature value written as a decimal number, where the representation is the decimal number and the temperature is the resource. It is possible to have different representations for the same resource which is a powerful concept for the REST architectural style, e.g. a server can serve HTML content for human consumption and XML or JSON for machines.
- State: All of the necessary state needed to complete a request must be provided with the request. The clients and servers are inherently stateless. A client cannot rely on any state to be stored in the server, and the server cannot rely on any state stored in the client. This does not, however, pertain to the data stored by servers or clients, only to the connection state needed to complete transactions.
- Transfer: The representations and the state can be transferred between client and servers.

REST is an architectural model that triggers a good extension to the web services by mapping various Hypertext Transfer Protocol (HTTP) methods and operations to transfer representations of resources between clients and servers with minimum consumption. Moreover, in a REST model, the Uniform Resource Identifiers (URIs) are used to encode transaction states as well as to identify the resource that will be handled. Another powerful concept of the REST is based mainly on its stateless server-design and its semantic content for the exchanged messages. This leads to implement simple and lightweight encodings formatted differently independent of the used standards. In fact, although XML-based formats frequently are used, they are only one of the many available options.
The drawback of the XML based formats are their size. The XML format is verbose and therefore is not suitable for low power and low data rate sensor networks. The data format we use in our work is JSON since it provides a good match for smart object systems where compactness of representation is important due to the inherent resource constraints. Indeed, this data format is more compact than XML as it provides an implicit data structure format without any XML parsing requirements on the network nodes. JSON serializes data structures, such as numbers and arrays, as strings formatted according to the JSON specification.

In the rest of the paper, we outline the adaptation features of the REST architectural style that are broken into two main groups: External RESTful APIs that belongs to the communication between sensor nodes within the same WSN and internal API (node API) that belongs to SWE services running on the sensor nodes themselves.

2.2 A RESTful API for sensors communications

We propose to use a RESTful API on each sensor node within the WSNs to provide information about their sensors and actuators to the central monitoring server as well as to communicate with their neighbors. To achieve this integration, two options are possible: either implementing a RESTful API in a gateway, which connects the sensor nodes with the Internet, or directly implementing the API on Web-enabled devices.
Indeed, the second option looks smarter due to the fact that deploying new device types does not require any changes to the gateway functionality (minimum time-consumption), in addition to guarantying an automatic discovery of sensor node as well as an efficient communication with its neighbors through IPv6 Low power Wireless Personal Area Networks (6LoWPAN) [8, 9, 10].
Therefore, we adopt this solution as depicted in Figure 2.

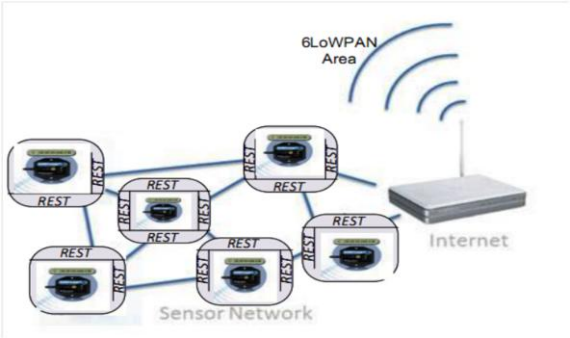

Fig. 2 RESTful sensors communications.

Since each sensor node may connect with different sensors and actuators, manual configuration of a central monitoring system is infeasible if a large number of sensor nodes are used [11]. Based on the RESTful API, [11] introduced a plug-and-play approach, which enables not only the automatic discovery of sensor nodes in a wireless network, but also of the functionality they provide. We propose to extend this plug-and-play approach by ensuring an efficient communication between each sensor and its neighbors through zero user-configuration.
This extension is based mainly on the new way of data transmission between the sensor nodes within the same WSN. In fact, each sensor node resource is described by an URI schema that is like: *ResourceURI::Protocol://Host/ ApplicationPath/ResourceType/ResourceID/ResourceNam e/ResourceObsValue*. In the same cluster, this URI can be devised into two segments: parent segment (*Host/ ApplicationPath/ResourceType/*) and child segment (*ResourceID/ResourceName/ResourceObsValue*). Then, each node can communicate with its neighbors to create, retrieve, update or delete data resource by just identifying their child segment. Moreover, this mechanism is useful for sensor service discovery: once a sensor node is powered, it automatically links to the wireless network and starts to communicate with its neighbors and collect observations that will be available for searching.
The powerful concept of this interface is that it is not fixed to one particular system but it can be accessed from any platform, operating system, and using any network technology. With this RESTful API, the interactions between sensor nodes within the same WSN can be built around standard methods to abstract the inner workings of heterogeneous WSNs. This can reduce pervasive data generated by the huge number of sensor nodes by decreasing transmission load rate of individual sensor nodes and the energy consumption of the overall WSN. One other advantage is that requests and responses between sensor nodes are presented in the JSON format which is a lightweight and platform independent data exchange format. Parsing and creating the JSON representation of data requires less overhead on resource-constrained sensor nodes compared to the XML format.

2.3 Adaptation of the SWE framework to the REST architectural style

The adoption of the REST architectural style for the SWE is depicted in Figure 3. This adoption is based on two features. The first one concerns the interface model. The second one is related to the information model and more specifically to the data encodings.
Each service of the SWE standards (SOS, SAS, SPS, and WNS) can be implemented as a server application that is encapsulated inside a REST interface. This API works as a proxy to this service for providing a RESTful interface to the data.

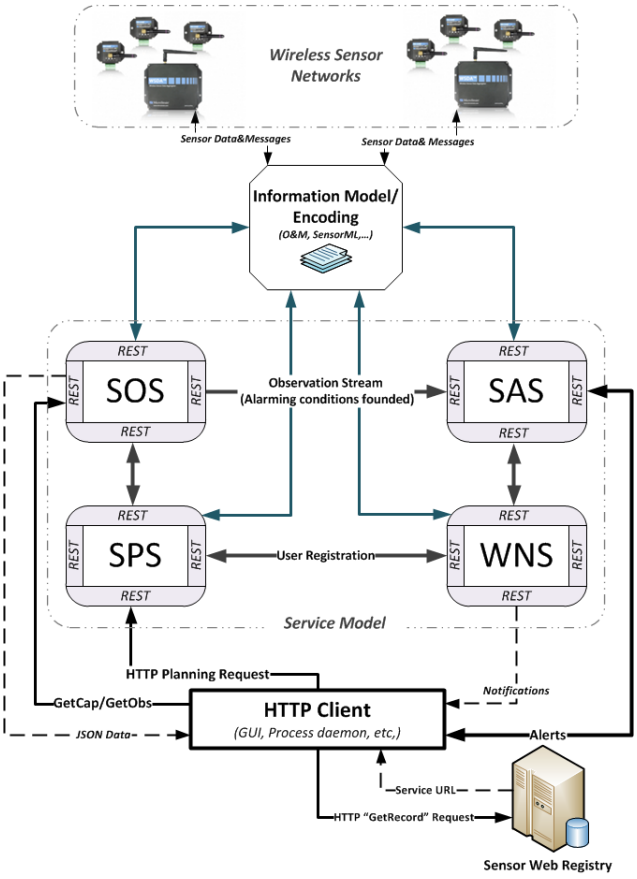

Fig. 3 REST architectural style for SWE.

For instance, Figure 4 shows the design principle of the most important service which is the SOS service. The RESTful-SOS acts as a proxy to the actual SOS and transforms the input calls to SOS queries of the GetObservation operation.

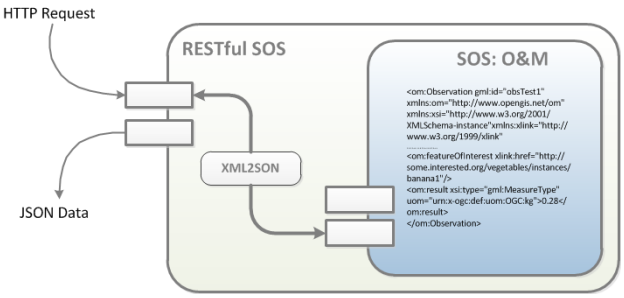

Fig. 4 RESTful-SOS.

The proxy is also able to transform the observations encoded in the Observations and Measurements (O&M) format to JSON data encodings, which is a lightweight and platform independent data exchange format.

Parsing and creating the JSON representation of data requires less overhead on resource constrained sensor nodes compared to the XML format.

As showed in Figure 4, the conversion of XML structures into JSON ones is implemented as an XML2JSON module that accepts XML string data as input and converts that into JSON-formatted data output.

The Web Registry Service (WRS) works as a middleware for connecting the client to the requested service. It is considered as a database server that receives the GetRecords request and returns a JSON document containing the endpoint URL of all existing services that satisfy the query. In a typical scenario, after getting the URL of the service, the client transmits a GetFeasbility request (includes the desired parameters if it is a Post request) to the Sensor Planning Service (SPS). If the task is feasible, the Get/Post observation request (with its parameters and the UserID) will be submitted through the SPS to the mismatched sensors to be executed.

By submitting the request, the SOS starts collecting the sensed observations and stored them. Upon the task is completed, the SPS notifies/alerts the client that the requested data (events) is available (is executed). The client then can get observations/events from the SOS that is used to collect the desired data.

The registration of a new service in the WRS is done as follows. The client sends a request to the WRS containing the service URL to be registered. After registration, the WRS connects to the specified service, fetches its capabilities and starts processing it. When this operation is completed, the WRS notifies the client by sending a notification message to announce that the added service becomes searchable through the WRS interface.

The architecture and the behavior of the XML2JSON module is depicted in figure 6.

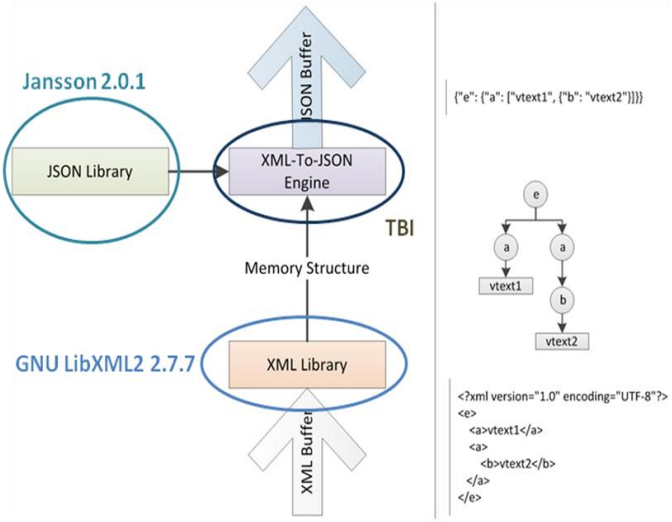

Fig. 6 XML2JSON architecture and behavior

This architecture contains three main components which are:

- XML Library (GNU XML2 library): contains primitives that loads an XML buffer and convert it to a memory structure (Tree of nodes).
- JSON Library (Jansson library): a set of APIs used to create JSON structures.
- XML-to-JSON Engine: combines calls to XML2 library primitives and Jansson library APIs in a specified algorithm to create a structured JSON string buffer from an XML buffer.

Then in order to reduce the size of the JSON output string buffer, we have made an optimization that consists in ignoring all extraneous formatting spaces and tabulations during the creation of a JSON structure.
In summary, unlike some existing parsers that just transform the input file directly to the desired output format, our parsing module contains a filter that tries to eliminate all non useful tags, empty tags, spaces from the input file. Then, the manual generation of JSON objects would be a huge overhead and prone to error, an additional library has been developed to support the reformatting of a response as a JSON object. The JSON library helps to create JSON objects by offering services which automatically generate the JSON envelope. As will be proved in next section, this has an important impact of the parsing time and the size of the output file.
Finally, the performance of this convertor and of the whole model in terms of buffer size and transmission time will be studied in next section.

3. Implementation and Performance study

As a middleware system for building Sensor Web infrastructures based on SWE, we have chosen the 52°North Sensor Web framework [12]. The 52°North Sensor Web framework provides implementations for the different SWE services. An implementation of the Sensor Observation Service enables querying as well as inserting measured sensor data and metadata. Discovery of sensors is supported by implementations of Sensor Instance Registry (SIR) and Sensor Observable Registry (SOR). To integrate sensor resources with the SWE service implementations, the 52°North framework comprises an intermediary layer, called the Sensor Bus [13], to which sensor resources and SWE services can be adapted to establish communication.
Looking from the perspective of the Sensor Web Enablement initiative, different sensors and other support data are required to extract reliable information. This case was the driving force for the SWE Working Group within the 52°North Open Source initiative to come up with an integrated framework named the OGC Web Service Access Framework (OX-Framework) [14, 15]. The aim of the OX-Framework is to provide an integrative view to access all kinds of OGC Web Services.

Based on the RESTful-SOS application presented in *https://svn.52north.org/svn/swe/incubation/OXRestWS/trunk/OX-RestWS/*, the idea of our experimentation scenario is that: after accessing to the RESTful-SOS application and send GetCapabilities request, the returned GetCapabilities response file will be parsed and used to then construct our proper GetObservation or DescribeSensor URLs. Loading these URLs should display the SOS GetObservation response that is returned with in a different format such as in JSON format.
Accessing to one of the SOS instances such as AirBase_SOS enables the client to discover the available resources and services. Such a sensor environment example is described in http://v-swe.uni-muenster.de:8080/52nRESTfulSOS/RESTful/sos/AirBase_SOS/ that we have extended with new RESTful implementation of the other services.

For the benchmarking we have considered two parameters, which are the Data size and the Transmission time. The results of measurements and performance studies show gains up to 60% and 15% in terms of transmission time and buffer size respectively.
To have an idea about the gain in term of data size, Table1 gives an example of seven different XML buffers that was used in our tests.

Table 1: Data buffer size gain (Bytes)

XML	861	1379	2133	2212	3417	12780	25474
JSON	454	889	1063	1146	1452	10300	20552
Gain	407	490	1070	1066	1965	2480	4922

The gain in terms of transmission time is shown in figures 7 and 8. Figure 7 gives a comparison between data transmission duration by considering SOAP and XML, REST and XML, REST and JSON.

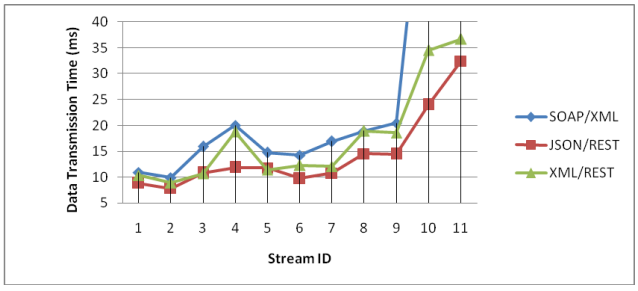

Fig. 7 Data transmission duration -HTTP/REST VS SOAP/XML VS XML/REST.

We can observe a gain up to 60% in term of data transmission duration. It is clear that the reduction of the transmission time is more important by using REST instead of SOAP. Even for REST, with JSON format the results are better than with XML since XML is verbose.

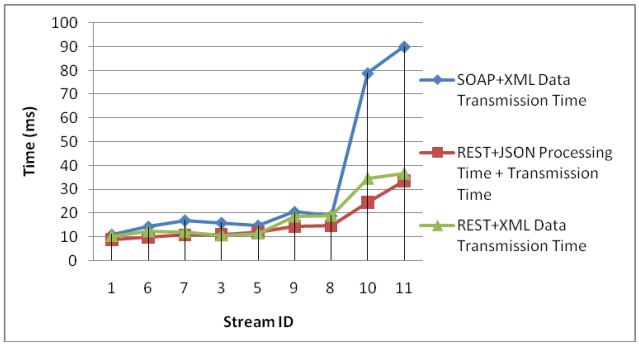

Fig. 8 SOAP/XML Transmission time VS REST/JSON Processing+Transmission time

In figure 8, we consider the transmission time and the processing time together. In this figure, the gain is also important. The processing time is the time for parsing XML files to JSON ones. This time is negligible compared to the transmission time.

4. Related Work

The integration of sensor networks with existing IT systems by providing a structured and interoperable mechanism for data acquisition, data storage, and data replication both within and outside of the sensor network, was discussed in several research papers and projects. Projects such as eSOA[16], SIRENA [17], SOCRADES [18], RUNES [19], and OASiS [20] have been developed in order to provide an SOA approach for embedded networks. The majority of these projects aim at making embedded devices directly accessible with Web Service technologies by installing an adopted Web Service stack, i.e. the Devices Profile for Web Services (DPWS) stack 2 [21]. However, while this approach is suitable for a certain range of devices, there will always be a class of very small and lightweight devices, which cannot deal with the additional overhead introduced by the Web Service technologies, and consequently, require more efficient SOA implementations.

Some works, such as [23, 24], have discussed the adequacy of DPWS for WSNs. These works recommended eliminating the use of SOAP and HTTP protocols due to their high overheads. Instead, they provide some solutions based on application-specific formats that are used in the proposed Tiny DPWS protocol stack. Although the proposed application-specific-format reduces the size of the transmitted messages in the network, it hinders the extensibility of the solutions. For any new service to be offered by sensor nodes, a new application-specific-format should be defined in order to make it work in the proposed infrastructure. [25] describes a SOA based middleware which mediates data exchange between heterogonous sensor platform and Web applications and services in a unified way.

In [26], Moritz et al. presented different XML specific and XML non-specific compressors and their influence on message size of the Devices Profile for Web Services (DPWS). They focused on the SOAP compression to makes DPWS applicable for deeply embedded devices in 6LoWPAN networks, which are characterized by very constrained resources such as small computing power, limited power supply, and a few tens of storage capacity. The results showed that most existing compressors suffer from the simplicity of XML structures, which are the results of non complex services deployed on the deeply embedded device.

To increase parsing performance, a new encoding Devices Profile for Web Services (encDPWS) approach was introduced in [27]. This paper investigated the applicability of DPWS in 6LoWPAN networks. Their main objective is to optimize the message encoding process in order to reduce the overhead of this SOAP-based protocol.

The Open Geospatial Consortiums (OGC) [15] is an international standardization consortium, which provides a framework that specifies standard interfaces to access geographical data in addition to encoding and exchanging these data over the Internet.

OGC Web Services follow the W3C's service-oriented web services framework and support publishing, automatically discovering and accessing geographical information over the web; leading to Spatial Data Infrastructures (SDI). In addition to these standardization efforts, there have been several proprietary solutions that illustrated the Sensor Web concept.

For instance, SensorMap [28], a Microsoft project, provides a set of tools that data owners can use to easily publish their environmental data, and a GUI enabling users to make queries over live data. SensorMap transparently provides mechanisms to archive, index and aggregate sensory data, and process queries [29]. The SensorMap GUI is a mash-up application that permits users to submit queries on available sensors and overlays the aggregated results on a map. The framework introduced in [30] facilitates access to both real-time and historical sensed data, through variety of access methods. It addresses the scalability issue by introducing a distributed sensor register. Although these solutions provide either an SOA based APIs or common interfaces that make sensing data

accessible for the users, their operational behavior is very much influenced by the role of the heavy application-level gateway and single-point-of-failure problem. The application-level-gateway plays a crucial role due to the absence of direct interaction among sensor nodes, the Internet applications, and users.

Some middleware systems for building Sensor Web infrastructures based on SWE are proposed. GeoSwift, another SOA-based framework that was proposed by Liang et al. in [31], is a distributed geospatial information infrastructure for the Sensor Web.

The PULSENet framework [32], which reuses and amends the open source components of the 52°North Sensor Web framework, allows the implementation of a SWE-based Sensor Web. NASA's Sensor Web 2.0 [33] system incorporates SWE services and combines them with Web 2.0 technology. The mash-up functionality is realized by incorporating the representational state transfer (REST) approach to access data. However, it remains unclear how the system provides REST access to sensor resources by leveraging SWE services.

IrisNet (Internet-scale Resource-Intensive Sensor Network Services) [34] and Tenet [35] are two other approaches that have adopted SOA in developing middleware solutions for WSNs. Tenet provides a SOA-based solution that is although its flexible accommodations for some applications, still heavily relying on the application-level gateway that plays an important role in the Tenet solution. On the other hand, IrisNet is a distributed software architecture, which provides high-level sensor services to users. An important characteristic of this architecture is that Sensor Agents are dynamically programmable.

Although these solutions expose WSN to be more accessible through the Internet by means of application-level gateways, they mainly suffer from the single-point-of-failure problem and scalability issues common to centralized gateway approaches.

Several other efforts aimed at using SOA in WSNs. In what follows, we present an overview of most relevant works.

In [36, 37], the authors addressed the feasibility of using RESTful Web services to integrate SOA with IP-based WSNs. In [37], the authors presented an approach to integrate tiny wireless sensor or actuator nodes into an IP-based network. Sensors and actuators are represented as resources of the corresponding node and are made accessible using a RESTful web service. Sensor nodes run a small web server on top of a TCP/IP stack to provide access to sensor data and actuators using HTTP requests. Data is represented in the JSON format, which is a more lightweight alternative as compared to XML. A prototype application based on TinyOS 2.1 on a custom sensor node platform with 8 Kbytes of RAM and an IEEE 802.15.4 compliant radio transceiver was implemented. A key feature in this approach is that compared to many existing approaches that provide Web services at a smart gateway, it proves the feasibility to provide Web services at each node, even when using a very resource-constrained hardware platform. The system explained in [38] uses two mechanisms to provide a good performance and low-power operation: a session-aware power-saving radio protocol and the use of the HTTP Conditional GET mechanism.

In [39], Rezgui and Eltoweissy explored the potential of SOA in building open, efficient, interoperable, scalable, and application-aware Wireless Sensor and Actuator Networks (WSANs). A prototype of service-oriented WSAN was developed using TinyOS. In [40], King et al. developed a service-oriented WSAN platform, called Atlas, which enables self-integrative, programmable pervasive spaces. Kushwaha et al. developed in [41] a programming framework, called OASiS, which provides abstractions for object-centric, ambient-aware, service oriented sensor network applications. OASiS decomposes specified application behaviors and generates the appropriate node-level code for deployment onto sensor networks. It enables the development of WSN applications without having to deal with the complexity and unpredictability of low-level system and network issues. In [42], Golatowski et al. proposed a service oriented software architecture for mobile sensor networks. An adaptive middleware is employed in the architecture that encompasses mechanisms for cooperative data mining, self-organization, networking, and energy optimization to build higher-level service structures.

In [43], the authors presented an approach to seamlessly integrate WSNs into business process (i.e. SOA) environments using the Business Process Execution Language (BPEL) and Web Services while using only very few resources on the sensor nodes. It introduces how application developers can use standard compliant techniques to describe business processes that are using services offered by WSNs, without the need for hand-crafted code for data conversion, etc. By adopting this approach, services offered by the WSN can be used seamlessly in enterprise-level business processes.

These services can also quickly be composed to higher-level applications by simply modifying the business process. Priyantha et al. described in [44] a Web Service-based approach based on standard technologies such as IPv6, 6LoWPAN, and HTTP. Considering the message serialization and transport, Web Service messages are exchanged using HTTP. In their approach, they tried to avoid using complex SOAP messages as much as possible. Instead, if Web Service messages do not contain complex data structures, simple URL encoded messages are exchanged to reduce the message size. In [44], Amundson et al. presented a SOA based approach for WSNs not

relying on Web services. Thus, to enable sensor nodes calling Web Services of Enterprise-IT systems, their solution imposes the use of a gateway, which converts between the proprietary middleware message format and standard Web Service message format.

5. Conclusion and future direction

In this paper, we have presented a RESTful implementation for the SWE. This implementation consists of developing REST interfaces for each SWE service and using the JSON data format for messages exchanges between services and sensors environments. Then, the results and measurements have showed the effectiveness of this adaptation in terms of file size reduction, and communication and response times.

The approach presented here can be considered as a first step towards the work on SWE. Several open challenges and future work in this context can be outlined.

Among these challenges, we are more interested in the improvement of interoperability, the facilitation of sensor and service integration, and the enablement of the Semantic Sensor Web.

Acknowledgments

The authors would like to thank Dr. Anis Koubaa for the useful feedback during the early stages of this work and for the several valuable discussions in terms of the Wireless Sensors Networks field.

References

[1] A. Sleman, and R. Moeller, “Integration of Wireless Sensor Network Services into other Home and Industrial networks; using Device Profile for Web Services (DPWS)”, Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference, 2008, pages 1-5.

[2] I. K. Samaras, J. V. Gialelis, and G. D. Hassapis, “ Integrating Wireless Sensor Networks into Enterprise Information Systems by Using Web Services”, In SENSORCOMM '09: Proceedings of the 2009 Third International Conference on Sensor Technologies and Applications, pages 580{587, Washington, DC, USA, 2009. IEEE Computer Society.

[3] M. Rouached, S. Chaudhry, and A. Koubaa. “Lowpans meet service-oriented architecture”. JUSPN, 1(1):39{48, 2010.

[4] Open geospatial consortium. http://www.opengeospatial.org/

[5] A. Brring, A. Broering, J. Echterhoff, S. Jirka, I. Simonis, Th. Everding, Ch. Stasch, S. Liang, and R. Lemmens. “New generation Sensor Web Enablement”. Sensors, 11(3):2652–2699, 2011.

[6] C. Pautasso and E. Wilde, “Restful web services: principles, patterns, emerging technologies”, In WWW, pages 1359{1360, 2010.

[7] C. Pautasso, O. Zimmermann, and F. Leymann, “RESTFUL Web Services vs. "BIG" Web Services: Making the Right Architectural Decision”. In WWW, pages 805{814, 2008.

[8] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, Transmission of IPv6 Packets over IEEE 802.15.4 Networks RFC 4944”, IETF, 2007.

[9] G. Mulligan, “ The 6lowpan architecture”, In EmNets’07, pages 78–82, 2007.

[10] Z. Shelby and C. Bormann, “6LoWPAN: the wireless embedded internet. Wiley Series on Communications Networking & Distributed Systems”. J. Wiley, 2010.

[11] L. Schor, P. Sommer, and R. Wattenhofer, “Towards a zero-configuration wireless sensor network architecture for smart buildings”, In Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, BuildSys '09, pages 31{36, New York, NY, USA, 2009. ACM.

[12] 52north initiative for geospatial open source software gmbh. http://52north.org/.

[13] T. Foerster, S. Jirka, and C. Priess, “An intermediary layer for linking sensor networks and the sensor web”.

[14] A. Broering and T. Foerster, “An integrated software framework for OGC web services.” In Foss4g2006, Lausanne, Switzerland, Sept. 2006.

[15] Development of Sensor Web Applications with Open Source Software, 2009.

[16] A. Scholz, I. Gaponova, S. Sommer, A. Kemper, A. Knoll, C. Buckl, J. Heuer, and A. Schmitt, “eSOA - Service-Oriented Architectures Adapted for Embedded Networks. In Proceedings of 7th IEEE International Conference on Industrial Informatics”, pages 599 {605, June 2009.

[17] F. Jammes and H. Smit, “Service-Oriented Paradigms in Industrial Automation”, Industrial Informatics, IEEE Transactions on, 1(1):62{70, April 2005.

[18] L. M. S. de Souza, P. Spiess, D. Guinard, M. Khler, S. Karnouskos, and D. Savio, “SOCRADES: A Web Service Based Shop Floor Integration Infrastructure”, In IOT, volume 4952 of Lecture Notes in Computer Science, pages 50{67. Springer, 2008.

[19] P. Costa, G. Coulson, and C. Mascolo, “The runes Middleware: A Recon_gurable Component-Based Approach to Networked Embedded Systems”. In Proc. of 16 th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC05, pages 11{14. IEEE Press, 2005.

[20] M. Kushwaha, I. E. Amundson, X. Koutsoukos, S. Neema, and J. Sztipanovits. “OASiS: A Programming Framework for Service-Oriented Sensor Networks”. In IEEE/Create-Net COMSWARE 2007, January 2007.

[21] Ioakeim K Samaras, John V Gialelis, and George D Hassapis, “Integrating wireless sensor networks into enterprise information systems by using web services”, 2009 Third International Conference on Sensor Technologies and Applications, (Xml):580–587, 2009.

[23] F. Jammes, A. Mensch, and H. Smit, “Service-oriented device communications using the devices profile for web

services", In Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing, MPAC '05, pages 1–8, New York, NY, USA, 2005. ACM.

[24] G. Moritz, E. Zeeb, S. Pr□uter, F. Golatowski, D. Timmermann, and R. Stoll. "Devices Profile for Web services in Wireless Sensor Networks: Adaptations and Enhancements". In ETFA'09: Proceedings of the 14th IEEE international conference on Emerging technologies & factory automation, pages 43{50, Piscataway, NJ, USA, 2009. IEEE Press.

[25] H. Abangar, P. Barnaghi, K. Moessner, A. Nnaemego, K. Balaskandan, and R. Tafazolli, "A Service Oriented Middleware Architecture for Wireless Sensor Networks", In Proceedings of the Future Network & MobileSummit 2010 Conference, 2010.

[26] G. Moritz, D. Timmermann, R. Stoll, and F. Golatowski, "Encoding and Compression for the Devices Pro_le for Web Services", In AINA Workshops, pages 514{519, 2010.

[27] G. Moritz, D. Timmermann, R. Stoll, and F. Golatowski. "encDPWS - Message Encoding of SOAP Web Services", In PerCom Workshops, pages 784{787, 2010.

[28] S. Nath, J. Liu, and F. Zhao, "SensorMap for WideArea Sensor Webs", Computer, 40(7):90{93, 2007.

[29] A. Santanche, S. Nath, J. Liu, B. Priyantha, and F. Zhao, "SenseWeb: Browsing the Physical World in Real time", In Demo Abstract, April 2006.

[30] S. H. L. Liang, A. Croitoru, and C. V. Tao, "A Distributed Geospatial Infrastructure for Sensor Web", Comput. Geosci.31(2):221{231, 2005.

[31] H. M. I. Rhead, M. Merabti and P. Fergus, "Worldwide Sensor Web Framework Overview", In Proceedings of the 9th Annual Postgraduate Symposium, The Convergence of Telecommunications, Networking and Broadcasting, 2008.

[32] S. M. Fairgrieve, J. A. Makuch, and S. R. Falke, "Pulsenet: An implementation of sensor web standards", In Proceedings of the 2009 International Symposium on Collaborative Technologies and Systems, pages 64{75, Washington, DC, USA, 2009. IEEE Computer Society.

[33] Y. Liu, L. Marini, R. Kooper, A. Rodriguez, D. Hill, J. Myers, and B. Minsker, "Virtual sensors in a web 2.0 virtual watershed". In Proceedings of the 2008 Fourth IEEE International Conference on eScience, pages 386{387, Washington, DC, USA, 2008. IEEE Computer Society.

[34] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan, "IrisNet: An Architecture for a Worldwide Sensor Web", IEEE Pervasive Computing, 2:22{33, 2003.

[35] J. Paek, B. Greenstein, O. Gnawali, K.-Y. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, and E. Kohler, "The Tenet Achitecture for Tiered Sensor Networks", ACM Trans. Sen. Netw.6(4):1{44, 2010.

[36] L. Schor, P. Sommer, and R. Wattenhofer, "Towards a Zero-Configuration Wireless Sensor Network Architecture for Smart Buildings", In BuildSys '09: Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy- Efficiency in Buildings, pages 31{36, New York, NY, USA, 2009. ACM.

[37] D. Yazar and A. Dunkels, "Efficient Application Integration in IP-based Sensor Networks", In BuildSys '09: Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy- Efficiency in Buildings, pages 43{48, New York, NY, USA, 2009. ACM

[38] A. Rezgui and M. Eltoweissy, "Service-Oriented Sensor-Actuator Networks: Promises, Challenges, and the Road Ahead", Computer Communications, 30(13):2627{2648, 2007.

[39] J. King, R. Bose, null Hen-I Yang, S. Pickles, and A. Helal, "Atlas: A service-oriented sensor platform: Hardware and middleware to enable programmable pervasive spaces", Local Computer Networks, Annual IEEE Conference on, 0:630{638, 2006.

[40] M. Kushwaha, I. Amundson, X. Koutsoukos, S. Neema, and J. Sztipanovits. "OASiS: A Programming Framework for Service-Oriented Sensor Networks", In IEEE/Create-Net COMSWARE 2007, 2007.

[41] F. Golatowski, J. Blumenthal, M. H, M. Haase, H. Burchardt, and D. Timmermann, "Service Oriented Software Architecture for Sensor Networks", In Proc. Int. Workshop on Mobile Computing (IMC03, pages 93-98, 2003).

[42] N. Glombitza, D. P_sterer, and S. Fischer, "Integrating Wireless Sensor Networks into Web Service-Based Business Processes", In MidSens '09: Proceedings of the 4th International Workshop on Middleware Tools, Services and Run-Time Sup- port for Sensor Networks, pages 25{30, New York, NY, USA, 2009, ACM.

[43] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, "Tiny Weeb Services: Design and Implementation of Interoperable and Evolvable Sensor Networks", In SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems, pages 253{266, New York, NY, USA, 2008, ACM.

[44] I. Amundson, M. Kushwaha, X. Koutsoukos, S. Neema, and J. Sztipanovits, "Efficient Integration of Web Services in Ambient-Aware Sensor Network Applications", In 3rd IEEE/CreateNet International Workshop on Broadband Advanced Sensor Networks (BaseNets 2006), October 2006.

Sana Baccar is a PhD student in the Computer and Embedded Systems (CES) research lab at the National School of Engineering of Sfax, Tunisia. Her research interests include Service Oriented Computing, Wireless Sensors Networks, and Semantic Web.

Mohsen Rouached is currently acting as an assistant professor in the College of Computers and Information Technology at Taif University. He received his M.S and Ph.D in computer science from Nancy University in 2005 and 2008 respectively. His research interests span over several areas related to Service Oriented Computing, Business Processes, Security, Privacy, and Forensics Management, Services Semantics, and Wireless Sensors Networks. He has published over 40 research papers in these domains. He serves as program committee member and reviewer at many international journals and conferences and has been participating in several research projects.