2010 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems

Wireless Sensor Networks as a Service

Flávia C. Delicato[1], Paulo F. Pires[1], Luci Pirmez[2], Thais Batista[1]
[1]*Dimap – Federal University of Rio Grande do Norte (UFRN)*
[2]*NCE/DCC-IM – Federal University of Rio de Janeiro (UFRJ)*
PO Box 2324 – 20.001-970 – Rio de Janeiro – RJ – Brasil
{fdelicato, paulo.f.pires, luci.pirmez, thaisbatista }@ gmail.com

Abstract

One major reason for the increasing interest in wireless sensor networks (WSN) in the last few years is their potential usage in a wide range of application domains. However, there is a set of challenges to be addressed in order to realize the true potential of current WSNs. The first challenge concerns the effort needed to develop WSN application since the developers are required to know several sensor network and protocol specific details. The second main challenge *is related to the fact* that the most of existent WSNs provide data in proprietary formats that can be accessed by final user only through a set of static predefined queries or a graphical interface, hindering the widespread use of WSN data in different applications. The third challenge concerns the semantic gap between the representation of the high-level application requirements and the low-level data provided by sensor nodes. We argue that, by adopting a SOA approach based on integrating WSN to the concept of web mashups we will be able to leverage the widespread use of WSNs.

1. Introduction

Recent technological advances have enabled the development of low-cost, low-power and multifunctional sensor nodes. These nodes are autonomous devices, often battery-powered, with integrated sensing, processing and wireless communication capabilities. A sensor is an electronic device able of detecting environmental data such as temperature, sound, light, movement, among others. The sensing device measures parameters from the environment surrounding the sensor node and converts them into electric signals. Properties of objects located and/or events happening in the vicinity of the sensor can be detected by processing such signals [20]. Sensor nodes send their sensed data, usually via a short range radio transmitter, to a data-collection station (a sink node). Typically a sink node includes software for sophisticated processing of data collected by sensors and is often connected to external networks.

A Wireless Sensor Network (WSN) is composed of a large number of sensor nodes, which are densely deployed either inside the monitored phenomenon or very close to it, and one or more sink nodes. Typically, sensors are deployed in an ad-hoc fashion and communicate through low bandwidth wireless links. Sensor nodes have to operate unattended, since it is unviable to service a large number of nodes in remote, possibly inaccessible locations. Therefore, energy saving is a crucial requirement in such environment. WSNs can play the role of a highly parallel, accurate and reliable data acquisition system.

Data transmission in the wireless media is the main source of energy consumption. For purposes of energy saving, the data reports are often sent to the sink node through a multihop short-distance communication, with intermediary sensor nodes forwarding their own and neighbors data. Intermediary sensor nodes and the sink node can perform operations of fusion and/or aggregation of the sensed data with the aims of filtering out erroneous data and anomalies, drawing conclusions from the reported data over a period of time, besides further saving energy [20].

One major reason for the increasing interest in wireless sensor networks in the last few years is their potential usage in a wide range of application areas such as civil engineering, health, military, habitat monitoring and security among others.

The first works in the area [14,10,16] considered that each sensor network would be designed for one specific target application. Nowadays, considering that WSNs can be potentially useful for a wide range of application domains and that the sensor network infrastructure is expensive, there is a strong trend in designing commercial-scale WSNs as being composed of heterogeneous sensor devices and assisting to a large range of applications for different groups of users. However, there is a set of challenges to be overcome in order to realize the true potential of current WSNs.

The first challenge concerns the effort needed to develop WSN applications. In the development of

978-0-7695-4005-4/10 $26.00 © 2010 IEEE
DOI 10.1109/ECBS.2010.54

applications for current WSN platforms the developers are required to know several sensor network and protocol specific details and build programs either by using the low level abstractions provided by the WSN operating system (for instance, nesC language provided by TinyOS system [17]) or directly over the sensor hardware (for instance, in the Sun Spot platform [24]).

The second main challenge regards the extraction and use of the sensor generated data. Most of existent WSNs provide data in proprietary formats that can be accessed by final user only through a set of static predefined queries or a graphical interface. Such approach constrain s the use of the data reported by sensor nodes to the predefined access formats and queries, hindering the widespread use of WSN data in different applications.

The third challenge is related to both the aforementioned ones, and concerns the semantic gap between the representation of high-level application requirements and the low-level data provided by sensor nodes. Application needs usually have high-level descriptions such as "report the detection of any 10 tons four-legged animal in region X" while individual sensor nodes typically provide individual raw data, accessed through very simple and low level APIs. Therefore, further processing is required to convert raw data reported by sensor nodes and represented in the primitive and low level formats provided by the networks to useful information to the applications.

In short, we are interested in answer the following question: given the huge number of devices monitoring the physical world already deployed and available for usage, how to enable people to create applications on top of such WSN systems? To achieve this goal, a crucial requirement is to provide a layer of abstraction to issue sensing tasks and queries to the WSN and to gather the sensor generated data.

We propose two main ideas to tackle the referred current challenges to leverage the use of WSNs. The first idea is to adopt a service oriented (SOA) approach to design WSNs. We argue that services consist in a suitable abstraction for developing WSN applications and the XML is a suitable format for data representation and exchange among applications and the network. The second idea is to integrating WSN and Mashups, since mashups represent a suitable example of easily building new applications on top of a virtual ecosystem of services and there are lots of many available tools to develop such applications.

In our proposal, sensor nodes act as data providers and each WSN as a whole acts as service provider for client applications. The provided services are: (i) raw data generated by sensor nodes, (ii) processed data, generated through several types of analysis, filtering and complex processing, and (iii) value added services provided by web mashups. We argue that, by adopting an SOA approach based on integrating WSN to the concept of web mashups we will be able to leverage the widespread use of WSNs.

This paper is organized as follows. Section 2 provides a brief description of concepts on WSN and Web Mashups. Section 3 presents our proposal, including the architectural layers of the system and the roles played by such components according to the SOA pattern, as well as details about WS* languages, protocols and approaches adopted for services description and data communication in our proposal. Section 4 concludes the paper presenting a discussion of related works, future works and final remarks.

2. Background

This section presents background concepts about WSNs and Web Mashups needed for the comprehension of the remaining of the paper.

2.1. Wireless Sensor Networks

Wireless sensor networks represent an increasingly important example of distributed event systems. Most of these networks work as a reliable data capture network. Data are collected in the distributed sensors and relayed to a small number of exit points, called sinks, for further processing and forwarding to the final user (or applications). Since energy saving is a crucial requirement for the battery operated sensors, the short range hop-by-hop communication is preferred over direct long-range communication to the destination. Therefore, the dissemination of information is done by nodes performing measurements and forwarding data through neighboring nodes to reach a sink node in the network. Data sent by different nodes can be aggregated in order to reduce redundancy and minimize the data traffic thus additionally saving energy. To enable data aggregation in network in an efficient way, application-specific code, such as data caching and collaborative signal processing should occurs as close as possible to where data is collected. Such a processing depends on attribute-identified data to trigger application-specific code and hop-by-hop processing of data [10].

WSN can be classified in proactive and reactive networks, according to the class of the target application. In proactive WSNs, nodes periodically (in a pre-defined interval) sense the environment and transmit data of interest. In reactive WSNs, nodes react immediately to sudden and drastic changes in the value of a sensed attribute. These classes of WSN are well suited for time critical applications. Once the type of

network is defined for a given application, protocols that efficiently route data from source nodes to sinks have to be used. Several WSN specific protocols were proposed in the last years [2,10,11,14,15,16,18], each one focusing a different type of application domain and/or fulfilling different parameters of Quality of Service (QoS). Examples of QoS parameters relevant to the context of WSNs are data delay and accuracy, network coverage and lifetime.

Most of WSN protocols rely on localized algorithms and data-centric communication, besides to exploit application-specific knowledge in the data dissemination. Localized algorithms are a special kind of distributed algorithms that achieve a global goal by communicating with nodes in a restricted neighborhood. Such algorithms scale well with increasing network size and are robust to network partitions and failures. Data-centric communication introduces a new style of addressing in which nodes are addressed by the attributes of data they generate (sensor type) and by their geographical location, instead of by their network topological location. Finally, the use of application knowledge in nodes can significantly improve the resource and energy efficiency, for example by application-specific data caching and aggregation in intermediate nodes [10].

Regardless the specific protocol adopted, all protocols depend on mechanisms for representing the application queries and interests as well as generated sensor data, and for triggering application-specific processing when pre-defined types of data or events are sensed. Data-centric protocols represent queries and data through high level descriptions (meta-data) and disseminate such descriptions in the network instead of the collected raw data. When a cluster-based approach is adopted, a further mechanism for representation of coordination messages exchanged among nodes is needed.

2.2. Web Mashups

A Mashup has the ability to create dynamic, user-centric solutions through the combined functionality or content from different and possibly unrelated existing sources. These existing sources can be SOAP or REST Web Services, RSS feeds or even just other Websites (in this case the data should be extracted by screen-scraping). Mashups can be seen as a composition technology, since its ultimate goal is to make possible the creation of new services and applications starting from the integration of different sources available on the Web. Compared to other web services enabled composition technologies, such as BPM [21] or Enterprise service bus (ESB) [3], mashups propose more agile and user-centric composition process. Such agility is achieved through the use of Web 2.0 related technology. Web 2.0 leverage WebPages from static HTML documents to more dynamic and interactive data application that can be easily consumed by different types of clients.

According to Merrill [19], a mashup application is architecturally comprised of three different components: content providers, the mashup site, and the Web browser.

The **content providers** are the origin of the information being mashed. A key issue for content providers is to facilitate data retrieval. This issue can be addressed exposing the content to be mashed through standard Web protocols such as REST, Web Services, and Atom.

The **mashup site** hosts the mashup logic. Mashups can be implemented similarly to traditional Web applications using server-side dynamic content generation technologies like Java servlets, CGI, PHP or ASP. Moreover, the mashed content can be generated within the client web browser through web browse scripting (for example using JavaScript). This client-side mashup logic is often the combination of code directly embedded in the mashup's Web pages as well as scripting API libraries or applets (furnished by the content providers) referenced by these Web pages.

The **client Web browser** is where the Mashup application is graphically rendered and also the place where the client-side mashup logic runs.

Frequently mashups use a combination of both server and client-side logic to perform data aggregation/composition. Data composition that requires powerful computations such as complex queries on multiple-sourced data are better performed by server-side mashups while local small sets of data can be mashed by client-side mashups.

3. Proposal: the WSN as a Service

Our work proposes a generic and flexible architecture for sensor networks based on the Web services and Web Mashup technologies. Web services are built according to the service-oriented architecture (SOA pattern) and they can be described by a trio of interoperability stacks [8]. In the proposed architecture, sensor nodes act as data providers and the WSN acts as service provider for client applications, providing: (i) raw data generated by sensor nodes, (ii) processed data, generated through several types of analysis, filtering and complex processing, and (iii) value added services provided by web mashups.

Section 3.1 describes the WSN physical components considered in this work. Sections 3.2 and

3.3 present the architectural layers of the proposed system and the roles played for such components according to the SOA pattern. Sections 3.4 and 3.5 details WS* languages, protocols and approaches adopted for services description and data communication in our proposal.

3.1 WSN Physical Components

In our proposal, we consider a WSN comprising of two main physical components: sensor nodes and sink nodes. The architectural components of the proposed system are present in both sink and sensor nodes, above network level protocols and operating system.

A **sensor node** can contain one or more specialized sensing devices. Furthermore, it can have routing and data aggregation capabilities. Thus, the routing function is distributed among all nodes. Concerning data aggregation capabilities, we assume that all the sensor nodes have enough processing and storage capacities to store and execute aggregation programs.

Sink nodes provide an interface through which the other architectural components (regarding mashups technology) of the proposed system can obtain the information collected by the sensor network. Such interfaces can be accessed locally or remotely (i.e., through the Internet). Sink nodes can also aggregate data, but they do not have sensor devices. We assume that they are more powerful regarding to processing and communication capabilities than sensor nodes.

3.2 Architecture Layers

The proposed system adopts a three layered architecture composed of: (i) Data Provision Layer, (ii) Data Extraction and Interoperability Layer, and (iii) Composition Layer (Figure 1). The ***Data Provision layer*** is physically composed of sensor nodes which are responsible for gathering the environmental raw data and performing simple data processing such as computation of averages, min/max, etc. The next layer, ***Data Extraction and Interoperability Layer***, is physically composed of sink nodes. This layer is responsible for extracting sensor generated data, possibly from different WSNs, and providing a common interface for accessing such data. Moreover, since sinks are powerful devices, sophisticated post-processing can be performed over the extracted data. The ***Composition Layer*** consists of Web Mashups. This layer provides value-added services through the composition of data extracted from different WSNs using the sink node common interface. Such mashups allow different levels of both visualization and processing of a WSN ecosystem. For illustration purpose, suppose two different WSNs. The WSN_1 provides temperature data of a geographic area A, while WSN_2 provides humidity and light data of a geographic area B. We should also consider that there is an overlapping area C between A and B. Now, consider a user interested in continuously monitoring the humidity data and in obtaining the average temperature of a region geographically located inside area C. Besides integrating data from WSNs 1 and 2, a Web Mashup allows easy spatial visualization of the integrated data, for instance, by using Google Map service [7].

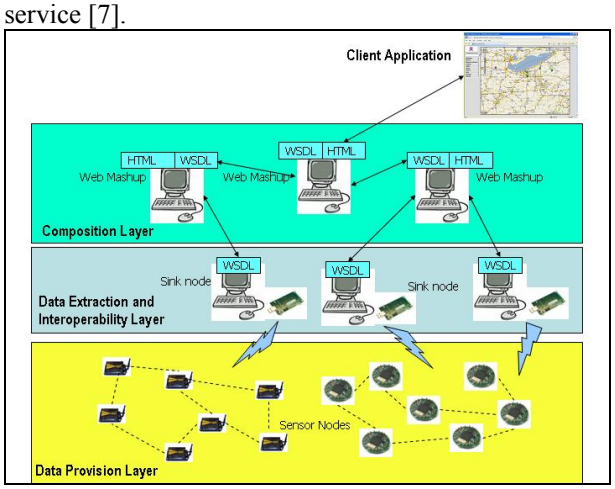

Figure 1. System Architecture Layers

3.3 System Components According to the Service-Oriented Architecture Pattern

The proposed system is based on the concept of service-oriented architecture (SOA) [8] (Figure 2). A client application or a Web Mashup interested in obtaining sensor data plays the role of **service requestor** of other Web Mashups. At the same time, Web Mashups act as service providers to client applications and other Web Mashups. This behavior of playing two different roles occurs since each mashup provides a different type of information and/or visualization obtained from different data composition and transformation. By its turn a given mashup composition can be built from other existent mashups. Web Mashups are published and discovered using the Mashup Catalogue that plays the role of service **registry**. The Mashup Catalogue can be implemented as a regular UDDI register [25].

Sink nodes act primarily as **service providers** to the external environment (in our case, Web Mashups). They expose the descriptions of services provided by the whole sensor network, and they offer access to such services. At the same time, sink node act as

requestors to the sensor nodes, requesting their specialized data services.

Sensor nodes are **data providers**, providing raw and aggregated data. During the initial phase of WSN configuration, sensor nodes send their services description to sink nodes, thus executing the basic publish operation. Sink nodes also act as **registries** (at a lower level comparing to the Mashup Catalogue), keeping a repository with services descriptions of each sensor type existing in the sensor network. Therefore, Sink nodes act as registries for a single WSN, providing access to its different types of data as a Web service, while the Mashup catalogue acts as a register that virtualizes several WSNs as a single one, providing a vision of an ecosystem of sensor data.

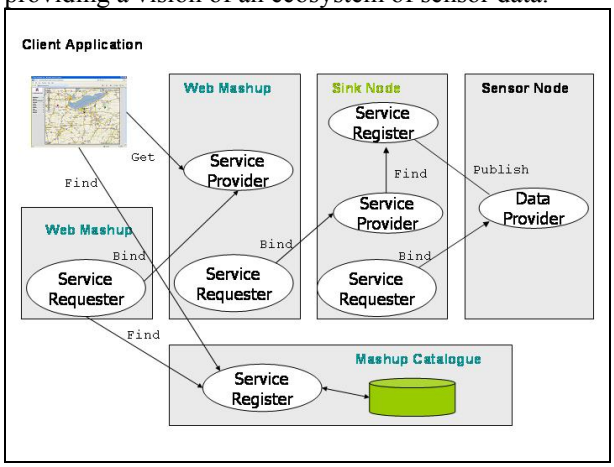

Figure 2. Architecture According to SOA

At the Sink nodes, the functionalities described by the SOA operations find and bind are grouped in one single one. Sink nodes provide the services description interface and, at the same time, provide access to such services. The Web Mashups interact with sink nodes, and sink nodes access sensor nodes passing the resulting data to the mashup. In fact, the operation find is only accomplished internally by sink nodes, which consult their repositories of services descriptions. When a Web Mashup submits a query to the WSN, it is actually executing a bind to the services supplied by the sensor nodes through Sink nodes. Therefore, a SOA bind operation issued by a Web Mashup is translated by the Sink to a **find** operation followed by a **bind** to the sensor nodes that can meet the request. The find operation sent to the Mashup Catalogue as well as the bind operation exchanged among Web Mashups follows the traditional SOA semantics.

3.4 Service Description

The generic services provided by a sensor network are described through two WSDL documents, describing the Sink node interface and the Web Mashup interface. In the Sink node interface the port types elements contain service descriptions used to access data provided by sensor nodes. In the Web Mashup interface the port type elements contain service descriptions used to access data provided by different WSN as well as other data compositions provided by other Web Mashups. Each service port type contains operations that can be thought as system APIs. Those operations contain parameters, defined in the document through messages. Bindings of operation definitions to their concrete implementation should be defined according to the underlying protocol. A port identification, indicating the place containing the operation implementation, can be done through any unique identifier (the device address, for sensor nodes and *uri*, for Sinks and Web Mashups).

The operations defined for the Web services specified in our system are [4]:

Publish_Sensor_Description: This operation is exposed only by Sink nodes and it is used by sensor nodes to advertise their sensing capabilities. Messages invoking such operation are exchanged by sensors in an initial phase of network configuration, soon after the WSN physical deployment. These massages include the node identification, a timestamp, the types of sensing devices available in each sensor node, geographical location, residual energy, maximum and minimum degree of confidence (concerns the accuracy of the sensor generated data), maximum data acquisition rate, among other parameters, and they are broadcast in the network until reaching a Sink Node. The following operations are exposed both by Sink nodes and Web Mashups.

Query_Sensors: This operation has the goal of exposing the specific features and capabilities in terms of data sensing of a given WSN (when this operation is invoked in Sink nodes) or of the sensing data compositions provided by a given Web mashup. The Query_Sensors operation is represented by two massages: an input message, without parameters, to call the respective operation, and an output message containing the answer which includes parameters as types of sensors, confidence degree, maximum data acquisition rate, etc. After knowing the sensing capabilities of a given *uri* (representing either a Web mashup or a Sink node associated to a WSN), the application (of a final user or a mashup application) is able to issue its interests, through different types of messages of interest subscription.

Subscribe_Synch_Interest: Operation used to send a message representing a **synchronous interest**. Such kind of interest corresponds to a simple query on the current state of some physical phenomenon monitored by the WSN. An example would be "which is the

temperature of region A?". So, messages to invoke such operation contain as parameters the type of data to be monitored (and so the type of required sensor), and the geographical location of the target area. On the other hand, asynchronous interests can correspond to long running queries or queries about some event detection, requiring different types of messages.

Subscribe_Long_Running_Interest: This operation is called by a message representing a long running query, as for instance: "which is the average temperature in area A for the next 24hs?". Parameters for these messages are the type of data (sensor), the geographical target area, the duration of the monitoring and the data acquisition rate.

Subscribe_Event_driven_Interest: This operation is invoked to report about the occurrence of a specific event in a target area, for instance: "Report whenever an elephant traverses area A". Messages to invoke such operation must include the type of sensor to be used (in this example, a movement sensor or an accelerometer), the description of the event to be detected and the geographical area to be monitored.

Publish_Data: When detecting data for which they have received a requisition (represented by interests), sensors nodes issue publish data messages (to trigger the respective operation in the Sink node). Such messages advertising data contain the data type, the instance (or value) of that type that was detected, the sensor current location (sensors can be mobile), the signal intensity, the confidence degree in the accomplished measurement, a timestamp, and the current sensor amount of energy. While in Sink nodes such operation is called whenever a sensor generates a raw data, or a data resulted from some simple aggregation procedure, this same operation when sent to Web mashups will return values of processed data generated in response to a given interest message, and resultant of the mashup composition process.

3.5 Communication Protocols

Web Mashups can obtain sensor data by issuing queries either to other Web Mashups or directly to a WSN through some sink node. The communication between Mashups and sink nodes is accomplished through conventional TCP/IP sockets (Figure 3). Web Mashups must generate a SOAP message describing their interests. Such a message is generated based on the sensor network service descriptions stored in the sink repository. Services descriptions are written in WSDL language. Since WSDL is an open and ubiquitous standard for services description, there are many tools [12,23] for automatic generating SOAP proxies. Proxies build SOAP messages and receive query results, thus representing the software interface among Web Mashups and sink nodes. By using our Web services complemented by the Mashup approach, instead of submitting queries in a WSN proprietary and pre-defined format, applications are able to choose the way they want to view and receive data.

The communication between Sink nodes and the sensor network is accomplished using a WSN specific data dissemination protocol (for instance [14]) and formatted as XML messages. Although SOAP is currently the de-facto Web services standard for message exchange, a fully compliant SOAP server implementation requires considerable processing and memory resources that are not available in most of the sensor devices. Moreover, the message size produced by SOAP is incompatible with the resource constrained environment of WSN [20]. Therefore, a more lightweight approach for sending Web services messages is needed to implement the communication between Sink nodes and sensor nodes. Fortunately, the WSDL language allows different bindings besides the SOAP binding. We make use of the HTTP binding [26] to encapsulate the message exchange between Sink and sensor nodes. A standard compliant HTTP server can easily implemented on most of the resource constrained WSN devices [20]. The HTTP binding is ideal for WSN communication since it reduces both the need of processing power and memory usage.

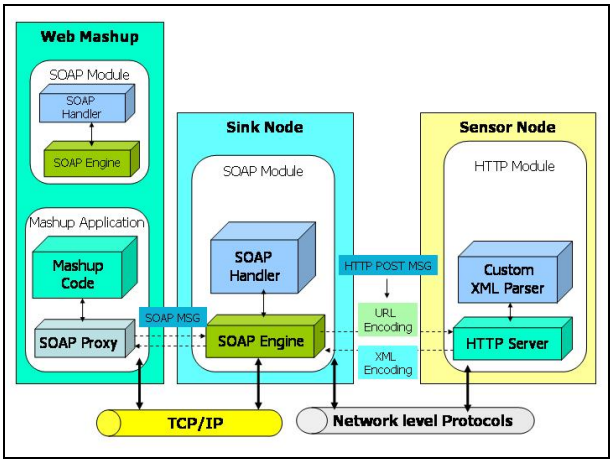

Figure 3. Communication Stack

Regarding the techniques for encoding operations and parameters within HTTP, the WSDL standard specifies different options [26]. In [20] the authors realized in-depth analyses of the performance of using the Web services approach on sensor devices. According to their results, the Web services implementations on sensor nodes should use *url* encoding or *url* replacement methods whenever possible since these techniques produces smaller size of messages and requires less processing compared to

sending a full XML message. However, there are situations where the sensor generated data are complex and requires a more structure message format. In this case, a POST-based XML encoding should be used instead of *url* encoding or *url* replacement techniques.

To enable sensor nodes to handle XML messages, an XML parser is needed. In order to minimize the processing requirements, a simplified and lightweight custom XML parser can be implemented within the sensor devices.

In our proposal, the communication is managed by two different components: SOAP modules and HTTP module. The SOAP module must be present in both Web Mashups and Sink nodes. The HTTP module must be present in sensor nodes.

The SOAP module is composed of a SOAP engine and a set of handlers. In Sink nodes, such module includes also a binding with the underlying network level protocol. The SOAP engine acts as the main entry point into the SOAP module. It is responsible for coordinate the SOAP message's flow through the various handlers and for ensuring that the SOAP semantics are followed. Handlers are the basic building blocks inside the SOAP module and they represent the messages processing logic, including the marshalling/unmarshalling of messages, header and attachments processing, serialization, conversions of data type, among any other basic functions.
The HTTP module encompasses the HTTP server and the custom XML parser.

4. Discussion and Work in Progress

In this paper we sketched steps for creating an ecosystem of sensor generated data by integrating different WSNs through the use of Web services and Web Mashup technologies. We argue that our proposal can promote the easy and fast access to environmental information by final users with different interests and expertise, thus exploiting the already deployed physical infrastructure of WSNs and enabling the building of physical mashups, i.e. small, ad-hoc composite applications encompassing real-world embedded devices.

Existent proposals that share our goal are presented in [1,9]. The work in [9] applies REST principles to embedded devices to enable the concept called "Internet of Things", which means the seamlessly integration of physical world (monitored by embedded devices) with computer networks. Based on the success of Web 2.0 Mashup applications the authors propose an approach for integrating real world devices to the Web, allowing for them to be easily combined to other virtual and physical resources. An import difference between such work and ours concerns the decisions on the communication framework. Instead of using an approach based on REST, we decided to adopt the Web services approach. Such decision was motivated by the following features provided by WS-* specifications: (i) support of asynchronous communication, which is often required in WSN environments; (ii) support for handling complex data structures; (iii) support for building a *message bus infrastructure* instead of only simple RPC communication model.

In [1] the authors propose the UbiSOA, an editor that allows creating ubiquitous computing mashups through simple tasks such as dragging and dropping graphical representations of the services involved in a target scenario. The focus of such work is only the editor and we intend to investigate the integration of UbiSOA with our proposed architecture, thus providing a complete framework for building WSN Mashup applications.

This proposal builds on a previous work developed by our research team [6]. Our current focus is to investigate existent mashup tools in order to integrate them in our solution. Finally, we will implement the complete architecture in real world sensors based on Mica motes [2] and SUN Spot sensor platforms [24].

Our service-based architecture provides the underpinning for building more general purpose networks instead of strictly task-specific ones, thus assisting a large range of users, possibly spread over the world, that share a common interest in an application domain.

5. Acknowledgements

This work is supported by the Brazilian funding agency CNPq under grants numbers 477226/2007-8, 477229/2009-3, 311515/2009-6, 480359/2009-1, 557.128/2009-9 and FAPERJ under grant number E-26/170028/2008.

6. References

1. Avilés-López, E. and García-Macías, J. A., "UbiSOA Dashboard: Integrating the Physical and Digital Domains through Mashups", In Proceedings of the Symposium on Human Interface 2009 on Conference Universal Access in Human-Computer Interaction. Part I: Held as Part of HCI International 2009, San Diego, CA, 2009.

2. Bajaber, F.; Awan, I., "Dynamic/Static Clustering Protocol for Wireless Sensor Network", Second UKSIM European Symposium on Computer Modeling and Simulation, 2008. EMS apos;08. Volume, Issue, 8-10 Sept. 2008 Page(s):524 – 529.

3. Chappell, D. A., *Enterprise Service Bus*, O'Reilly Media Publisher, June 2004.

4. Crossbow Technology. Available in http://www.xbow.com/. Last access: January/2010.

5. Delicato, F. C. et al., "A flexible middleware system for wireless sensor networks", In the Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware, Rio de Janeiro, Brazil, 2003.

6. Delicato, F.C., "Middleware Baseado em Serviços para Redes de Sensores sem Fio", (In Portuguese), PhD Thesis, Federal University of Rio de Janeiro, Brazil, June, 2005.

7. Google Maps API. Available in: http://code.google.com/apis/maps/. Last access: January/2010.

8. Graham, S. et al. Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI, Sams Publishing, 2002.

9. Guinard, D. and Trifa, V., "Towards the Web of Things: Web Mashups for Embedded Devices.", In Proceedings of Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web, International World Wide Web Conferences, Madrid, Spain, 2009.

10. Heidemann, J., et al., "Building Efficient Wireless Sensor Networks with Low-Level Naming", In Proceedings of the 2001 ACM Symposium on Operating Systems Principles, 2001, Canada.

11. Heinzelman, W., Chandrakasan, A. and Balakrishnan, H, Energy-Efficient Communication Protocol for Wireless Microsensor Networks. Proceedings of the 33rd Hawaii International Conference on System Sciences, 2000.

12. Huang, Mei-Wen; Liu, Hsu-Jung; Hsieh, Wen-Shyong, "A Hybrid protocol for Cluster-based wireless sensor networks", In Proceedings of the 13th Asia-Pacific Computer Systems Architecture Conference, 2008.

13. IBM White Paper, Web Services Toolkit. Available in: http://www.alphaworks.ibm.com/tech/.

14. Intanagonwiwat, C., Govindan, R., Estrin, D., "Directed diffusion: a scalable and robust communication paradigm for sensor networks", In proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000), pages 56-67, Boston, MA, USA, Aug 2000.

15. Krishnamachari, B. and Heidemann, J., "Application specific modeling of information routing in wireless sensor networks", In Proceedings of the IEEE International performance, computing and communications conference, vol. 23, pp. 717-722, 2004.

16. Kulik, J., Rabiner, W., Balakrishnan, H., "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks", In Proceedings of the 5th ACM/IEEE Mobicom Conference, USA, Aug. 1999.

17. Levis, P. et al., TinyOS: An Operating System for Sensor Networks, In Ambient Intelligence, Eds. Werner Weber, Jan M. Rabaey and Emile Aarts, Springer Berlin Heidelberg Pubs., 2005.

18. Manjeshwar, A. and Agrawal, D., "APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks", In Proceedings of the 16th International Parallel and Distributed Processing Symposium, IEEE Computer Society Pubs, Washington, DC, USA, 2002.

19. Merrill, D., "Mashups: the New Breed of Web App.", IBM DeveloperWorks, Aug. 2006. Available in: http://www.ibm.com/developerworks/xml/library/x-mashups.html. Last access January, 2010.

20. Priyantha, N., Kansal, A., Goraczko, M., and Zhao, F., "Tiny web services: design and implementation of interoperable and evolvable sensor networks", In the Proceedings of the 6th ACM conference on Embedded network sensor systems, Raleigh, NC, USA, 2008.

21. Smart, P.A, Maddern, H. and Maull, R. S., *Understanding Business Process Management: implications for theory and practice*, British Journal of Management, on line 2008.

22. Stojmenović, I. (Editor), Handbook of Sensor Networks: Algorithms and Architectures, Wiley Pubs, October 2005.

23. SUN Microsystems, "Implementing Services On Demand and the Sun Open Net Environment (Sun ONE)". Available in: http://www.sun.com/software/sunone/wpimplement/wp-implement.pdf, 2001.

24. Sun SPOT World, Available in http://www.sunspotworld.com/. Last access in set/2009.

25. UDDI.org, "UDDI Technical White Paper". Available in: http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.PDF, September 2000.

26. Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001. Available in: http://www.w3.org/TR/wsdl. Last access January, 2010.