Using Web Services for Bridging End-User Applications and Wireless Sensor Networks

Truong Ta[1], Nuru Yakub Othman*[1], Roch H. Glitho[1,2] and Ferhat Khendek[1]

[1] Concordia University, 1455 de Maisonneuve Blvd. W., Montreal Qc., H3G 1M8, Canada

{t_ta, ny_othma, glitho, khendek}@ece.concordia.ca

[2] Ericsson Canada Inc., 8400 Decarie Blvd., Montreal Qc., H4P 2N2, Canada roch.glitho@ericsson.com

* Tel: +1 514 487 1859 Fax: +1 514 848 2802 Office: EV015.163

Abstract. **Applications are the ultimate consumers of the information collected by sensors. There exist several frameworks for the interactions between end-user applications and sensors. They range from low-level APIs to databases and include Web Services. This paper is devoted to the use of Web Services for bridging end-user applications and wireless sensors networks. Its first contribution is a systematic evaluation of the current frameworks for the interactions between end-user applications and wireless sensors networks. The evaluation shows the potential of Web Services, compared to the other frameworks and motivates their usage in the case study. The second contribution is the definition of Web Services for the I-centric telecommunication services, and their implementation in a wireless sensor network that does not support Web Services. We demonstrate that Web Services are very promising as "bridges" and share the lessons we have learned.**

***Index Terms*— I-Centric telecommunications services, Middleware, Web Services, Wireless Sensor Networks.**

I. INTRODUCTION

THE end-user applications that can benefit from the information collected by Wireless Sensor Network (WSN) range from disaster prevention and relief to scientific exploration and include home intelligence. The interactions between applications and WSN are done via one or several nodes of the WSN, which act as gateway(s). These nodes offer an interfacing framework. Several frameworks have been proposed so far. They include low-level APIs, databases mobile code and Web Services.

This paper focuses on the use of Web Services for the interactions between applications and WSNs. It compares Web Services with other frameworks to show its potentials. In addition it defines Web Services for I-centric telecommunications services and shows how they can be implemented. Its contribution is therefore twofold: an evaluation of the existing frameworks and an example of how Web Services can be concretely used in a given application domain.

The paper is organized as follows. Section 2 proposes an evaluation of the existing frameworks for bridging applications and WSN. It demonstrates the potential of Web Services. In the following section, we introduce the application domain on which the case study focuses, i.e. I-centric telecommunications services. The fourth section describes the proposed Web Services and the overall architecture, before introducing and discussing the prototype in Section 5. This is followed by a discussion of related work and the lessons learned. We conclude in the last section.

II. AN EVALUATION OF EXISTING FRAMEWORKS

The trend in WSN is towards allowing applications from different domains to make use of sensor data. Developers are not expected to be sensor experts. It is foreseen that there will be numerous sensor networks, offering a variety of services, the owners of which may choose to charge for them, directly or via service providers. In this section we establish high level criteria to efficiently accommodate this trend. We then survey the existing frameworks and evaluate them based on our proposed criteria.

A. Criteria

For an application developer, an ideal framework should:

(1) *Provide a high level of abstraction* so that developers should not have to learn much, if at all, about sensors. The sensor network should be treated as a black box, rather than a white box with access to individual nodes.

Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06)
0-7695-2588-1/06 $20.00 © 2006 IEEE

(2) *Be a common technology based and easy to integrate with existing applications*.

(3) *Introduce very little or no overhead* (performance) in terms of time delay and network load.

(4) *Support both synchronous and asynchronous mode of data access*. Applications should not have to wait for response (blocked), nor should they be made to keep on re-querying the same information (polling). Once interest is expressed, the framework should be able to proactively disseminate collected data to interested parties.

(5) *Support different programming languages*; to allow access by applications written in different languages.

In order to accommodate different business models, the framework should also:

(6) *Provide security mechanisms.* This includes authentication, authorization, data integrity and non-repudiation. The framework should also allow the sensor networks to publish their services, and, should allow potential applications to discover and access them.

(7) *Provide mechanisms for publication and discovery.* This is to allow sensor networks to publish their services and allow potential applications to discover and access them.

B. Existing Frameworks

The main frameworks for collecting sensor data are the APIs, Database, Mobile Code and Web Services. The APIs come as low level commands and in specialized languages or in high level languages like C/C++ or Java, exposing the sensor capabilities to programmers. Examples include nesC [1] and galC [2] as special languages, and clientLib (Java libraries) for cricket sensors [3]. In the database approach, the sensor network is viewed as a distributed database [4]. Applications inject SQL like queries to the sensor network via the gateway and get the results back. Examples include TinyDB [5] and IrisNet [6], which uses XML instead of SQL.

In Mobile Code approach, the applications inject scripts or programs into the sensor networks via the gateway to visit the nodes, collect data and send them back, or ask the nodes to send back the data whenever certain phenomenon is detected. This has been done in the form of mobile agents, for example Agilla [7] and active networks as in SensorWare [8]. As for Web Services [9], they have been used in different forms, one of which is by being placed at the gateway to expose the services of the sensor networks. This involves mapping the Web Services to the APIs or other underlying frameworks. Several middleware for sensors have been built, but since they are proprietary techniques, we do not include them in this evaluation.

C. Evaluation

Reviewing these frameworks with respect to our requirements, we find that Database, Mobile Code and Web services fully provide high level of abstraction; while, as expected, low level APIs do not. Web Services and Database approach are very familiar to most developers and can easily be integrated into applications. Mobile Code is a paradigm not widely adopted, and, the APIs generally require knowledge of sensors and/or learning a specialized language. Compared to other frameworks, Web Services suffer from performance problems as they introduce significant overhead due to SOAP processing and XML encodings. Synchronous and asynchronous access is supported by all frameworks implementations. Web Services could meet the security requirements through the use of WS Security [10] and WS Security Non-repudiation [11] specifications; however such specifications are not yet fully adapted in existing implementations. The Database approach provides most of the security features, but it lacks comprehensive non-repudiation. APIs do not come with any form of security and specifications for Mobile Code security are not yet comprehensive. While both database and Web Services can be used by several programming languages, APIs and mobile code are generally language specific. Of the frameworks assessed, only Web Services provide publication and discovery mechanisms.

Table I below summarizes this evaluation of the frameworks in general, specific implementations may differ. The indicators F, P and N refer to Fully, Partially and Not meeting the requirement, respectively.

TABLE I
SUMMARY OF THE FRAMEWORK EVALUATION

	APIs	Database	Mobile Code	Web Services
Level of Abstraction	N	F	F	F
Familiar/Easy Integration	N	F	N	F
Little or no overhead	F	F	F	N
Synchronous/asynchronous	F	F	F	F
Support different languages	N	F	N	F
Security mechanisms	N	P	N	P
Publication and discovery	N	N	N	F

Web Services is one of the very promising frameworks, well known drawback being performance, which is being addressed by ongoing research [19] [20].

III. I-CENTRIC TELECOMMUNICATION SERVICES

The world wireless research forum [12] envisioned I-centric services as value added services in which the individual end-user, "I", is the centerpiece of service provisioning. The I-centric paradigm [13] puts the individual needs, preferences and environment as parameters in the communication system, while adapting in real-time these services to the ever changing situations and resources. Three distinct features are needed in an I-centric environment: ambient awareness, personalization,

and adaptability. Ambient awareness provides ambient information while personalization provides the individual preferences.

A. Ambient information

Ambient information in an I-centric environment refers to any collected information that is relevant to the current situation of the user. Ambient awareness is defined as the act of sensing and exchanging current ambient information. This information is classified into three main categories: spatial information, which includes location information and its derivatives (i.e. velocity, acceleration, orientation, proximity, etc), environmental information such as ambient lighting, temperature, air quality, noise level and physiological information such as blood pressure, stress level and heart rate. In this paper, we focus on spatial and environmental information.

The goal of acquiring user's spatial and environmental information is to provide services in an active context. Based on the acquired information, application and services adapt and change their behavior to fulfill user's needs and preferences. An example of adaptation can be a media streaming application that changes the sound output to text output on the display while the user moves from a silent environment to a noisy one. Environment and spatial monitoring is required to enable adaptation and is provided by ambient awareness.

In addition, service providers can offer personalized services to mobile users in a way that suit their individual needs and preferences at a specific place and time. For instance, a service can direct the user to her/his closest bank. In such a case, the service will require the knowledge of user location and her/his preferred banking institution. While personalization can be specified explicitly (i.e. users input their preferred banking institution), some mechanisms used to achieve implicit personalization require the knowledge of spatial and environmental information. Rule-based systems can be used to learn user preferences by observing user behavior (i.e. recurrence of a location or environmental conditions in which the user is consistently performing a task). Hence, spatial and environmental data are key elements in both personalization and adaptability aspects of I-centric communications.

B. I-centric applications

The basic idea of I-centric communication is to provide end-users with their own services that might change over time, place and situation. For instance, a media streaming application can restrict its functionality to "play the audio only" or "record for later playback" when the user is driving his car alone. Using location data, an application can notify an employee when his co-worker is back in his office from a long meeting and automatically establish a call among them (see Section 5.1). Another application can send targeted advertisement to shopper's portable devices (i.e. cell phone and Personal Digital Assistant) when they are standing still in a particular section of a retail store.

IV. Web services for I-centric telecommunication services

In this section, we first introduce the world model in which the proposed Web services operate, and then we proceed with the proposed Web services

A. The world model

A world model is necessary in order to establish a common ground for the interpretation of location and environmental information. There are three basic concepts in our world model: a *Space*, an *Entity* and an *Observable*. A Space is a static three-dimensional expanse defined by a unique center and range (or radius). For instance, an office room is a Space where the center is the center of the office and the range is the distance to one of the walls. An Entity is an object that can be located and that has an owner. An Entity is classified by its type such as laptop, PDA, printer or user. An Observable is a sensed and measured environmental phenomenon (i.e. temperature, the ambient light, etc).

B. The Interfaces

We propose six Web services to provide spatial and environmental information. Subscribe_Location() and Subscribe_AreaEnvironmentalData() are the basic Web services, while the rest are complex services that may be composed of the first two. The common parameters for these Web services are the service metadata parameters such as QualityOfContext which defines the desired minimum freshness of the sensed information, OneTimeOnly which specifies if the service should automatically unsubscribe after sending one response as opposed to periodically sending updates, and RateOfNotification which indicates the maximum and minimum rate for the service response. NotificationTrigger specifies a condition, based on the value of the sensed data, in which a notification (or service response) is necessary. Finally, Granularity and UnitsType dictates the desired level of abstraction of the sensed data and its associated units.

Subscribe_Location(): given an Entity, this returns its location information in terms of coordinates or Space, depending on the specified Granularity. The notification trigger is based on changes in coordinates or changes in the closest Space.

Subscribe_AreaEnvironmentalData(): given a set of Spaces, this returns the sensed environmental information associated with one or more observable. NotificationTriggers can be a threshold value while the granularity can be the measured value or a defined state (hot, cold, bright, etc).

Subscribe_Proximity(): given an Entity, this returns a list of sorted/filtered Entities within the specified proximity range. The Notification trigger is based on changes in the list of

sorted Entities.

Subscribe_PhysicalPresence(): given an area, this returns a list of Entities that are physically present there. The NotificationTrigger is based on the event of an new Entity entering or leaving the area.

Subscribe_Velocity(): given an Entity, this returns its velocity magnitude and direction.

Subscribe_EntityEnvironmentalData(): given an Entity, this returns the sensed environmental information around that Entity.

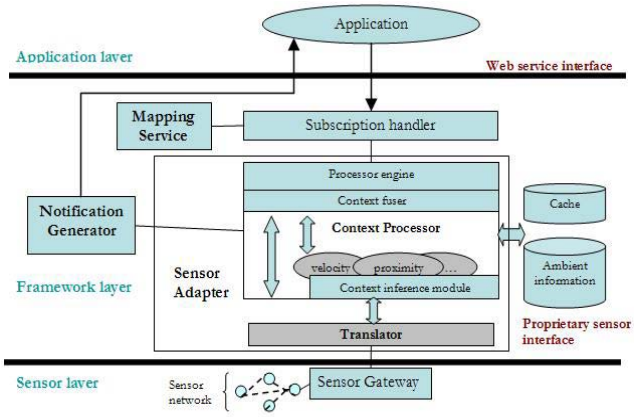

Fig 1. The overall functional architecture of the framework

C. *Architecture of the framework*

Fig. 1 shows the overall functional architecture. The main components are the Notification Generator, the Sensor Adapter and the Mapping Service. The Notification Generator builds and sends the notifications to the application. The Sensor Adapter is composed of a Translator and a Context Processor. The Translator receives and converts sensor-specific data and converts its data format into a standard format used by the context processor. The Context Processor has three roles fulfilled by the processor engine, the context fuser and the context inference module. The context fuser retrieves ambient information from persistent storage and integrates it with the results from the Translator or context inference module, into one single, coherent and more confident result. The processor engine monitors the ambient information and generates the notification events based on the specified trigger. The context inference module is the service logic responsible for computing high level ambient information based on low level ambient information sources. The Mapping Service maps the world model elements such as Entities, Spaces and Observables to elements in the sensor world. For instance, an Entity might map to a SensorID or other necessary data for the retrieval of the sensor reading. The Mapping Service also determines which Translator or which context inference module is required for delivering the ambient information. In some cases it maps into more than one source, for example, velocity can be sensed by an accelerometer sensor or inferred by sampling several instances of location data at a known time interval. The Subscription Handler receives the Web service requests and, based on the mapping service, dispatches it to the Sensor Adapter with the appropriate Translator and context inference module. The system also contains a shared cache for optimization reasons and a database for all logging all measurements.

V. PROTOTYPE

Here we outline an example of applications built using the proposed Web services. The mapping to the backend sensors is also elaborated in this section.

A. *SenseCall*

SenseCall is an application that establishes a 3rd party call between two users when they are both in their respective office space. SenseCall, shown in Fig. 2, uses the framework to retrieve the location of both users and a Web service built on top of a Parlay gateway [18] to deliver the 3rd party call. SenseCall uses Subscribe_Location() and its goal is to test and demonstrate the ambient awareness framework.

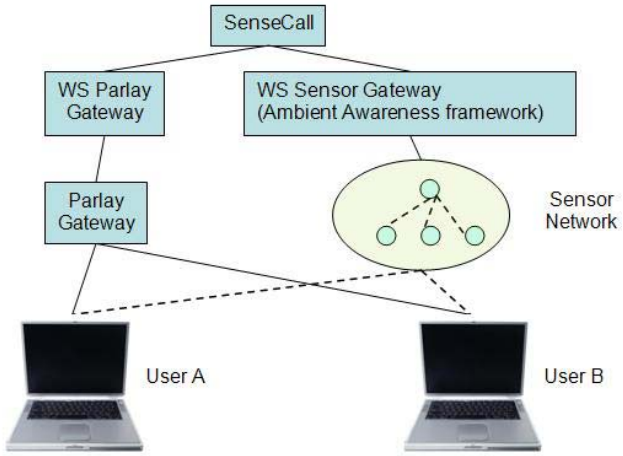

Fig 2.The overall architecture of SenseCall

B. *The mapping to sensors*

The ambient awareness framework uses MIT crickets [3] for location data and Crossbow [16] MTS300 [17] for environmental data. Crickets provide a sharp accuracy of 1-3 centimeters and have coverage of up to 10 meters indoors. Location information can be expressed in terms of coordinates or Space identifiers. The MTS300 can sense ambient light, sound and temperature.

For Subscribe_Location() and Subscribe_Area EnvironmentalData() we do not need to implement the inference module as ambient data is provided directly by the

sensor hardware. However, we define the mapping functionalities in the mapping service (Fig. 1) as relational databases. An Entity-Relationship database table is defined to link an Entity to its LocatorTechnology. This tells us where to look for locating that Entity and gives us the necessary information (IP addresses, port, sensorID, etc) to retrieve the location data. However, for simplicity we only consider one technology for acquiring location hence the Entity-LocatorTechnology Table maps to the same Locator Technology.

For Subscribe_AreaEnvironmentalData(), there is also an Entity-Relationship table that correlates Space, Observable and information on how to retrieve the environmental data. The complex Web services are implemented as inference module using a combination of location and/or environmental data. For instance, Subscribe_Proximity() uses the cache and other techniques (collision detection algorithm, filtering list) to optimize overall efficiency. Similarly, Subsribe_Velocity() requires several location coordinates at a known time to derive velocity.

VI. Related work and lessons learned

Open GIS Consortium, Inc. [15] (OGC) is defining the standard specifications to enable the sensor web vision. Sensor web focuses on environmental data related to Earth's ecosystem (i.e. pollution, temperature, water levels). Their work aims at solving the interoperability of network sensors by defining standard interfaces for sensor data and enabling their ubiquitous access through the Internet. However, their Web service interfaces for accessing sensor data are defined at a low level. GetObservation() in the Sensor Collection Service requires the knowledge of the sensor identifiers for the data collection. There is also no support for specifying notification triggers.

Parlay-X is a set of telecommunication capabilities exposed as Web services. Parlay-X defines a basic Web service interface to get the location information expressed in latitude and longitude (not suitable for indoor location). Parlay-X does not define any notification triggers and the communication scheme is based on request-reply model. Finally, Parlay-X Web service is limited to location information only from the ambient information point of view.

From an application developer perspective, the usage of high level Web services allows rapid design and implementation of ambient aware applications such as SenseCall. This comes at no surprise in addition to the increase of overhead in response time and network load. In some cases, response time is doubled when comparing Web services to the proprietary sensors APIs. On the other hand, it seems that Web services are not an ideal environment to handle sensors APIs that employs non-serializable programming constructs (daemon Thread, Sockets, i/o Stream). Since Web services often needs to serialize it state, dealing with these programming constructs may become highly inefficient.

VII. Conclusion

In this paper, we evaluated current frameworks for bridging applications and WSN and we proposed the usage of Web service as a foundation for a framework. The framework at its core is an adapter responsible for communicating with sensors at one end and exposing the processed data as Web services at the other end. We defined six Web Services for delivering spatial and environmental information and we demonstrated the usage of the framework by building a prototype application.

We envisage performance evaluation for future work. For instance, we plan to take measurements concerning the delays or network load incurred from Web Service message overhead. The results can then be used to study the suitability of this framework for different types of I-centric services.

Acknowledgment

We would like to thank Rajesh Karunamurthy and Joana Sequeira da Silva for their help in this project. This work has been partially supported by Natural Sciences and Engineering Research Council of Canada (NSERC) and Ericsson.

References

[1] Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., Culler, D.: The nesC Language: A Holistic Approach to Networked Embedded Systems. Proceedings of the ACM SIGPLAN 2003, San Diego, California, USA, (2003) 1–11

[2] Cheong, E., Liu, J.: galsC: A Language for Event-Driven Embedded Systems. Design, Automation and Test in Europe (DATE'05), vol. 02, no. 2, Design (2005) 1050-1055.

[3] Adam Smith, Hari Balakrishnan, Michel Goraczko, Nissanka Priyantha.: Tracking Moving Devices with the Cricket Location System. Proc. 2nd USENIX/ACM MOBISYS Conf., Boston, MA, June 2004

[4] Govindan, R. et al.: The Sensor Network as a Database. Tech-Rep 02-771 CS Department, University of Southern California, September 2002

[5] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstain, Wei Hong.: TinyDB: an acquisitional query processing system for sensor networks. ACM Transactions on database systems (TODS), volume 30, Issue 1, March 2005, 122 - 173

[6] Phillip B. Gibbons, Brad Karp, Yan Ke, Suman Nath, Srinivasan Seshan.: IrisNet: An Architecture for a World-Wide Sensor Web. IEEE Pervasive Computing, Volume 2, Number 4, (October-December 2003).

[7] Chien-Liang Fok, Gruia-Catalin Roman, Chenyang Lu. :Rapid Development and Flexible Deployment of Adaptive Wireless Sensor Network Applications. In Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'05),

Columbus, Ohio, June 6-10, 2005, pp. 653-662. Available at: http://mobilab.wustl.edu/projects/agilla/

[8] A. Boulis and M. B. Srivastava. A Framework for Efficient and Programmable Sensor Networks. In proceedings of OPENARCH 2002, New York, June, 2002. Available at: http://nesl.ee.ucla.edu/projects/sensorware/

[9] F. Curbera et al.: Unraveling the Web services Web: An Introduction to SOAP, WSDL and UDDI. IEEE Internet Computing, Vol. 6, No2 (March-April 2002) 86-93

[10] Specification: Web Services Security: http://www-06.ibm.com/developerworks/webservices/library/ws-secure/

[11] Web Services Security: Non Repudiation Proposal Draft 05, 11 April 2003 http://xml.coverpages.org/ReactivityWSNonRepudiation-05.pdf

[12] Wireless World Research Forum: http://www.wireless-world-research.org/

[13] Arbanowski, S., Ballon, P., David, K., Droegehorn, O., Eertink, H., Kellerer, W., Van Kranenburg, H., Raatikainen, K., Popescu-Zeletin, R.: I-centric Communications: Personalization, Ambient Awareness, and Adaptability for Future Mobile Services. IEEE Communications Magazine (September 2004) 63-69

[14] Parlay-X Web Services Specification: http://www.parlay.org/specs/

[15] Open Geospatial Consortium, Inc.: http://www.opengeospatial.org/

[16] Crossbow Technologies: http://www.xbow.com/

[17] MTS300 sensor: http://www.xbow.com/Products/productsdetails.aspx?sid=75

[18] Torreira da Silva, J.S., Hassan, K., Glitho, R., Khendek, F.: Web services for Conferencing in 3G Networks: A Parlay based implementation. Proceedings of ICIN'2004, Bordeaux, France (October 2004)

[19] N. Abu-Ghazaleh, M. Lewis, M. Govindaraju.: Differential Serialization for Optimized SOAP Performance. In Proc. 13th IEEE International Symposium on High Performance Distributed Computing, Honolulu, Hawaii, June 2004, pp. 55-64.

[20] M. Migliardi, R. Podesta.: Performance Improvement in Web Services Invocation Framework. In Proc. 18th International Parallel and Distributed Processing Symposium, Santa Fe, New Mexico, April 2004, pp. 110-122.