International Conference on Computational Science ICCS 2010

Wireless sensor networks in knowledge management

José-Fernán Martínez, Pedro Castillejo, Marta Zuazua, Ana-Belén García, Lourdes López, Antonio DaSilva, Vicente Hernández

Department of Telematic Engineering and Architectures (DIATEL)- Technical University of Madrid (U.P.M.)
Crta. Valencia, Km.7, 28031 Madrid, Spain

Abstract

Research in Wireless Sensor Networks (WSN) necessarily touches on many research topics of Computer Science, Electronic Engineering and Telecommunication, basing on the existing work in related fields. However, the peculiarity of the WSN field is the interplay and integration of these foundation subjects, yielding a distinct topic worthy of further study in its own right.

One of the main open issues in WSN research is to abstract the applications of complex low-level mechanisms, and one of the most powerful and flexible ways to achieve this is creating a Middleware layer that cover all this functionality, provide services to applications, allows the intercommunication among components, adapts dynamically to different operation modes and clearly differentiated from low-level components. Knowledge management and ontologies are also helpful when WSN are used to monitoring and taking decisions. We deploy a WSN in a testing scenario in order to control environmental parameters according to user profile stored in the system.

© 2010 Published by Elsevier B.V.

Keywords: Wireless Sensor Network; Middleware; Metadata; Knowledge Management; Reasoning

1. Introduction

Wireless Sensor Network (WSN) is a form of ad hoc network consisting of large number of heterogeneous tiny sensors with communication, processing, and storage capabilities. Applications of such networks range from battlefield communication systems (e.g., intrusion detections, target surveillance) to environmental monitoring networks such as habitat monitoring, chemical sensing, infrastructure security, inventory, and traffic control [1]. For example, sensors may be distributed across a house in order to detect and identify one person and change environmental parameters (temperature, humidity, etc.) to fit individual profile. Sensors are very small electronic devices, and thanks to last advances in miniaturization technologies their size will be even more reduced, achieving...
the goal of embed sensors in any place. A sensor consists of several components (see figure 1):

- Battery
- Communication module
- Central unit with processor, memory, etc
- Different kinds of sensors and actuators.

![Figure 1 Components of a node](image)

To hide the underlying platform differences and to decouple the OS from hardware platform, we need Middleware (MW), a connectivity software that consists of a set of services that allow multiple applications running on one or more machines to interact across a network[2].

Middleware services provide a more functional set of Application Programming Interfaces (APIs) than the offered by the operating system and network services to allow an application to[3]:
- be independent from network services and other lower layers (i.e. operating systems)
- provide interaction with other applications or services
- be reliable and available
- scale up in capacity without losing functionality

The main goal of the middleware layer is to facilitate the application developers its task supporting the development, maintenance, execution and reprogramming the most frequently used sensed-based application in a generic WSN, considering that they are composed by heavy constrained devices, commonly known as motes [4].

Focusing on WSN Middleware detailed problems, we basically come across that Middleware in WSN must support generic applications and the basic operation of the net, taking into account the following aspects:
- these nodes are tiny devices (their volume will be 1cm3 in few years)
- they are limited in energy, but they could get it by solar cells (depending on their location)
- they tend to fail due to exposure to adverse environmental conditions or losing their power
- they are limited in resources regarding the CPU, the amount of memory, radio bandwidth and coverage

There are some handicaps regarding this kind of networks. Apart from losing communication or changing the network topology, we must bear in our minds that these devices are heterogeneous, as they may come from different manufacturers, have different calculation power, different memory capacity, etc.

Having a large number of nodes in a network is positive due to the redundancy it causes (increases reliability), although it could cause scalability problems.

Traditional distributed middleware solutions (e.g., DCOM, CORBA, PVM, MPI) are normally heavyweight in terms of memory and computation requirements, and therefore unsuitable for WSN with scarce energy and processing resources. Instead, simple, easily implemental, lightweight designs are required. Moreover, the
middleware design needs to address the unique operating modes of WSN that are significantly different from traditional networks, including ad hoc deployment and dynamic operating environments. Energy is the scarcest resource of WSN, for this reason, algorithms and protocols need to address for lifetime maximization, robustness and fault tolerance, and self-configuration.

1.1. Structure

This article is divided in the following chapters:

In first chapter we introduce the WSN, its network structure, the composition of the nodes and the handicap when working with them. Also includes Middleware definition and its relation to such networks.

The second chapter presents a series of works related to knowledge management in wireless sensor networks and suggests the contribution of our system compared to those that already exists.

The third chapter discusses the architecture of the middleware and the various components that comprise it and focused on the knowledge management-based components.

The fourth chapter presents the results obtained with the middleware described above.

The fifth chapter discusses the conclusions and future work to develop from this paper

2. Related work: knowledge management in WSN

Knowledge Management (KM) comprises a range of practices used by organizations to identify, create, represent, distribute and enable adoption of what it knows, and how it knows it. It is also the process of handling knowledge from the available sources for an organization and connecting people with that knowledge.

Obviously, knowledge management is not a single technology but is instead a collection of indexing, classifying, and information-retrieval technologies coupled with methodologies designed to achieve results desired by the user. There are kinds of Information Technologies applied to Artificial Intelligence technologies that are the major knowledge management enablers, such as ontologies, intelligent agent, database, data mining, browsers, decision support, XML, pointer to people, knowledge base system, intranet, groupware, document retrieval, and distributed data model.

One of the problems when building a knowledge-based system is the knowledge sharing. There is a large diversity and heterogeneity of knowledge representation formalisms. It is really difficult to share knowledge across systems. If knowledge is from another knowledge-based system, we also need protocols to provide interoperability between kinds of knowledge-based systems. However, the greatest difficulty is, in a higher, knowledge level, there is often a higher-level modeling, taxonomical, and terminological mismatch of different systems, even if they belong to the same application domain. One of the main ideas to solve this problem is to establish standard models, taxonomies, vocabularies and domain terminologies, and use them to develop knowledge and reasoning modules. Ontologies provide the basic structure so that knowledge bases can be built[5]. Each ontology is an explicit independent specification of some topic or formal solution of a meta-problem in some subject areas. It specifies concepts to describe knowledge in that subject area. This knowledge includes types of entities, attributes and properties, relations and functions [6]. Each ontology describes the related terms in that subject area. It also provides the logical statement to describe what they are, how to relate to each other, and how they cannot relate to each other. With ontologies, knowledge could be shared and reused in knowledge-based systems. Similar to the Object-oriented way, ontologies provide taxonomic hierarchies of classes and the subsumption relations. Ontologies make possible to define an infrastructure in order to integrate intelligent systems at knowledge levels.
Ontologies offer the knowledge being shared among members in the system and facilitate information integration. Moreover it is reusable for building blocks in knowledge modeling [7]. Therefore ontologies are thought to be useful in collaboration, interoperation, education and modeling. Nowadays, a lot of knowledge-based systems are based on a metadata model, which actually is an ontology-based metadata model, and RDF (Resource Description Framework) can describe such models.

Related work in this field includes “Global Sensor Network” project (GSN, Karl Aberer et al. [8]). In this work, authors propose a system to provide a infrastructure for integrating heterogeneous sensor network technologies thanks to the abstraction. This platform consists in discovering and integrating sensor networks and data, providing distributed query, filtering and data combination. The advantage of the project is the use of a declarative XML-based language distributed in the network for monitoring some parameters. In comparison with GSN, our proposal includes shared knowledge management and data aggregation schemas. In military context, G. de Mel et al. have developed several works such as “Reasoning and Resource Allocation for Sensor-Mission Assignment in a Coalition Context”[9] and “Flexible Resource Assignment in Sensor Networks: A Hybrid Reasoning Approach”[10]. Both works are focused on mission task and battlefield sensor network deployments. Although they provide good solutions (an efficient resource allocation algorithm in the form of a decision support tool and a combination of rule-based and ontology-based reasoning inspired on current semantic web technologies) they do not provide generic solutions (as they are focused on military scenarios) and are constrained to semantic web languages limitations. Our proposal uses generic data types (byte or integer) to share data, in order to make it as open as possible, and also offers interaction with actuators networks.

3. Design of middleware architectural context and abstraction levels

The following architecture overview will define each node’s structure, and it will help the developers to do their work in an easier way. This architecture is based in layers, and in each layer is divided into components with a different functionality. Next figures show an overview of this architecture and abstraction levels that will be explained in the following paragraphs.

Figure 2 Node architecture approach (left) and mapping abstraction levels to services layer schema (right)

Figure 2 shows the architecture of a normal node. Gateway architecture differs of the previous seen node, since it has to deal with two different networks. Gateway protocol stack is shown on next figure.
The proposed middleware architecture is Service Oriented (SOA). Each layer will provide services to the upper layers. The design is thought to have different levels of abstraction to make it more flexible and to be more adaptable to future modifications.

High Services Layer provides high abstraction level services. This layer will abstract from the hardware and the low services of the application. This layer consists of 4 elements: Query (to translate events to a higher level, a query to link the middleware and the application level is needed), Configurating (offers to the high level firmware reprogramming services, adding the new commands for the management components of the node and also letting the management of new sensors adding afterwards, providing the system of flexibility at high level), Profile Management (contents a data base of all the profiles in a zone) and Reasoning (takes the decisions to spread the information in the network (or to aggregate it in some cases), and sending it to the sink when it is necessary; these decisions will depend on the information from the nodes, from the sink, from the actuators, etc.)

Control Layer deals with the communications between motes, and sink. Each event must pass through this layer, which automatically redirects it to the destination component. It works like a link between all the motes. As a great amount of events are to be exchanged between layers and components, the creation of this entity is greatly justified.

Low Services Layer provides services at first abstraction level, it means with infrastructure knowledge. It will help the developer not to pay attention to some details from the hardware level. Network Configuration component will manage some networks parameters like addresses, routes, and link attributes. V.A.S. (Virtual Actuators / Sensors) is an abstraction layer of sensors and actuators integrated in the mote. This way it is not necessary to control through the physical layer the function of these devices, meaning the programmer spares himself the effort developing certain algorithms and controllers related to this level that may result quite thorny. V.A.S covers all steps from the physical to the programming library level. In our case, thanks to the use of .NET, part of the V.A.S is already implemented and we have at our disposal sensor access libraries at a programming level. However the V.A.S described in our design provides functionality not implicit in the .NET libraries, and therefore it will be necessary to combine them.

3.1. Communication protocol between components

The communication model we are treating works by way of events depending on the kind of event. An event can be dispatched inside a node among different components and can be sent to another node by means of the ‘networking configuration’ component.

3.1.1. Intra-Node components communication

At the ‘Networking Configuration’ component there will be a method to ‘listen to’ all the other components, waiting to have an event to start doing networking process.
When a component is going to throw a type of event it publishes it using direct-calls to the corresponding component. In this procedure the key method is `public bool send(int type, object load)`, that allows a component to inform that it will throw events of some `eventType`. The following figures explain the way to act:

![Diagram](image1)

**Figure 4** Room in (left) and room out (right) cases

There are special actuators which have the task of detecting users whenever they enter/exit established zones (RFID based). The profile detected information is directly sent to the database, which will deal with transmitting the data to the WSN through the sink.

In order to make faster calculations, inside each mote a profile data base will be included, so each node will know which users are in its zone.

### 3.1.2. Inter-Node components communication

The inter-node communication between two components is made by encapsulated events in a PDU (Protocol Data Unit) by adding the destination.

![Diagram](image2)

**Figure 5** Network PDU

Data are broadcast transmitted every time there is any communication between actors, therefore each mote must be able to handle the information that passes through.

Whenever any relevant action or notification of the system is performed, it is transmitted to the group of actors that must manage or monitor it in any way. The motes events are divided into two groups: Sensor events and tracking events, they must be sent to the gateway, to the server or both.

The Network Configuration component will always be paying attention to in case of an event occurs, so when a component throws an event of a type, the referred component will be able to realize and to act consequently.

This is the way a component can make an event made by itself to appear in another node.

### 3.2. Reasoning

Most important system modules are reasoning modules. In the application level this module deals with the reception of the data sensed by the motes and will consequently act depending on the priority level of such data, if this one is higher than the priority level of the system’s stored data, it will reconfigure the network parameters and will store the new information. If this information is different to the one stored in the system, but has a lower priority level, the system will not change nor store any parameter.

Middleware level is where there is more reasoning weight; it will be in charge of generating the PDUs (Protocol Data Unit) with the user priority, as well as storing the data into the system. It will also decide whether to increase or decrease the parameter levels.
In network level a field is included to the PDU, consisting of a hash of the received PDU, which will be compared to the hashes of the last \( n \) received PDUs.

In order to make this component as open as possible, ontologies must be used. For this reason, this component has two inputs (data and algorithms), and one output (results), as shown in Figure 6 (left). Inputs ontologies have been created for inter-compatibility purposes. Inputs and outputs always consist of an extensible collection of key-value pairs, in order to identify data collected/returned and its value.

![Figure 6](image_url) Reasoning elements (left) and key-value pair environmental example (right)

When environmental parameters are been evaluated, data will be stored in user profiles, which are composed of parameters (such as temperature, humidity, etc...), as shown in Figure 6 (right). Parameters were made as flexible as possible. The number of parameters included in each profile is open, so each profile might have several numbers of parameters. The same applies to algorithms. Each algorithm deals with one or more parameters included in one or more profile.

4. Testing scenario and results

The proposed scenario (shown in Figure 6) is based on a house with a sensor network (WSN) spread all over it. Furthermore, the user would be provided with a RFID compatible device with the installed network, with the task of identifying the user inside the mentioned network so as to enable the use of profiling techniques.

Once the user enters the network coverage range, the device transmits the identifier for its distribution. This way, the network always knows which users are in which rooms. With this information, as well as the environmental one sampled by the sensors (humidity, temperature, luminosity...) and the one stored for each profile, the network makes decisions concerning the environmental parameters of the house. On order to modify such parameters, the network sends information to actuators through the gateway so that these can act consequently.

The stored profile is dynamic. If the user would like to change his/her preferences, these changes would be updated in his/her profile for future actions. To show each user profile, or even change it, touch screens are used. Whenever a user gets close to one of the touch screens, his/her profile is displayed, as well as the allowed modification options. A password could be required to fulfill this step, if the system is wanted to have a higher security level.

![Figure 7](image_url) Testing scenario deployment
The testing was made in one of the U.P.M. (Universidad Politécnica de Madrid) research laboratories. During the deployment, WSN consists of a set of three Imote2 (Crossbow) nodes, one of them connected to the gateway (sink node). Once we have deployed the network architecture, we proceed to activate the motes.

Each of the effects has been verified in the elected types of conditions which may occur and were applied to each Use Case defined for its validation.

For each condition we must read the description of the scenario, causes and effects, check if the stage is configured and started correctly, read the associated test case scenario actions, check the behavior described in the test case and fill in the document or evidence the results of the test case.

The execution results of test cases will lead to an analysis based on which improvements and recommendations for future versions of the prototype will be drawn.

Below is just one example of how the system works in one of the different cases/conditions that exist.

**Figure 8** Testing scenario: two persons in the same zone with different sorts of priorities

In the upper figure a person enters in a room with a priority ‘1’ (the same value or higher). There is another person inside, and needs the exact value (priority ‘3’). There is no incompatibility. The parameter with more accuracy is more restrictive than the priority of 'more than or equal'. There is no incompatibility between the two preferences, priority parameter is placed exactly. For the other parameters, the average will be done.

Once compared the obtained results with the expected ones, we can conclude that our system fulfils use case initially raised in its design, therefore, perform their functions properly.
5. Conclusions and future work

The aim of modern Wireless Sensor networks is to establish an artificial intelligent system which is able to automatically collect environment data when necessary and make correct decisions as soon as possible. Unfortunately, almost every deployment of sensor networks has hundreds of nodes so decisions cannot be made in a very short time. In the other hand, some situations are very similar and consequently there is a resemblance in different processes. Regarding KM used in WSN middleware, the system translates useful information into shared meta-knowledge, helps nodes make decisions using other shared knowledge, and shares this problem descriptions and solutions with other systems.

Current WSN middleware uses database approach to organize the node structure, which could be also used applying intelligent database techniques in KM. Challenges always increase our interest in exploring these unknown areas, hence we address these knowledge issues and associate them with current sensor network problems.

Regarding to our project, our V.A.S component, which abstracts the programmer from having to manage with the physical layer, is also an innovating solution to take into account.

WSN are quickly developing and are supposed to be a revolution in people’s life style in the near future. That is why the middleware must be capable of offering a great adaptability to a large bunch of situations in which it could be deployed and used, always trying to offer, at least, a good performance.

Once the testing phase has been made, and a consequence of the previously observations, this paper will finish with a final set of conclusions and future works to be done:

1) Given the results of tests exposed, we prove that the system is effective and uses little memory of the nodes. The future work for this case is to prove it with real actuators (not only simulators) and to increase the number of nodes which take part in the test.

2) We detected in our tests that with our code settings was not producing any bottleneck or any flow memory. We consider necessary to repeat this kind of testing with real actuators and more nodes.

3) As we have risen in the point about scalability, a new challenge for this system would be the use of other network topologies, namely the one based in clusters, enabling us to increase the number of nodes in our network and see if it works at high scale.

This work covers specific items and scenarios, but many future improvements can be applied. As future investigation lines, we propose including more security and Quality of Service into the system, as well as improving tracking (with a more precise localization system), or even including biometrical parameters to recognize profiles. Other future approaches could include different sensing parameters or adapting our scenario to outdoor conditions.

Acknowledgements

The work presented in this paper has been partially funded by the Spanish Ministry of Industry, Tourism and Trade in the framework of the European Research Project ‘ESNA’ (European Sensor Networks Architecture), which belongs to the ITEA 2 (Information Technology for European Advancement 2) programme.
References


