A Reference Architecture for Sensor Networks Integration and Management

Valentina Casola, Andrea Gaglione, Antonino Mazzeo

Dipartimento di Informatica e Sistemistica
Universita' degli Studi di Napoli, Federico II
Naples, Italy
{casolav,andrea.gaglione,mazzeo}@unina.it

Abstract. Sensor networks have become a highly active research area due to their potential for providing diverse new capabilities for a wide variety of real world applications. Distributed applications require to collect information from a lot of different sensor systems, retrieved data are usually heterogeneous from many points of view and they need to be integrated to cooperate for a common objective. In this paper we present the SeNsIM framework, a scalable software architecture for the integration of heterogeneous sensor systems. SeNsIM enables the deployment of applications based on multiple sensor systems by providing a standard way to manage, query, and interact with sensors. We propose the architectural and data model of the SeNsIM framework and provide a method to describe sensor systems using XML as modeling language in order to facilitate sharing of structured data across them.

1 Introduction

Sensor networks have become a highly active research area due to their potential for providing diverse new capabilities for a wide variety of real world applications. They have made many novel applications possible to emerge in the fields of environmental monitoring [1], detection and classification of objects in military settings [2] and health applications [3]. Moreover, in the last years a proliferation of *Wireless Sensor Networks (WSN)* technologies is increasing. Such systems are composed by low-cost and low-power sensor nodes ("*motes*") able to measure different parameters and to communicate over wireless channels [9].

The diffusion of sensor systems, together with their applications have led to a large heterogeneity in the logic for interfacing and collecting data from these systems. As for WSNs, ad hoc programming languages (e.g. nesC [4]) and operating systems (e.g. TinyOS [20]) have been developed to support motes programming and to express the application processing in terms of messages exchange among near nodes. Furthermore, different **middleware** platforms based on macro-programming models have been proposed in order to bridge the gap between the application and the underlying hardware and network platforms. However they are commonly used when a single application operates over a single WSN, while the application development for multiple WSNs is a rather cumbersome work.

Nowadays the interaction with multiple sensor systems is required by a lot of applications, as those typical in the emerging pervasive computing paradigm adopted in several domains (e.g. telemedicine, crisis management, military [11]). Moreover monitoring applications of wide geographical areas have highlighted the research problem of the integrated management and correlation of data coming from various networks that cooperate for a common objective [6]. In order to embody sensing infrastructures into computing paradigms based on the application development for multiple WSNs or sensor systems, specific integration frameworks for accessing different data sources are needed.

This paper describes the design of the *SeNsIM (Sensor Networks Integration and Management)* framework, it is not just a middleware for WSNs, but it is a more general integration platforms for heterogeneous sensor systems. SeNsIM (i) makes possible the deployment of applications based on multiple sensor systems/networks and (ii) allows a generic user or an application to easily access data sensed by a network. Furthermore, SeNsIM is able to ensure scalability since it is very easy to deploy new networks in the system.

The architectural model for the integration has been realized by exploiting a *wrapper-mediator* paradigm: the mediator accesses the sensor data by means of ad hoc connectors (wrappers), one for each networks, which strongly depend on sensor network technologies. So the mediator is responsible to format and forward user requests to the different networks, while the wrappers are responsible to translate the incoming queries and forward them to the underlying sensors.

The remainder of this paper is organized as follows. In Section 2 an overview of solution for WSN management is reported. In Section 3 we describe the proposed architectural model to manage sensor networks integration and the proposed data model for information exchange. In Section 4 some details on the system architecture and implementation are provided. Final discussions are reported in Section 5.

2 Related Works

In this section the most well known **middleware** solutions for WSN and some **integration platforms** for heterogeneous sensor systems are presented.

Middlewares for WSNs refers to software and tools which provide a system abstraction so that the application programmer can focus on the application logic without having to deal with the lower level implementation details [5]. Different middleware solutions for WSN management have been proposed [13, 5, 16]; they differ in terms of querying and data aggregation models and for the assumptions about the kind of sensor nodes, topology, size, and other features of the network. It is important to point out that all middlewares platforms are mainly focused on the possibility to access to a single WSN implemented with a specific technology but they do not provide functionalities to access to different heterogeneous networks. At this aim some interesting researches related to integration techniques for heterogeneous sensor networks have taken place, but nowadays only few architectures have been proposed.

In [8], Ahn and Chong propose an intelligent bridge as an interoperable architecture for messages exchange between heterogeneous wireless sensor networks. They define a general messages exchange mechanism that uses XML as message style and SOAP as transmission protocol. They also conducted a case study based on two WSNs with different network protocols (ZigBee and Bluetooth). Hourglass [18] provides an Internet-based infrastructure for connecting sensor networks to applications. It offers topic-based discovery and data-processing services, but focuses on maintaining quality of service of data streams in the presence of disconnections. GSN (Global Sensor Networks) [7] facilitates the flexible integration and discovery of sensor networks and hides arbitrary stream data sources behind its virtual sensor abstraction. It enables the user to specify XML-based deployment descriptors in combination with the possibility to integrate sensor network data through plain SQL queries over local and remote sensor data sources. Other works try to define a middleware integration architecture that enables interoperability between sensor systems. For example the ESP framework [15] enables sensor systems to be queried without having to deal with the low-level implementation of specific access methods. It provides a mechanism to describe and model sensor systems using *ESPml*, an XML-based language, by which information regarding the sensor deployment can be specified. Finally, IrisNet [10] can be considered a hybrid approach to integration. It is a web infrastructure for easy deployment of wide-area sensing services. The architecture consists of *sensing agents* (SA) which collect and pre-process sensor data and of *organizing agents* (OA) which store sensor data in a hierarchical, distributed XML database that supports XPath queries.

Most of these approaches are still in a design phase since they lack of a real implementation. Some try to define a common exchange mechanism among different sensor systems in order to facilitate the integration, but often they are strongly related to the adopted standards and technologies. Others are based on new technologies (i.e. web services for ESP), but the impact on performances is not discussed, yet.

In contrast to the examined approaches, we will illustrate a more general architecture to provide a general-purpose infrastructure for sensor systems integration, and a more general way to express queries introducing a query visual language. Our system provides a complete data retrieval and management platform with a simple user interface by which is possible to execute queries. Moreover our architecture is flexible and can easily provide APIs by which an application can easily interact with the different sensor systems.

3 Design principles

The need for deployment of applications based on multiple sensor systems and the management of wide geographical areas have highlighted a new research issue, i.e. the possibility of integrating and managing, in an integrate way, all data coming from the various data sources. Unfortunately sensed data are not available in a unique container, but are distributed in heterogeneous repositories. So the major issue for an integration system lies in the heterogeneity of repositories that makes data management and retrieval processes hard tasks to achieve.

To face such problem, we need to define:

- an *architectural model* able to support in an efficient way the management of such data even when sensed by different networks;
- a *data model* capable of representing in a unique logical view the "sensor data", and that can be used by any applications.

3.1 The architectural and data model

As already said, sensor systems are adopted in many application fields and their proliferation has massively increased; due to these reasons there is a tremendous heterogeneity in the logic for interfacing and collecting data from these systems. In most sensor systems an application can directly access to sensor hardware by means of opportune drivers. In the case of WSN, an operating system between the hardware platform and the application allows an easier use of the available sensing functions and often a further middleware layer between the operating system and the application provides an easy way to access sensor data. Furthermore sensed data may be differently structured according to the specific representation of different sensor systems.

We propose a novel integration platform (*SeNsIM, Sensor Networks Integration and Management*) which aims to bridge the gap between heterogeneous sensor systems and to provide a unique way to manage, query and interact with them. SeNsIM is made of a *mediator* component which hides networks heterogeneity to end users or applications by means of ad hoc connectors (*wrappers*) (one for each network). Each wrapper explores and monitors the local sensor networks and sends to the mediator an appropriate description of the related information according to a common data model. The mediator organizes such information and keeps a unique view of all systems in order to satisfy user or application queries.

The architectural model we propose is made of four logical layers (see figure 1):

1. an application or user layer to submit queries and elaborate the retrieved data;
2. a mediator layer to format and forward queries to specific wrappers;
3. a wrapper layer to extract and manage network information and data;
4. the sensor system layer with or without a specific middleware or operating system.

As for the data model, in the literature, sensors have been modeled by using two kind of approaches [17, 19]: (i) structural approach focuses the attention

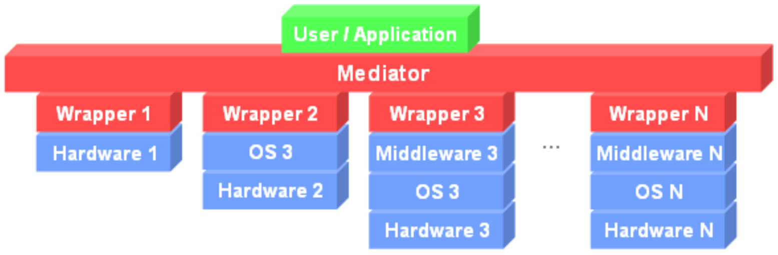


Fig. 1. Abstraction layers of the proposed architecture

on the sensor-structure in terms of hw/sw components, while (ii) data-oriented approach schematizes a sensor using a behavioral description. The latter mainly refers to sensor global information (type, producer, description, etc...) as well as to variables that a sensor can measure (temperature, light, humidity, etc...), predicates that a sensor can calculate (e.g. temperature greater than a threshold) and sensor operating state/mode (On, Off, Sleep, etc...).

In our approach we define a *sensor node* as an object characterized by a tuple of information which combine both structural and behavioral features. The state of a sensor can be modified by means of classical getting/setting functions, while the measured variables can be accessed using the sensing function. Collection of sensor nodes, disposed according to clustering policies and to a given topology form a sensor network. According to our model a *network* object has to include global information such as type of sensor system, middleware (if present), supported sensor board as well as information related to sensor components (list of sensors, possible list of clusters, topology matrix). Network global predicates (e.g. average temperature of the network) also can be represented in our model.

For brevity sake we do not report all the details of the data model, we just say that it has been represented in XML format, since XML provides platform independence, interoperability and can be easily parsed [21]. XML-based descriptors let have a unifying grammar by which systems can describe their abilities and define a standard language protocol with which the different entities in the framework can communicate. Our XML descriptor (*structXML*) is directly derived from the data model and represents features of both networks and sensors. Each wrapper builds a structXML descriptor after having injected a discovery queries on the underlying system (generally at startup). If some parameters could not been extract in an automatic manner (i.e. sensor producer, middleware for WSN), a wrapper administrator can manually fill the structXML descriptor with the missing information.

As an example, figure 2 shows a network (N) and a pseudo-XML description of it. For simplicity we do not show the real structXML descriptor, trying to point out only the main features which can interest the reader. The network is a WSN composed by 4 temperature sensor nodes (*S1* and *S3* are in the "On"state, while *S2* and *S4* are in the "Sleep" state, being used as redundant units), grouped

```
S1 =
<
 {(id,1), (type,'mote')}, {(voltage,200)},
 {on}, {temperature},
 {CheckTemp(t,threshold,CondOpearator)}
>
S2 =
<
 {(id,2), (type,'mote')}, {(voltage,200)},
 {sleep}, {temperature},
 {CheckTemp(t,threshold,CondOpearator)}
>
S3 =
<
 {(id,3), (type,'mote')}, {(voltage,200)},
 {on}, {temperature},
 {CheckTemp(t,threshold,CondOpearator)}
>
S4 =
<
 {(id,4), (type,'mote')}, {(voltage,200)},
 {sleep}, {temperature},
 {CheckTemp(t,threshold,CondOpearator)}
>
S5 =
<
 {(id,5), (type,'sink')}, {},
 {on}, {},
 {}}
>
```

```
N=
<
 {(id,1), (midlleware,'tinydb'),(type,'WSN')},
 {
  -,-,-,-,child-parent wl,
  -,-,-,-,child-parent wl,
  -,-,-,-,child-parent wl,
  parent-child wl,parent-child wl,parent-child wl,parent-child wl,-
 },
 {S1, S2, S3, S4, S5},
 {(C1,[S1,S3]), (C2,[S2,S4])}, {},
 {TopologyDiscovery()}
>
```

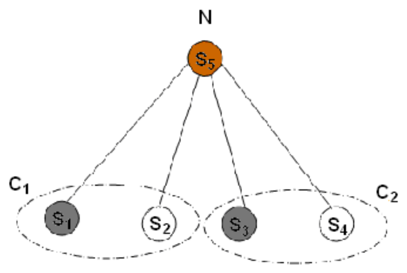


Fig. 2. An example of network description

into two clusters (*C1* and *C2*), and by one *base station* (*S5*). The base station, also called *sink*, is a mote, in general connected with a PC-class device, which acts as a gateway between the network and the end user.

4 The Reference Architecture

In this section, we firstly describe the two main components of the SeNsIM system: the wrapper and the mediator. Then we illustrate interactions taking place between a generic wrapper, the mediator, the user and the underlying network during two main usage scenarios of SeNsIM.

As already said, wrapper components work as adapters between the mediator layer and the sensing platforms. They should gather the features of the underlying network and of its sensors (e.g. discovering the network topology with its clusters/groups of sensors, the state of single sensors, etc...) and, above all, they should able to access sensor data by querying single sensors, clusters or the entire network. The mediator should classify sensor information sent by wrappers and should provide a simple way to users or applications for querying the networks. In figure 3 the architecture of wrapper and mediator components is reported, illustrating its main modules. The macro-modules of both components, which are represented in dashed lines, carry out a the main features of the related component.

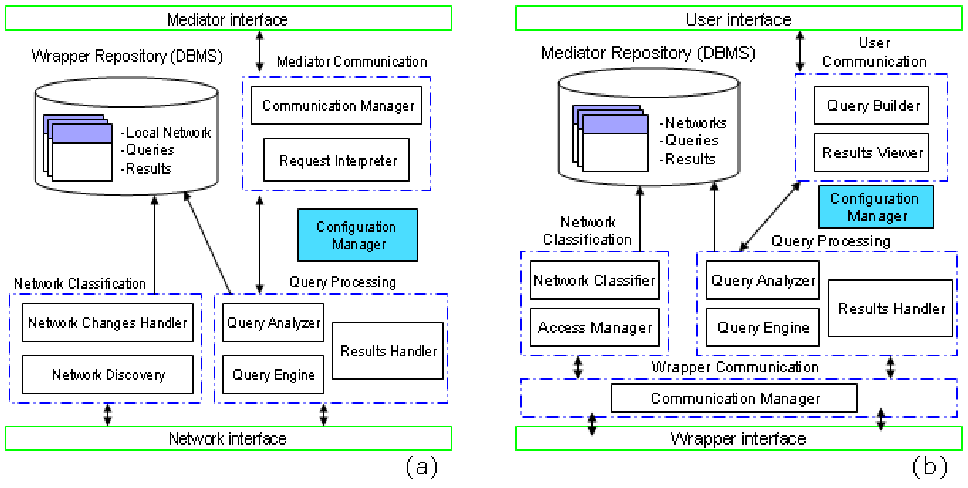


Fig. 3. (a) Wrapper architecture; (b) Mediator architecture.

As for the wrapper, (i) it has to discover, extract and manage information about the underlying network and its sensors (*Network Classification*); (ii) it has receive user queries from the mediator and execute them on the system by using its APIs and the local query language (*Query Processing*); (iii) it has to manage the communication process with the mediator (*Mediator Communication*). Network Classification and Query Processing modules interact with DBMS for storing and accessing information related to network/sensors (according to the data model), queries and related results. Wrapper is also provided with a *Configuration Manager* module, which the can be used by an administrator of the system during the discovery phase of the system to set the state of sensors or to define clustering/grouping policies.

The mediator, on the other side, (i) has to classify and manage metwork/sensor information sent by wrapper (*Network Classification*); (ii) it has to manage user queries (*Query Processing*); (iii) it has to manage the communication with wrappers (*Wrapper Communication*); (iv) it has to interact with the user, by taking his queries and showing him the related results (*User Communication*). Also in this case Network Classification and Query Processing modules interact with a DBMS which stores data related to networks with their sensors (intensional part of the data model), user queries and related results. Finally, mediator is provided with a *Configuration Manager* module: it is used during the initialization phase of the system to define the admissible information for a network and its sensors according to the data model.

Figure 4 shows the interactions that take place among main system components in the two main usage scenarios, *registration* and *querying*:

– *Registration* Any wrapper needs to register itself before communicating with a mediator. At first, each wrapper creates an XML document describing

the system as a whole; this is done by analyzing the sensor system (i.e. by injecting a discovery query), and generating the appropriate XML to represent the system. Then a wrapper sends a registration request message to the mediator, that verifies the possibility of including a new system in the framework and sends a response message to the wrapper. If the registration request is accepted, the wrapper sends the XML document to the mediator, which stores the related information in a DB.
– *Querying* The querying process starts when a user sends a query request through the mediator user interface after having selected the destination of the query (a network or a specific sensor, among those connected to the framework trough the wrappers). The mediator takes the query parameters and creates an XML document which sends to the appropriate wrapper. The wrapper extracts the parameters by parsing the XML document and executes the query on the local system. The query results are grouped by the wrapper, which also creates another XML document, and periodically send it to the mediator. Finally the mediator extracts the results from the XML and shows them to the user.

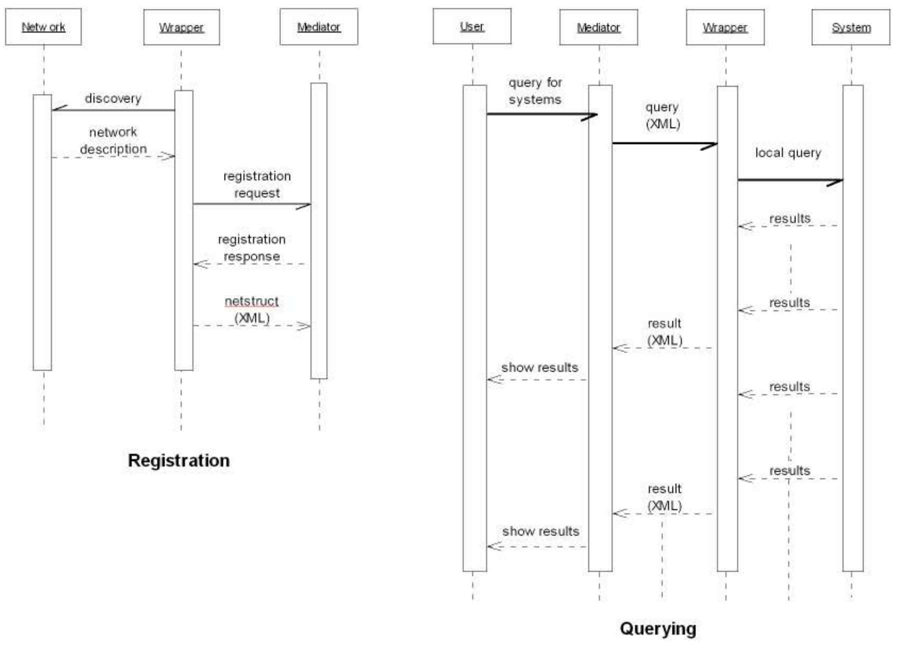
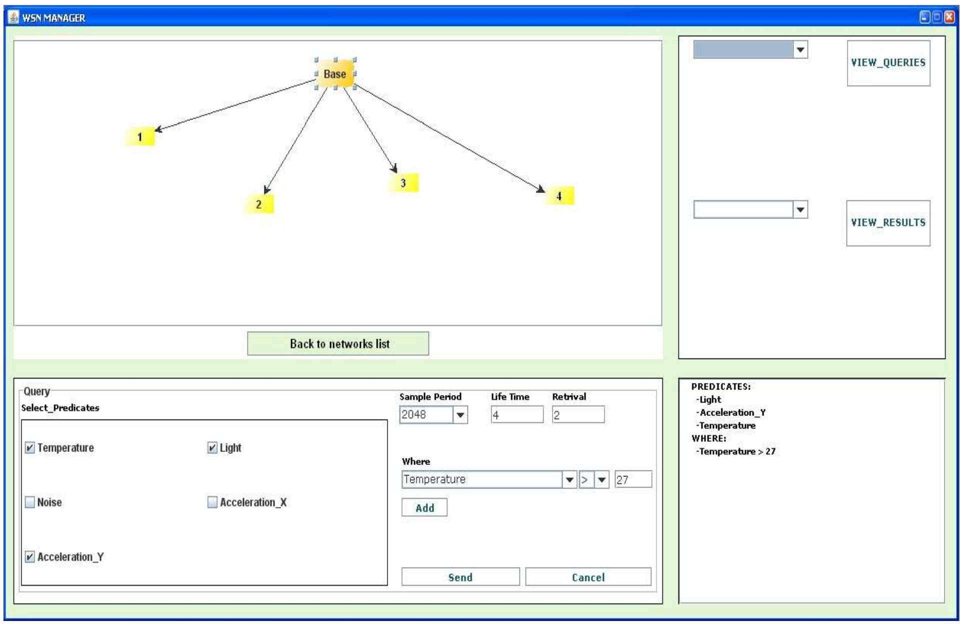


Fig. 4. The communication protocols in the Registration and Querying scenarios

The system provides support for monitoring queries that retrieve the requested data from the sensor systems and return the corresponding responses in real-time as well as for event queries. Many context-aware applications need to trigger some actions after that some events have been generated from sensor systems.



[✓] Temperature [✓] Light
[ ] Noise [ ] Acceleration_X
[✓] Acceleration_Y

Sample Period: 2048
Life Time: 4
Retrival: 2

Where: Temperature > 27

PREDICATES:
-Light
-Acceleration_Y
-Temperature
WHERE:
-Temperature > 27

Fig. 5. Network querying screen

4.1 Implementation details

In the current implementation of SeNsIM, the mediator component and all its modules have been fully realized. Furthermore, we have implemented the wrapper component for TinyDB based networks. TinyDB [14] is a middleware for WSN which provides a query processing system for extracting information from a network of TinyOS sensors. Unlike existing solutions for data processing in TinyOS, TinyDB does not require users to write embedded C code for sensors. Instead, it provides a simple SQL-like interface to specify the data to extract, along with additional parameters as, for example, the data refreshing rate. Given a query, TinyDB collects that data from motes in the environment, filters it, aggregates it together, and routes it out to a PC, it also implements power-efficient in-network processing algorithms.

All SeNsIM components are written in the Java programming language exploiting its libraries for networking (java.net), multithreading (java.lang.Thread), managing DBMS (JDBC) and manipulating XML files (DOM). The communication protocol between wrappers and mediator has been implemented by exploiting both UDP and TCP sockets: in particular we used UDP sockets to exchange simple messages (e.g. registration request in the registration scenario), while TCP sockets were used to exchange XML data files (networks structure, queries and results). An ORACLE 10g DBMS has been adopted to store data in the wrapper and mediator repositories and a graphical interface has been realized to simplify the interaction of SeNsIM with users. Finally we set up a first experimental testbed with two TinyDB based networks in order to demonstrate the validity of our approach. Figure 5 provides a GUI screenshot showing the panel for querying a network after it has been selected by the user from the networks list.

5 Conclusions and Future Works

We have presented a system for the integration and management of sensor networks. It allows a single unified view of a lot of systems and enables a flexible deployment and interconnection between them, even if located in different places. The architecture is based on the Mediator/Wrapper paradigm in order to provide a layered and scalable architecture. The Wrapper is responsible for managing hardware and systems heterogeneity while the Mediator is responsible of managing the communication with the wrappers and provides a uniform interface for queries on sensed data and network features to user and applications. Future works will be devoted to evaluate system scalability and performances and to develop service oriented interfaces in order to integrate our system with recent standards proposed by the OGC and W3C consortium to model sensors and sensor networks.

References

1. D. Steere, A. Baptista, D. McNamee, C. Pu, J. Walpole, “Research challenges in environmental observation and forecasting systems”, in Proc. ACM/IEEE MOBICOM ’00, Boston, August 2000.
2. Y. Hu D. Li, K. Wong and A. Sayeed, “Detection, classification, and tracking in distributed sensor networks”, In IEEE Signal Processing Magazine, pages 17-29. IEEE, March 2002.
3. L. Schwiebert, S. K. S. Gupta, and J. Weinmann, “Research challenges in wireless networks of biomedical sensors”, in Proc. ACM/IEEE MOBICOM ’01, pp. 151-165, 2001.
4. D. Gay, P. Levis, D. Culler, E. Brewer, “nesC 1.1 Language Reference Manual, http://nescc.sourceforge.net/papers/nesc-ref.pdf”, March 2003.
5. S. Hadim, N. Mohamed, “Middleware for wireless sensor networks: A survey”, in Proc. 1st Int. Conf. Comm. System Software and Middleware (Comsware06), New Delhi, India, Jan. 8-12, 2006.
6. F. Flammini, A. Gaglione, N. Mazzocca, V. Moscato, C. Pragliola, “Wireless Sensor Data Fusion for Critical Infrastructure Security”, Proceedings of the International

Workshop on Computational Intelligence in Security for Information Systems CI-SIS'08 - Advances in Soft Computing, pp. 92-99, 2008.

7. K. Aberer, M. Hauswirth, A. Salehi, "The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks", Technical Report, 2006.
8. S. Ahn, K. Chong, "Building a Bridge for Heterogeneous Sensor Networks", Proceedings of the Fourth IEEE Workshop on SEUS-WCCIA 06.
9. I.F. Akyildiz, M.C. Vuran, O.B. Akan, W. Su, "Wireless Sensor Network: A survey REVISITED", Computer Networks Journal, 2005.
10. P.B. Gibbons, B. Karp, Y. Ke, S. Nath, S. Seshan, "IrisNet: An Architecture for a World- Wide Sensor Web", IEEE Pervasive Computing 2(4) (2003).
11. S. Hadim, N. Mohamed, "Middleware: Middleware Challenges and Approaches for Wireless Sensor Networks", IEEE Distributed Systems Online, March 2006.
12. W.B. Heinzelman, A.L. Murphy, H.S. Carvalho, and M.A. Perillo, "Middleware to support sensor network applications", IEEE Network, 18(1):6-14, 2004.
13. K. Henricksen, R. Robinson, "A Survey of Middleware for Sensor Networks: State-of-the-Art and Future Directions", MidSens '06: Proceedings of the international workshop on Middleware for sensor networks, ACM Press, Melbourne, Australia, p.60-65 (2006).
14. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: An acquisitional query processing system for sensor networks", ACM Transactions on Database Systems, 30(1):122-173, 2005.
15. S. Reddy, T. Schmid, N. Yau, G. Chen, D. Estrin, M. Hansen, M. B. Srivastava, "ESP Framework: A middleware architecture for heterogeneous sensing systems", December 2006.
16. K. Romer, "Programming Paradigms and Middleware for Sensor Networks", GI/ITG Fachgespraech Sensornetze, Karlsruhe, 26-27 Feb 2004.
17. SensorML Project. URL: http://vast.uah.edu/ SensorML/
18. J. Shneidman, P. Pietzuch, J. Ledlie, M. Roussopoulos, M. Seltzer, M. Welsh, "Hourglass: An Infrastructure for Connecting Sensor Networks and Applications", Technical Report TR-21-04, Harvard University, EECS (2004) http://www.eecs.harvard.edu/U syrah/ hourglass/papers/tr2104.pdf.
19. T. Skov, R. Bro, "A new approach for modelling sensor based data", Sensor and Actuators B: Chemical, vol. 106 (2), 2005, 719-729.
20. TinyOS Project. URL: http://www.tinyos.net.
21. W3C Architecture Domain, Extensible Markup Language (XML). URL: http://www.w3.org/XML/.