Reflective Middleware for Wireless Sensor Networks

Flávia C. Delicato, Paulo F. Pires, Luiz Rust, Luci Pirmez
NCE – Federal University of Rio de Janeiro (UFRJ)
P.O Box 2324, Rio de Janeiro RJ, 20001-970, Brazil
+ 55 21 2598-3158
{fdelicato,paulopires,rust,luci}@nce.ufrj.br

José Ferreira de Rezende
GTA - UFRJ
P.O. Box 68504, RJ, 21945-970 - Brazil
+ 55 21 2562-8645
rezende@gta.ufrj.br

ABSTRACT
Wireless Sensor Networks (WSNs) are distributed systems whose main goal is to collect and deliver data to applications. This paper proposes a reflective, service-oriented middleware for WSN. The middleware provides an abstraction layer between applications and the underlying network infrastructure and it also keeps the balance between application QoS requirements and the network lifetime. It monitors both network and application execution states, performing a network adaptation whenever it is needed. Simulation results show that the network residual energy can be increased in more than 100% when adopting an adaptation strategy, while the application QoS requirement is respected.

Categories and Subject Descriptors
C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms
Design, Standardization, Management

Keywords
Wireless Sensor Networks; Reflective Middleware; Adaptation.

1. INTRODUCTION
Wireless Sensor Networks (WSNs) constitute a new domain of distributed computing that attracted great research interest over the last years. WSNs are composed of battery-operated devices, with reduced processing and storage capabilities, linked by wireless connections. WSNs main goal is to collect environmental data and to deliver them to client applications. Several protocols for WSNs have been proposed to support the delivery of the collected data in a robust and energy efficient way [7]. As the most expensive resource in WSNs is radio communication, and sensor nodes usually have spare processing capacity, they can perform some computation, such as data aggregation and route optimization that reduces the amount of communication.

The emergent class of WSNs applications has very specific features and requires some minimum level of Quality of Service (QoS). QoS requirements and other characteristics, such as the required data delivery model, are highly dependent on each application. The choice of the most appropriate communication protocol for each application can lead to an optimal network global performance, in terms of energy consumption and, consequently, of the network lifetime.

The use of middleware systems in traditional computing environments facilitates the work of application developers, unburdening them from dealing with the native complexity of distributed systems. Similarly, WSN systems could make use of a middleware layer to implement some specific funcionalities, such as the selection of the appropriated network protocol and the necessary sensor node management. However, in the current middleware technologies the intrinsic complexity of a distributed system is hidden from user and developers. Completely hiding the details of network operation from applications may be inefficient in WSNs. WSN applications need to detect and react to changes in the environment (regarding connectivity, bandwidth and available energy), as well as to changes in the application interests. The application ability for changing the network behavior can be fundamental to improve the system performance.

Therefore, a middleware layer should act as a broker between applications and the WSN, translating application requirements into WSN configuration parameters. Due to the dynamism of WSN environments, applications should have some degree of context awareness to best reach their QoS requirements. The middleware should supply mechanisms that allow the application to monitor the network state through a high level interface. Middleware systems that provide context awareness, instead of transparency, are known as reflective middleware systems [2].

This paper proposes a reflective and service-oriented middleware for WSNs. The middleware provides an abstraction layer between applications and the underlying network infrastructure. Besides supplying an abstract programming model to WSN applications, it keeps the balance between application QoS requirements and the network lifetime. Such middleware is in charge of decisions about communication protocols, network topologic organization, sensor operation modes and other infrastructure functions typical of WSNs. The middleware monitors network and application execution states performing a network adaptation whenever it is needed, with or without application interference.

The main contributions of the proposed middleware are three-folded. First, the system provides an interoperability layer between different applications and the WSN. Applications can use the network through different protocols and organizations, without caring about the details of network operation. Second, the services provided by the middleware are accessed in a flexible

SAC'05, March 13-17, 2005, Santa Fe, New Mexico, USA.

way through a standard and high-level language. The use of XML [9] and SOAP [8] in the network communication supplies a maximum of flexibility and interoperability. Finally, the provided services of decision about network configuration and of dynamic adaptation with context awareness aim to increase the network global lifetime, while meeting the applications requirements.

The remaining of this work is organized as follows. Section 2 presents related works. Section 3 describes the proposed system, its architecture and operation. Section 4 presents the system evaluation and Section 5 draws the concluding remarks.

2. Related Works

In [1] and [11] a database approach is proposed for designing WSNs. In contrast, our work adopts a service-based approach, in which interactions between applications and the network are based on a consumer-provider relationship. Such approach is inspired from the Web Services area [4] and it exploits several companion technologies, as WSDL [4] and SOAP [8]. Another feature of our work is the adoption of the ubiquitous XML and SOAP standards as message formatting and data representation mechanisms, instead of relying in proprietary languages, such as proposed in [1]. We also provide mechanisms for system adaptation, context-awareness and to adapt the WSN configuration to each particular application requirements.

MILAN [6] is a middleware for WSNs that receives a description of application requirements and chooses the best sensor and network configuration matching such requirements while maximizing the network lifetime. It does not address the issue of providing a standard representation for application requirements and sensor data. In its first version, MILAN adopts a totally centralized approach, which is not appropriate for large WSNs. CARISMA [2] is a middleware for mobile computing which exploits the principles of reflection to support the building of context-aware and adaptive mobile applications. The main difference from our proposal is that our system is tailored for the specific requirements of WSNs, addressing most of their generic infrastructure services, such as network protocol configuration, data aggregation and sensor management. CARISMA is target to generic mobile and wireless network applications.

3. System Description

The proposed middleware receives a description of the application requirements, monitors the network and application states, and optimizes the sensor and network configurations trying to maximize the system lifetime while guaranteeing that the application requirements are met.

WSN applications execute in extremely dynamic physical environments. The available resources, as bandwidth and residual energy, can also vary, as well as the application interests and QoS. Therefore, to achieve a reasonable QoS, applications need to be *aware* of their execution context [2]. To allow context awareness, the system behavior with respect to each specific application is described in a profile. A profile defines, for each service requested by the application, different execution policies. Each execution policy is represented by a set of QoS requirements to be met in each different execution context foreseen by the application developer.

The middleware keeps a valid representation of the execution context by directly interacting with the underlying network protocols and devices. Whenever a significant change is detected in the context, the middleware searches the corresponding application profile to find out how the system should to behave in such configuration. Whenever requirements defined for the current execution context are not being met, the middleware activates a pre-defined adaptation policy to adjust the system.

As both the user needs and the execution context change quite frequently, an initial profile is built from the application inputs, and applications are provided with dynamic access to this profile through the inspection capacity of the reflective middleware. Each application should decide whether and when to use the reflection capacity of the middleware to customize the system behavior.

3.1 System Architecture

WSNs are composed of hundreds of sensor nodes, and one or more sink nodes. Sink nodes are computational powerful, not energy constrained devices, acting as entry points for applications requests and gathering points for sensor-collected data**.**

The proposed middleware has been designed using a service-oriented approach [4]. For an external point of view, applications are service requestors and sink nodes are service providers. Sink nodes release the descriptions of the services provided by the WSN and offer access to these services. From an internal point of view, sinks are the service requestors and sensor nodes are the service providers. Sensors send the descriptions of their services to sink nodes, which keep a repository of the service descriptors of each type of existing sensor in the network.

The main architectural components of the proposed system are the communication module and the middleware services.

3.1.1 Communication Module

This module is composed of a message router, a set of handlers, a set of XML drivers and a SOAP Proxy. The communication module in sink nodes is based on the SOAP protocol. To avoid the overhead incurred by the adoption of SOAP, inside the network all data messages are represented in XML language, encapsulated through a specific and compact format.

SOAP proxies are programs that translate function calls made in the application programming language to SOAP messages and vice-versa. XML drivers are programs that translate the middleware XML messages to the underlying network protocol language and vice-versa. The message router coordinates the flow of SOAP or XML messages through the several handlers. Handlers represent the message processing logic and act as dispatchers to the middleware services. Basic handlers are responsible for parsing and composing XML messages, header processing and data type conversions. Specific handlers are defined for each implemented middleware service. Three specific handlers are currently implemented. The *Matching_Data* handler accomplishes a matching function among sensor generated data and previously received application interests. The result of the matching is used by the routing protocol in its forwarding decisions. The *Parse_Interest* handler accomplishes a matching function among the sensor features (type and geographical location) and a received interest. The *Matching_Aggregation* handler represents the activation of aggregation functions implemented as service components of the middleware. Requests

for aggregation functions are sent as part of the interests submitted by the application and are registered in nodes that participate in the current network task.

3.1.2 Middleware Services

The basic service provided by a WSN is the delivery of sensing data. The proposed middleware provides the application with an abstraction of such service so that the system is able to control the different execution policies that can be requested.

The middleware also offers a decision service to facilitate the application development; services to assist generic application needs and an adaptation service to provide the dynamic behavior of the system. The several services provided by the middleware are implemented as modular software components.

Generic services in a WSN are: naming, location, security, aggregation and service discovery. The naming service is used whenever a unique local identification for each node of the network is required. The localization service is useful when some nodes of the network do not have GPS receiver. The aggregation service implements aggregation functions to be applied to sensor-collected data to reduce the redundancy among data and to save transmission energy. The use of XML allows the service components to be implemented in any programming language and easily "plugged" in the middleware.

3.2 System Operation

3.2.1 Service Discovery

Before system start-up, it is necessary to discover the services provided by the WSN. Two levels of service discovery are required. An internal level is required by sink nodes to discover the characteristics of the sensor nodes in the network. Internal discovery starts with a configuration phase, during which nodes exchange XML messages describing their services. Configuration messages include the sensor identifier and type, geographical location, residual energy, maximum & minimum confidence degrees. Configuration messages are broadcast to reach at least one sink node in the network. Sink nodes store the contents of received configuration messages in a local repository.

The external discovery is used by an application to find out which WSN supplies the required services and how to invoke them. The use of SOAP and XML, both part of the Web Services architecture [4], makes the UDDI [4] the natural choice for the service discovery protocol. After using UDDI to find the WSN, the application obtains through the sink the WSDL document that describes the interface of the network services.

3.2.2 Submission of Interests and Requirements

The WSDL provided by the sink node tells to the application how to invoke the services available in the WSN. The application submit its interests using different types of SOAP messages for interest advertisement [3], which contain the sensor type, the geographical coordinates of the target area, the data acquisition duration and rate, data aggregation functions, among other information. Beyond interests, applications send, for each requested service, a list of execution policies, indicating the QoS parameters to be respected in each execution context, during the delivery of the service.

3.2.3 Network Configuration

The middleware provides a service of automated decision to select the best protocol/topology to be used in the network for a given application. A decision algorithm [3] is responsible for choosing the best data dissemination strategy and network topology, based on the information contained in the SOAP message of advertisement interest. Choices are based on the results of previous experiments and simulations.

3.2.4 Creating an Application Profile

The application profile is an XML-based data structure which contains: (i) the description of the application interests, extracted from the SOAP message of interest and (ii) the execution policies for the requested service, i.e. QoS requirements for each execution context. The middleware inserts its decisions about protocol and topology configuration in the application profile. Furthermore, the different adaptation policies applied during the execution of the application can be included in the profile.

QoS requirements for the service of data delivery can be defined in terms of delay, accuracy and data rate. QoS requirements for the aggregation service are defined according to the aggregation degree (ratio of the number of received messages to the number of sent messages) and the aggregation delay (delay incurred by the time a sensor node should wait for incoming messages to be aggregated before their transmission).

The execution context is represented by a set of application and system parameters. Application parameters are information known by the current application and are defined in terms of the values of sensing data [6]. System parameters regard information known by the middleware and include: battery level; power/transmission rate and geographic location of nodes.

3.2.5 Data Dissemination

Whenever a sensor node generates data, they are transferred to the communication module, where an XML driver converts them into an XML representation. The *Matching_Aggregation* handler verifies if the data attributes match some aggregation service request. If there is a match, the data is dispatched to the respective aggregation service. The aggregated data is forwarded to the dissemination protocol, as a new XML message of data advertisement.

When receiving a packet from another sensor, the dissemination protocol verifies if the packet contains an application message (data or interest advertisement) or an infrastructure (control) message. In case of an application message, it is transferred to the communication module. A basic handler is responsible for verifying the message type (interest or data) and for dispatching it to the specific handler, which process the message and forwarding to the respective services indicated in the message.

3.2.6 System Inspection and Adaptation

The middleware inspection capacity allows an application to request information on the current execution context (reification process [2]). The request and the respective response are represented as SOAP messages. From the analysis of the provided information, the application may decide to modify the system behavior, changing some previously registered QoS parameter or execution policy. The adaptation module keeps a table to register

the parameters that each application requests to monitor. Monitoring components existent in the sensor nodes periodically check the values of requested parameters.

Adaptation policies are pre-registered in the system as sets of actions to be performed when the application QoS requirements are not being fulfilled for a given execution context. Adaptation policies created for the proposed middleware are: (i) increase the data reliability (data accuracy); (ii) decrease the energy consumption; (iii) increase the available bandwidth. A policy of decreasing the energy consumption may be implemented by two actions: decreasing the data rate and turning off some sensors.

4. System Evaluation

The proposed system was evaluated according to two different approaches: a qualitative and a quantitative. The goal of the qualitative evaluation was to implement a prototype of the proposed middleware as a proof of concept for the system. The prototype was developed using Java Language and the Apache Axis Web services platform [10]. The prototype is available from http://www.nce.ufrj.br/labnet/research/sensornet/software.htm.

In the quantitative evaluation we performed simulations with two mains goals: (i) to show that the adoption of a network protocol meeting the specific application requirements could improve the WSN overall performance, in terms of energy consumption; and (ii) to prove that gains in the WSN lifetime could be obtained through the adoption of an adaptive behavior, in contrast with the static configuration of the network.

The WSN performance can be evaluated according to different metrics [7]. In the first stage of simulation, we were interested in measuring the average dissipated energy, which is a ratio of the total dissipated energy per node in the network to the number of distinct events delivered to the sink nodes, and it is directly related to the WSN lifetime. In the second stage we addressed the tradeoffs between QoS requirements issued by an application and the overall energy consumption in the WSN.

4.1 Quantitative Evaluation

In the first stage of evaluation we adopted a data dissemination protocol for WSN called directed diffusion (DD) [7], and simulated different configurations of such protocol, according to different application requirements. We implemented a driver to convert XML messages to the format adopted by DD and vice-versa. We implemented a sub-part of the decision algorithm used by the middleware decision service, for deciding between two operation modes of DD: pull and push [5]. Each mode is more suitable for a different set of application requirements. The simulations were performed in the network simulator NS-2 (http://www.isi.edu/nsnam/ns). Results are reported in [3]. We found improvements of up to 25% in the average dissipated energy when selecting the DD operation mode according to the application, in a transparent way for application developers.

In the second stage of evaluation our goal was to measure the gains obtained with the adoption of an adaptation policy provided by the middleware.

For many applications of environmental monitoring, data accuracy is more important than transmission delay. WSNs have a high density of nodes, generating high redundancy in the reported data. Such redundancy can be exploited as a mechanism for fault tolerance, or to increase the accuracy of the supplied data. However, if all available nodes are always kept actives, the network lifetime will decrease. Therefore, data accuracy can be traded off against energy consumption, and the percentage of active nodes (network logic topology) can be a parameter of an action activated to implement a middleware adaptation policy.

An application was simulated using the value of data accuracy as its QoS requirement for the data delivery service. To guarantee the requested accuracy, the application defined in its profile two values for error. The error was represented as the mean square error (MSE), given by the difference between a set of values assumed as the "real" values and the value generated by each sensor, considering its nominal precision. The first value defined in the profile (zero) was the optimum value, providing the maximum accuracy about the phenomenon. Such value was to be used when the total energy spent by the network was below a specified threshold. The second value (0.05) provided the minimum accuracy tolerated by the application, and should be used when the total energy spent exceeded the specified threshold. The goal of the adaptation service was to monitor the network residual energy in each given interval of time. Whenever the energy exceeded the threshold (established as 60% of the network initial energy), an adaptation policy was applied to reduce the energy consumption. The action to implement such a policy was to reduce the percentage of active nodes. Such reduction was gradual, and the accuracy values supplied in each case were monitored to guarantee the requested QoS.

A sensor field was created with 150 nodes randomly distributed in a square area with 400m x 400m. Each node had a radio range of 40m and a sensing range of 20m. The energy dissipation model adopted by the radio is described in [7]. The initial energy of all sensors was 20J. Sensors that generated data (sources) were randomly selected from nodes in a 100m by 100m square within the field. Temperature data was generated every 10 seconds. XML messages for data advertisement had length of 130bytes (header plus data). Application was interested in the average temperature in the target area, over each cycle of monitoring. A single sink node was placed in the central part of the sensor field. In this evaluation stage, we were not interested in simulating any specific communication protocol. Therefore, we assumed hypothetical protocols, delivering data generated from sources to the sink node through the shortest path, without data loss. The JIST simulator (http://jist.ece.cornell.edu/) was used in this stage.

Simulations run for 1000 seconds, divided in 5 cycles, at the end of which the network residual energy was obtained and the mean square error was calculated. In the first cycle, all nodes were selected to be active. In subsequent cycles, if the threshold of energy consumption was exceeded, the percentage of active nodes was decreased, initially to 90%, then 80%, and so forth, while the accuracy of generated data was monitored. If the error value got greater than the requested by the application, previously disabled nodes were reactivated. To select the active nodes in each cycle, a greedy algorithm was adopted, which gave priority to keep active nodes with larger relevance for the application (for being source nodes) or for data routing. The number of sources was varied, to analyze its influence in the economy of energy obtained with the adaptation policy. Figure 1 shows the network residual energy at the end of the simulation, as a function of the number of sources. Table 1 shows the obtained error (mean square error) in each case. The graph shows that, with a static approach for network

configuration, the residual energy of the network, and therefore its lifetime, decreases proportionally to the number of sources. By adopting the adaptation policy, the network lifetime can be extended, while the error is kept under the limit dictated by the application.

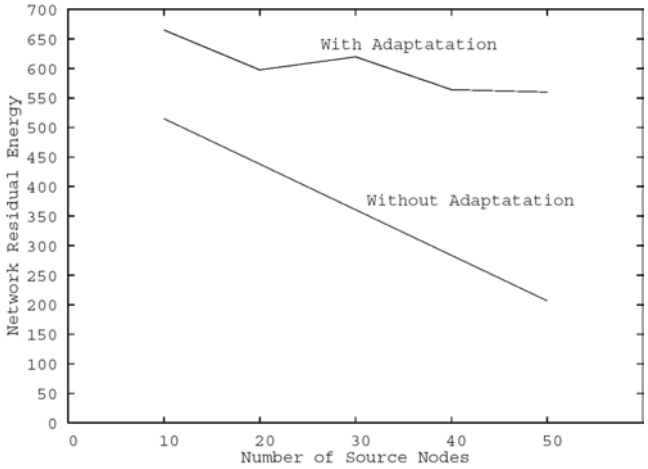

Figure 1. Residual energy for different number of sources

Table 2 shows values of residual energy and error for different percentages of active nodes. Results show that the network residual energy can be aggressively increased when activating up to 70% of nodes, while the error is maintained inside of the requested limits.

Table 1: MSE

Num of Sources	MSE – Mean Square Error	
	With Adaptation	Without Adaptation
10	0.0	0.0
20	7.78E-4	0.0
30	0.0038	0.0
40	0.006	0.0
50	0.023	0.0

Table 2: MSE and residual energy

Active Nodes %	MSE	Network Residual Energy
100	0.0	350.877
90	0.0	576.203
80	0.0008	868.936
70	0.0003	1131.429
60	0.07	1404.824
50	0.1	1600.050

5. Concluding Remarks

The mis-en-oeuvre of the proposed middleware allows us to cover the large spectrum of WSN applications requirements, providing both QoS and context-awareness, but still satisfying the needs of handling efficiently energy issues. The following features and strategies have been selected to support these goals: (i) the use of a high-level interface to interact with applications and to acquire their requirements; (ii) the use of profiles to map QoS parameters into execution contexts, and (iii) the employment of components for state monitoring, system inspection and policies activation.

Since the middleware decides about network infrastructure on behalf of the application, it facilitates the job of WSN application developers. The adoption of XML language and SOAP protocol, supplies a high degree of flexibility and interoperability to the system. New components may be implemented by third parties and seamlessly incorporated to the middleware architecture. Applications may be written in whichever high level programming language and interact with the WSN through the support of automated tools for SOAP message generation.

The middleware services for network configuration and adaptation were evaluated and the results were very promising. Concerning the network configuration service, improvements of up to 25% in the network average dissipated energy were achieved when the parameters of the communication protocol are selected according to the application needs. The middleware decision module, without any application interference, has integrally carried out such selection. Using the adaptation service, results shown that up to 100% lesser energy consumption can be obtained, while still satisfying application QoS requirements. Further simulations will be performed to evaluate other adaptation policies and the participation of application in the system behavior, through the middleware capacities of reflection and inspection.

6. ACKNOWLEDGMENTS

This work has been supported by CNPq, CAPES, FAPERJ and RNP/FINEP/FUNTEL.

7. REFERENCES

[1] Bonnet, P., Gehrke, J. E., Seshadri, P.: Towards Sensor Database Systems. In *Procs. of the 2nd International Conference on Mobile Data Management*. Hong Kong, Jan 2001.

[2] Capra, L., Emmerich, W., Mascolo, C.: CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications. *IEEE Transac. on Software Engineering*, 29(10):929-945, 2003.

[3] Delicato, F. et al.: Service Oriented Middleware for Wireless Sensor Networks. *TR. NCE04/04*. Available in http://www.nce.ufrj.br/labnet/research/sensornet/publications.htm, 2004.

[4] Graham, S. et al.: Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI. Sams Pub, 2002.

[5] Heideman, J., Silva, F., Estrin, D.: Matching Data Dissemination Algorithms to Application Requirements. ISI-TR-571, 2003.

[6] Heinzelman, W. et al.: Middleware to Support Sensor Network Applications. *IEEE Network Magazine Special Issue*, Jan. 2004.

[7] Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Procs. of the ACM/IEEE MobiCom*, pp. 56-67, USA, Aug. 2000.

[8] W3C Note on Simple Object Access Protocol (SOAP) 1.1. Available in: http://www.w3.org/TR/SOAP/, May 2000.

[9] W3C Recommendation. Extensible Markup Language (XML) 1.0 (Snd Edition). Available in: http://www.w3.org/TR/REC-xml, 2000.

[10] Web Services Project Apache. Available in: http://ws.apache.org.

[11] Yao, Y., Gehrke, J. E.: The Cougar Approach to In-Network Query Processing in Sensor Networks. *Sigmod Record*, Vol. 31, Num 3, Sept. 2002.