SEMSuS: Semantic Middleware for Dynamic Service-Oriented Sensor Network

V. Sangeetha and L. Jagajeevan Rao

Abstract Sensor network appears to be the technology of the 21st century. Middleware for wireless sensor networks (WSN) acts as the interface point between wireless sensor nodes and higher application layers. A middleware can introduce and enhance access to the underlying networks in intelligent ways. In this paper we present a middleware for wireless sensor networks that uses a set of technical statements, such as patterns and styles, in order to achieve flexibility, compatibility, autonomy and adoption. The middleware exposes the functionality of the network as semantic web services, so that applications can access its functionality through web services. Sensor web combines sensors and sensor networks with a Service-Oriented Architecture (SOA). The Service Oriented Architecture allows us to discover, describe and invoke services from a heterogeneous software platform. Dispatcher and Interceptors are used inside the network. We propose a scenario in which two services are exposed a semantic web services and designed to run in a constrained environment and they are exchanged in accordance with capabilities of the network.

Keywords Wireless sensor network · Middleware · Service oriented architecture · Semantic web services

1 Introduction

The main purpose of Middleware for sensor networks is to support the development, maintenance, deployment and execution of sensing-based applications [A]. The emerging technologies and concepts to increase the flexibility of software solutions

V. Sangeetha (✉)
KL University,Vaddeswaram, Guntur, India
e-mail: sangeethalokanadam@hotmail.com

L. J. Rao
School of Computing, KL University, Vaddeswaram, Guntur, India
e-mail: jeevan@kluniversity.in

N. Chaki et al. (eds.), *Computer Networks & Communications (NetCom)*,
Lecture Notes in Electrical Engineering 131, DOI: 10.1007/978-1-4614-6154-8_7,

in various contexts have emerged in computing. It will potentially add millions of new data sources to the web. In distributed systems, mechanisms to access, reflect on and change their own interpretation is a reality [1]. At run time the systems are reflexive, in which reflection is related to the ability to inspect and adapt the system at dynamic process [2].

This paper aims to show the design, implementation, and execution of the SEMSuS (Semantic Middleware for Dynamic-Service-Oriented Sensor Network). It allows run-time adaptation of the functionalities that run on its platform. Adaptation as considered in this work results from the adoption of architectural styles, such as Service-Oriented-Architecture (SOA) and design patterns, such as the Dispatcher [3] and Interceptor [4].

Reflection can be applied in different types of systems, such as in wireless sensor networks (WSN), but requires considerations concerning the intrinsic characteristics of these networks. A WSN consists of several small nodes with restrictions on computing power, the bandwidth of wireless communication and the amount of energy available [5]. Therefore the gradual increase in computational cost, as a result of the addition of flexible mechanisms, must be minimized in order to extend the lifetime of the network. Sink node is one type of the network gateway, so that the communication node sources to any external system network should be through the sink.

Another feature of WSNs is that they are closely related to the environment in which they are deployed. The sensor nodes are scattered in an environment of observation for the extraction, processing and sending data to requesting information outside the network through one or more sink nodes.

When using SOA, the WSN plays the role of service provider, while the application is its client. Associated with the need for adaptation and the characteristics of WSNs, is the solution for middleware platforms capable of performing various types of applications. In order to maximize the automation of activities related to services, they are implemented as Semantic Web Services (SWS). Then a network service can be found by matching semantic. After the choice semantics is invoked at runtime, a specific implementation of the dynamic service among the network.

In Sect. 2, we look at the concepts and technologies needed for future enhancement of the work, such as SOA, WS, SWS and Ontologies. Section 3 gives the clear explanation of middleware SEMSuS. In this section are first analyzed the structural components and then the interactions between these components.

In Sect. 4 we show the use of middleware in order to evaluate the implementation and finally in Sect. 5 presents the conclusion and the future work.

2 State-of-Art

Similarly to that presented in [1], a WSN can be considered from the perspective of a provider of services that meets the needs of applications running on its platform. The adoption of services makes transparent its implementation and standardized access features, allowing the addition of flexibility and dynamism to the solutions, becaused

different services can be selected and exchanged it with no change in the form of client interaction. Adding semantic Web technologies to the services description promotes greater degree of automation by making improvements in the mechanisms of selection, composition, invocation and monitoring of services.

2.1 Dynamic Service-Oriented Architecture and Web Services

The identification of common data operations and transformations on sensor data has to introduced the sensor web paradigm. Sensor Web combines Sensors and Sensor networks with a Service Oriented Architecture (SOA). A SOA allows us to discover, describe and invoke services from a heterogenous platform using XML and SOAP standards. The components are available as independent services that are accessed in a standardized manner [6]. There are three main actions that are performed in SOA: publish find and bind/invocation. To support them there are three main components the client, the provider and the registry. Figure 1 represents the SOA with the components (in blue), and their interactions (yellow) in the sequence.

The basic interactions or activities are: (1) Publish—the provider makes the publication of service they can provide in a place known by the clients (the record); (2) Find—The client requests information about the services published in the registry that can meet their needs; (3) Bind/Invocation—possession with the address of the service that meets its needs, the client interacts the service provider to deliver it.

Web Services (WS) are the most popular realization of SOA [6]. The invocation of the WS is made through XML messages [7] that follow the SOAP standard [8]. The operations contained in a Web Service, as well as its inputs and outputs parameters are described in WSDL [9]. The service's WSDL is published in a registry called Universal Description, Discovery and Integration (UDDI), which is consulted by clients seeking services. This use of a set of standards makes interoperable the WS.

2.2 Ontologies and Semantic Web Services

An Ontology is a formal and explicit specification of a shared conceptualization [10]. Where "formal means readable by computers, explicit specification with regard to concepts, properties, relations, functions, constraints, axioms explicitly defined,

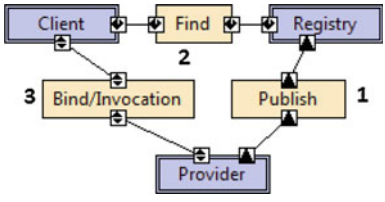


Fig. 1 Service oriented architecture

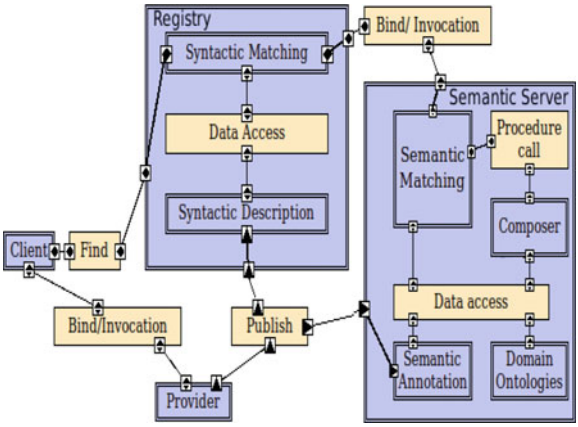


Fig. 2 Reference architecture for SWS

shared knowledge that is consensual, and conceptualization concerns an abstract model of some real-world phenomenon" [11]. So, an ontology defines concepts and relations between them in a field of knowledge.

In the Fig. 2, the more internal components shows the division from the logical viewpoint, while the external shows the deployment viewpoint. In this architecture there are two matching in different locations: one syntactic, in the Registry, and other semantic, in the Semantic Server. The Registry performs the matching comparing, syntactically, the client request with the relevant service publication, while the Semantic Server performs the semantic matching from the semantic descriptions of both services in the requests. To carry out the semantic descriptions it is necessary to use domain Ontologies for semantic annotation.

The SWS is designed by the addition of semantic technologies to the patterns of WS, promoting the enrichment of the services description and behavior. Figure 2 shows a reference architecture that expresses the operation of the SWS. The mechanisms that allow the services provided by a WSN to be accessed through Web Services provide interoperability through a standards-based framework for exchanging information. Such patterns for WS are designed to represent the service interface, how they are distributed and how they are raised. However there is a limitation in expressing what services do. The automatic integration is compromised because it is based solely on the syntax and semantics rather than a description of the service [6].

3 Related Work

To design a successful dynamic Service-Oriented Architecture for sensor network, each classification focuses on different aspects and different purposes. Some of the main middle wares for WSN are Agila [12], MATE [13], TinyDB [14] and Tiny Lime

[15] among others. More specifically there are approaches that make use of SOA, an can be found in [1]. Semantic Web technologies relate to other works as in [16, 17].

The SOA-based approach proposed in SEMSuS intended to allow flexibility in adding new services in a manner similar to that implemented in [1], which shows a set of services, among which service configuration and adaptation of the network that allows activation of adaptation policies when any change occurs in the environment. Differences in the middleware proposed work in relation to SEMSuS is that it has as the execution platform the Java Platform Micro Edition (Java ME) [18], which are less stringent as that based on Mica motes. So there is a greater capacity to perform the middleware in the nodes. Another difference in the middleware proposed in [1] is that its implementation is only in the nodes, while in SEMSuS middleware is distributed by a number of devices that transcend the limits of WSN, such as a web server, allowing load balancing middleware so that they can be used Mica nodes in the network. Another difference is that in [1] the middleware does not use semantic web technologies in their mechanisms.

A work that makes use of semantic services is [16] which use ontologies to represent data from sensor nodes, as well as its services, and allows the descriptions of the services have semantic annotations. The paper presents a three-layer model, in which reside the Platform, which is related to the hardware and operating system. The software layer allows making transparent the heterogeneity of the platform, and the semantic layer makes transparent the semantic heterogeneity of the Software. An interesting fact in [14] is a proposal to make use of Ontologies that describe the data and services in WSN. That is, each node can represent its skills and its data through the domain ontology, which are presented and analyzed in the work. The use of Ontologies allows nodes of different platforms, and using different representations of data, can communicate with each other made the network more flexible.

The difference between the work in [16] and SEMSuS lies in the fact that the mechanisms that use Ontologies to describe data and services are performed in the environment of the WSN. In SEMSuS the publication and use of Ontologies comes at a higher level of architecture because of the limitation of sensors that do not allow the use and processing of files such as OWL [19]. Another factor is that the proposal in [16] is performed to describe the capabilities of each node of a WSN, while in SEMSuS is made to describe functionalities of a WSN.

Another work is [17] which makes use of semantic web technologies to build a semantic middleware for autonomic WSN. The system that allows autonomy for network uses a set of rules of inference on Ontologies and logic-based fuzzy. Data networks are mapped to domain Ontologies and rules act on the Ontologies, which provide a high level of abstraction to the applications. Locus inferences also run on sensor nodes in order to provide necessary data for a set of services that implement the autonomous capabilities. The difference in [17] and SEMSuS is that the services that implement the autonomy using data provided by inference engines, which does not occur in SEMSuS. Another difference is that the SEMSuS uses SOA for providing means of access to the WSN via the web.

4 The SEMSuS

The SEMSuS is a middleware for WSN that aims to present sensor networks as providers of semantic services. The adoption of SWS allows client applications and networks to exchange information that can be semantically interpreted by both. Because the use of late binding technique promoted by the Manager and Dispatcher, the SEMSuS choose the implementation of the service according to the network capacity, i.e. if the network is with fewer resources is chosen implementation that consumes less resources at the expense of guarantees, for example, the response time.

In SEMSuS, services are implemented inside the nodes by means of language nesC [20] on the Tiny OS operating system [21], which were especially developed for constrained environments. The selection of service implementation in WSN is held in the main network component that implements the standard Interceptor, which is called the Manager.

4.1 Architecture of Middleware

The architecture of middleware is designed by specifying the provider components in the lowest level that form the provider. One of these components is the Web Server and the other is the WSN. The Provider architecture is depicted by the Fig. 3.

The Web Server component is a server that has the functions to receive requests from client applications and publish the capacity of the existing networks in the Registry. While the server does not have resource constraints, the other component that binds it (WSN) possess constraints, since it is performed by Mica nodes. Therefore, the semantics that connects them is an adapter between two different contexts.

The gateway implements the main features of the Web Server to the system. As its name suggests, it is the component for which the provider performs the interaction with the others external components. Such interactions are reflected by the publication of services in the Registry and Bind/Invocation of the client.

In WSN there are two important components: the Sink and Source Node. The Sink is the drain of the WSN data, so that any network communication with the external environment occurs through it. Usually the Sink node has more resources as compared to the Source Nodes.

In the Sink there are two logical components: the Manager and Dispatcher. The first one has the capability to manage and choose the identifiers of each service implemented by sensor nodes. Management concerns to placing an appropriate value that identifies a service so that the Dispatcher can then call run-time algorithm that really implements the service.

In sources nodes components is the implementation of services. All services follow a common interface. Then the addition of new services becomes as easy as possible. The added service must implement the interface system and have an identifier associated with it. Then the service can be invoked when the Dispatcher is

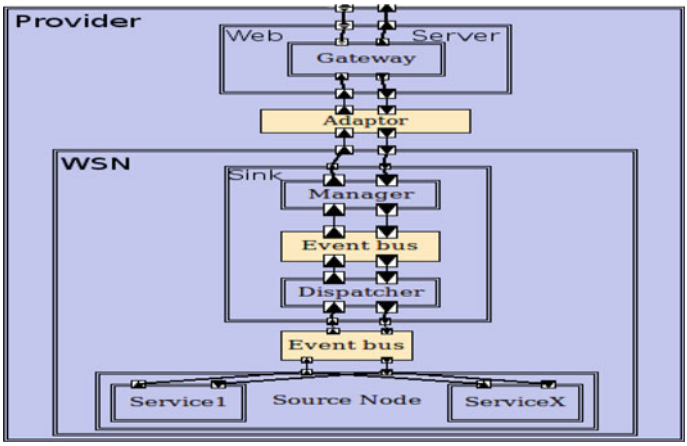


Fig. 3 Reference architecture of the provider

required. When removing a service is required to be removed its identifier, so that so that the service is no longer invoked.

The opening of the middleware architecture allows, for example, a service that monitors the state of the WSN, and to report to the Manager the current network conditions, can be added so it can choose the best implementation for the requested service. So the Manager acts as an interceptor that can interrupt the control flow allowing it to run other code.

4.2 Interaction Between the Main Components of Middleware

Figure 4 shows the interaction between the main components of SEMSuS. The process begins with the application seeking the service that can meet its requirements. This search can be performed either in a syntactic or semantic. The two options are provided through the implementation of the API (Application Programming Interface) allowing, thus, the flexibility of choice for application development. If the service is found, their identification is returned for later invocation. By holding the address of the service provider and knowing its interface, the application relies on passing the necessary data and receiving results of its implementation. In implementing the system, the Registry is performed by the jUDDI tool and the Semantic Server by Matchmaker. The jUDDI implements the syntactic matching and the Matchmaker implements the semantic matching.

Another important activity is the invocation of service for the client application. The interactions between the components involved in this process are presented in Fig. 5. It can be possible to observe that the process begins with the invocation of the application by the service provider. The first software component in the provider

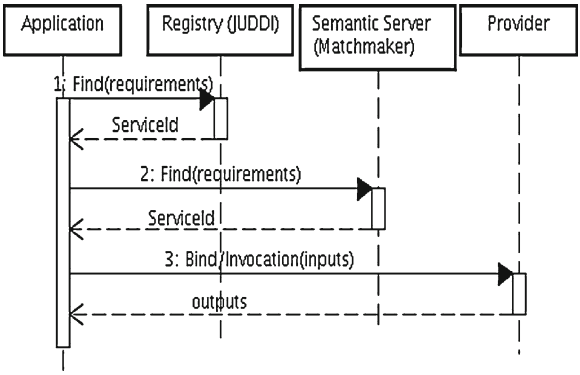
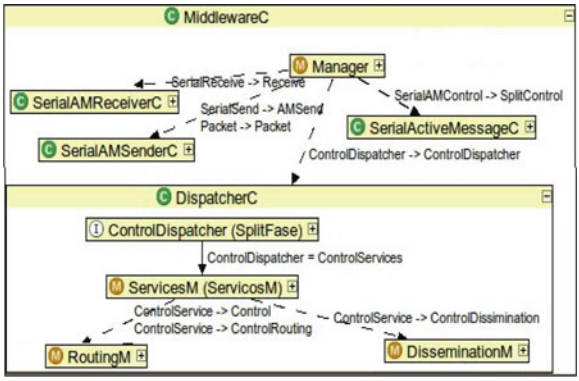


Fig. 4 Interaction between main components of the system

Fig. 5 Interaction between invocation of services



that will receive the requests is the Gateway, which is a web server Apache Tomcat. In the Gateway, the requests are transformed into messages to be sent to the WSN.

In WSN component, the Manager component receives the message from the Gateway then sends the request with the identifier (id) code that implements the service requested by the Client. The choice of service (id) depends on the current capabilities of the network. The results of the implementation of the service are returned in the opposite direction until reach the Client. Thus, the process allows the invocation of the code in a WSN as SWS.

5 Implementation and Usage Issues

To validate the SEMSuS, was implemented a scenario in which two network services are exposed as SWS. Both services have the function of communication in a WSN, but with different characteristics. One performs a communication by means of routing

and another by food of messages. Figure 6 shows the main components implemented in nesC language.

The Manager is the first component in the WSN to receive requests and it is in the sink. The Manager uses other components, such as SerialActiveMessage, to deal with the communication via the serial port, which connects the Sink to the Web Server. Because all services implement the same interface, the ControlDispacher, they can be switched without major problems.

The routing service is the Tymo, which is based on the Dymo (Dynamic MANET On-demand) for TinyOs. The Tymo inherits the main features of Dymo, such as point-to-point communication and find routes to hosts in the network. Therefore, messages for recognition of the paths are disseminated in network only when a node needs to communicate with another. The memory savings is due to small amount of data stored by the communication and energy "on-demand". Furthermore, the Dissemination service is based on the dissemination protocol to send data to all nodes in the network constituents. It works by flooding the network with data.

The routing has the advantage of saving more network resources, as it decreases in the average number of messages that are exchanged, saving bandwidth connections and energy of the nodes, when compared with the dissemination.

Moreover, the dissemination to a lower response time, since it is not necessary to exchange messages to the knowledge of the route. Thus, if the network has sufficient resources, the dissemination should be chosen to implement the service which governs the form of network communication.

In the interaction between the components of SOA, the client application uses the Matchmaker Client to conduct the semantic search, and OWL-VM to invoke the SWS that are published in OWL-S, in the registry.

6 Conclusion and Future Work

Software solutions found for distributed systems are often complex and heavy. This fact is due to the generality of Middleware transparency in dealing with the distribution to the most varied requirements of different applications. The adoption of open systems, flexible and adaptable is proving useful in the treatment of this fact.

In middleware proposed in this paper, the SEMSuS uses patterns and architectural styles consolidated to achieve openness, flexibility and adaptability. Concern to the SOA style is used as a form of interaction between the WSN and applications, specifically its implementation is done through SWS. The application is the client while WSN is the service provider. The adoption of SWS offers flexibility and dynamism. Not just the interaction between network and application are the mechanisms that promote flexibility, but also within the network. This fact is achieved by using the Manager and Dispatcher components, which implement, respectively, the patterns Interceptors and Dispatcher. As a result there is possibility of changing the implementation at runtime of the service provided by the network according to the capabilities of the network.

Within the architecture, the Manager component has key role by selecting the service implementation. This choice is deterministic. Different approaches of choice can be made to support the Manager, including some approaches a more flexible programming, such as those based on logical or on Ontologies and inference engines. Implementations of such approaches can be easily added in the SEMSuS as services, if the implementations satisfy the interface of Dispatcher.

References

1. Delicato FC (2005) Middleware-based services for wireless seNsor netwok. Doctoral thesis of the Federal University of Rio de Janeiro, Brazil, Rio de Janeiro, p 53, June 2005
2. Capra L, Emmerich W (2001) Mascolo C reflective middleware solutions for context-aware applications. Proceedings of the reflection 2001, Lecture notes in computer science 2192, Springer Verlag, Japan, pp 126–133
3. Gay D, Levis P, Culler D (2007) Software design pa tterns for TinyOS. ACM Trans Embed Comput Syst (TECS) 6(4):22-es. doi:10.1145/1274858.1274860
4. Markus V, Michael K, Uwe Z (2004) Remoting patterns: foundations of enterprise, internet, and realtime distributed object middlware. Wiley, England, pp 130–133
5. Wang M, Cao J, Li J, Dasi SK (2008) Middleware for wireless sensor networks: a survey. J Comput Sci Technol 23:305–326
6. Cary P, Jorge C, John AM, Richard SP, Ivan V (2007) Introduction to web services. In: Cardoso J (ed) Semantic web services: theory, tools and applications. IGI Global, Hershey, pp 134–154, March 2007
7. Extensible Markup Language (XML) 1.0 (Fifth Edition). http://www.w3.org/TR/RECxml/. Accessed 29 Oct 2009
8. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). http://www.w3.org/TR/soap12-part1/. Accessed 29 Oct 2009
9. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl. Accessed 29 Oct 2009
10. Borst WN (1997) Construction of engineering ontologies for knowledge sharing and reuse. Doctoral thesis of the University of Tweenty, Enschede, The Netherlands
11. Almeida MB, Bax MP (2003) An overview about ontologies: survey about definitions, types, applications, evaluation an building methods. Brasilia 32:3, 7–20. doi:10.1590/S0100-19652003000300002
12. Agilla: a mobile agent middleware for wireless sensor networks. http://www.cs.wustl.edu/mobilab/projects/agilla. Accessed 29 Oct 2009
13. Levis P, Culler DE (2002) Mat: a tiny virtual machine for sensor networks, architectural support for programming languages and operating systems. http://www.cs.berkeley.edu/pal/pubs/mate.pdf
14. TinyDB: a declarative database network. http://telegraph.cs.berkeley.edu/tinydb. Accessed 19 Oct 2009
15. TinyLime. http://lime.sourceforge.net/tinyLime/index.html. Accessed 19 Oct 2009
16. Iqbal M, Lim HB, Wang W, Yao Y (2009) A service-oriented model for semantics-based data management in wireless sensor networks. In: Proceedings of the 5th IEEE international workshop on heterogeneous wireless networks (HWISE 2009), May 2009
17. Rocha AR, Delicato FC, de Souza JN, Gomes DG, Pirmez L (2009) A semantic middleware for autonomic wireless sensor networks. In: Proceedings of the 2009 workshop on middleware for ubiquitous and pervasive systems (WMUPS '09), vol 389. ACM, Dublin, Ireland, New York, NY, pp 19–25, June 16–16. doi:10.1145/1551693.1551697
18. Java ME. http://java.sun.com/javame/index.jsp. Accessed Oct 2009

19. OWL Web Ontology Language. http://www.w3.org/TR/owl-features/. Accessed 29 Oct 2009
20. Gay D, Levis P, von Behren R, Welsh M, Brewer E, Culler D (2003) The nesC language: a holistic approach to networked embedded systems. In: Proceedings of the programming language design and implementation (PLDI), June 2003
21. TinyOs. http://www.tinyos.net/