Cluster Computing 8, 211–221, 2005

A Service Approach for Architecting Application Independent Wireless Sensor Networks

FLÁVIA COIMBRA DELICATO *
Núcleo de Computação Eletrônica—NCE, Federal University of Rio de Janeiro, P.O. Box 2324, Rio de Janeiro, RJ, 20001-970, Brazil

PAULO F. PIRES†
Núcleo de Computação Eletrônica—NCE, Computer Science Departament–DCC, Federal University of Rio de Janeiro, P.O. Box 2324, Rio de Janeiro, RJ, 20001-970, Brazil

LUCI PIRMEZ[1] and LUIZ FERNANDO RUST DA COSTA CARMO
Núcleo de Computação Eletrônica—NCE, Federal University of Rio de Janeiro, P.O. Box 2324, Rio de Janeiro, RJ, 20001-970, Brazil

Abstract. The current sensor networks are assumed to be designed for specific applications, having data communication protocols strongly coupled to applications. The future sensor networks are envisioned as comprising heterogeneous devices assisting to a large range of applications. To achieve this goal, a new architecture approach is needed, having application specific features separated from the data communication protocol, while influencing its behavior. We propose a Web Services approach for the design of sensor network, in which sensor nodes are service providers and applications are clients of such services. Our main goal is to enable a flexible architecture in which sensor networks data can be accessed by users spread all over the world.

Keywords: wireless sensor networks, Web services, XML

1. Introduction

Many scientific applications require the acquisition of precise data measurements over time for a large geographic region of interest. While these measurements can sometimes be accomplished at a distance using remote sensing techniques, it is often necessary to collect data using in-situ sensing, where sensors are placed directly in the target area [29]. There already exists this kind of environmental sensor in operation, as for example the system developed for NASA for monitoring the lands of Mars [29]. Although such system can carry out multiple measures, its capability is limited to collect spatial and temporal discrete events. The next step in the area of environmental sensing is to be able to capture geographically distributed measurements simultaneously for long periods of time. One potential solution for such application scenario is the design of large multi-point sensor networks comprised of nodes with sensing, processing (elementary) and communication capabilities. Such systems can use hundreds or thousands of sensor nodes, interconnected by a wireless network and play the role of a highly parallel, accurate and reliable data acquisition system.

Typically, sensors are devices wich bare limited energy and processing capabilities, deployed in an ad-hoc fashion, and they have to operate unattended, since it is unlikely to handle a large number of nodes in remote, possibly inaccessible locations. Therefore, energy saving is a crucial requirement for such an environment.

Sensors data are transmitted from multiple acquisition sources toward one or more processing points, which may be connected to external networks. Since sensors monitor a common phenomenon, it is likely that significant redundancy among data generated from different sensors would appear. Such redundancy can be exploited to save transmission energy, throughout filtering and data aggregation procedures in-network. To save further energy, the short-range hop-by-hop communication is preferred over the direct long-range communication to the final destination. Thus, nodes send their own data and their neighbors' data through paths, preferably optimized, to some exit point in the network.

For some classes of applications, such as environmental monitoring, the aggregated information from multiple nodes in a geographical area of interest is more important than data from an individual node. Other applications, such as parking-lot networks, can require the identification of individual nodes. Regardless the class of application, it is more useful to hava nodes identified by the type of sensor device (data type) or by their geographical location. Several works [13,16] have suggested the use of data-centric naming systems, instead of traditional address-centric schemes, like IP. In the data-centric approach, nodes are addressed by attributes, such as the type of data they provide, or by their interest in some type of sensing data.

Current works [3,13,15,16] consider sensor networks as being designed for specific applications, with data communication protocols strongly coupled to the application. In fact,

* Corresponding author.
E-mail: fdelicato@nce.ufrj.br
†CAPES-Brazil grant holder.

the network requirements, organization, and routing behavior change according to the application. In spite of the application specific behavior of the current sensor networks, many authors [22] envision the future sensor networks as being composed of heterogeneous sensor devices and assisting to a large range of applications, for different groups of users. To achieve this goal, a new architectural approach is needed, in which application specific requirements are separated from the data dissemination functions. In such architecture, the components should be loosely coupled, having well defined interfaces. To achieve energy efficiency, applications should be able to dynamically change the network behavior. However, these changes should be expressed in a powerful and flexible way, through a common protocol, preferably accepted as a ubiquitous standard. Such features will allow the design of networks to be independent from the applications that will use them.

We propose a service approach for the design of sensor networks. Services are defined as the data provided by sensor nodes and the applications (for instance, a filtering program) to be executed on those data. Clients access the sensor network by submitting queries to those services.

Services are published and accessed by using by using the Web services technology [4]. By adopting the Web Services paradigm, we propose a novel architecture for sensor networks, in which the Web Services Description Language (WSDL) [31] is used to describe data and functionalities of sensor nodes. Sink nodes are Web Services that offer a standard interface for accessing services which are provided by the network. Queries submitted by user applications are accomplished through such an interface.

WSDL is an XML-based language (Extensible Markup Language [33]) used for describing services available on the Web, named *Web services*, in a standardized way. In the same way as middleware systems like COM [18] and CORBA [19] use interfaces, a WSDL document is a contract between service providers and their clients.

Web Services builds on SOAP [37] protocol's capability for distributed, decentralized network communication by adding new protocols and conventions that expose users functions to interested parties over the Internet from any Web-connected service [4]. Any software component or application can be exposed as Web services so that it can be discovered and used by another component or application. One important point is that a Web Service, despite of its name, needs not necessarily exist on the World Wide Web. A Web Service can live anywhere on the network (Inter- or intranet).

Using specific routing protocols for sensor networks, such as direct diffusion [13], we intend to offer a flexible and powerful way of manipulating, extracting and exchanging data in a sensor network. Applications access the sensor network and modify the underlying data dissemination behavior through an interface which is bith common and application independent. Such an interface is provided by the Web Services available on sink nodes.

Our approach enables the construcion of generic sensor networks which are capable of meeting the requirements of a large range of independently designed applications. The use of standard protocols provides the necessary mechanisms to enable the interoperability among different networks. Besides, since users and applications access the sensor network through a common service interface, they are shielded from the physical details of contacting the relevant sensor nodes, processing the sensor data and sending back the results. Promoting the independence between sensing data and the presentation of such data to a given user, a single set of sensors, connected through a communication framework, is able to provide different "views" of information generated by the sensor network.

The present work defines the architectural components as well as the WSDL elements which are needed to implement a direct diffusion scheme in the proposed architecture. The article is organized as follows. Section 2 covers the background concepts. Section 3 presents the related work. Then, Sections 4 and 5 detail the system architecture and description. Finally, Section 6 outlines the conclusions and future works.

2. Background

Wireless sensor networks represent an increasingly important example of distributed event systems [10]. Most of these networks work as a reliable data capture network. Data are collected in the distributed sensors and relayed to a small number of exit points, called sinks, for further processing.

Since the energy saving is a crucial requirement for sensor networks, the short range hop-by-hop communication is preferred over direct long-range communication to the destination. Therefore, the dissemination of information is carried out by passing data through the several nodes which perform measurements and relay data through neighboring nodes until reaching one or more sink in the network. Data sent by different nodes can be aggregated in order to reduce redundancy and minimize traffic and thus energy consumption. To enable data aggregation in network in an efficient way, application-specific code, such as data caching and collaborative signal processing are to occur as close as possible to where data is collected. Such processing depends on attribute-identified data to trigger application-specific code and hop-by-hop processing of data [9].

Attribute-based naming systems offer a data-centric approach which differs from the traditional address-centric approaches. Data-centric communication and application-specific processing are the trend of networking service design in sensor networks. Resource management and energy-efficient routing have to be done tightly-coupled with application in order to achieve energy-efficient communication [22]. The next section briefly discusses the concept of data-centric communication.

2.1. Data-centric communications in wireless sensor networks

Data-centric addressing has been proposed for sensor networks, in which nodes are identified by the data generated by them or by their geographic location. SPIN (*Sensor Protocols for Information via Negotiation*) [16] and directed diffusion [13] are examples of data-centric protocols.

Directed diffusion [13] is an example of a data-centric protocol specifically designed for sensor networks. In directed diffusion, individual nodes reduce the sampled waveform generated by a target into a relatively coarse-grained "event" description. Such description contains a set of attributes. Applications which request data send out interests through some sink in the network. These interests are also represented as a set of attributes. If the attributes of source nodes generated data match these interests, a gradient is setup within the network and data will be pulled toward the sinks. Therefore, it is essentially a receiver-initiated routing protocol. Intermediate nodes are capable of caching and modifying data. Directed diffusion also facilitates the design of energy-efficient distributed sensing applications. It provides Geographic and Energy Aware Routing Protocol [39], which helps to define a closed geographic region for propagating *interests* that improves performance by avoiding the *interest* messages to flood the entire network.

SPIN [16] has several similarities with directed diffusion. Data are named by using application-specific and high-level data descriptors, the *meta-data*, similar to the *interest attributes* used in directed diffusion. SPIN uses meta-data negotiations to eliminate redundant data transmissions over the network. In the SPIN protocol, nodes which have new data *advertise* the data to the neighboring nodes in the network by using meta-data. When the neighboring node wants this kind of data, it sends a *request* to the initiator node for the data. The initiator node responds and *sends* data to the sinks. Therefore, SPIN is essentially a *sender-initiated* routing protocol.

Despite the advantages of data-centric addressing in sensor networks, recent works [13,16] assume that data representation is totally application-specific or offer schemes with low flexibility and expressivity. Such current approaches require a strong coupling between the model that is adopted for data and interest representation and the application querying the network.

We can envision a class of future sensor networks as being accessed by several different applications submitting queries through arbitrarily localized sinks (probably through the Internet). To enable such a scenario, the network should be accessed through a common and application independent interface.

The present work suggests a Web services approach for architecting wireless sensor networks. We propose the use of a service description language—WSDL [31] and the associated protocol—SOAP [37], both accepted as Internet standards, as being the basis for describing and communicating data and interests on a flexible way in a sensor network. The next section briefly introduces the Web services technology.

2.2. *The Web services technology*

Web services can be define as modular programs, generally independent and self-describing, that can be discovered and invoked across the Internet or an enterprise intranet. Like components, Web services expose an interface that can be reused without worrying about how the service is implemented. Unlike current component technologies, Web services are not accessed via protocols dependent on a specific object-model. Instead, Web services are accessed via ubiquitous Web protocols and data formats, such as Hypertext Transfer Protocol (HTTP [6]) and XML [33], which are vendor independent.

Web Services Description Language (WSDL) [31] is an XML language for describing the interface of a Web service enabling a program to understand how it can interact with a Web service. Each Web service publishes its interface as a WSDL document (an XML document) that completely specifies the service's interface so that clients and client tools can automatically bind to the Web service. A WSDL document defines services as collections of network endpoints or ports [31]. Besides, messages and port types are defined. *Messages* are abstract descriptions of the data being exchanged, and *port types* are abstract collections of operations. In WSDL, there is a separation between the *abstract* definition of messages and their *concrete* network implementation. This allows the reuse of abstract definitions of *messages* and *port types*. The concrete protocol and data format specification for a particular port type defines a reusable *binding*. A *port* is specified by associating a network address with a reusable binding. A *service* is defined as a collection of ports.

We can say that WSDL is a protocol specification language. It is precisely what we need if we are to go beyond "fixed" protocols such as IP and HTTP towards application-specific protocols.

The SOAP protocol extends XML so that computer programs can easily pass parameters to server applications and then receive and understand the returned semi-structured XML data document. The SOAP specification has four parts [37]. The SOAP *envelope* construct defines an overall framework for expressing what is in a message, who should deal with it, whether it is optional or mandatory, and how to signal errors. The SOAP *binding framework* defines an abstract framework for exchanging SOAP envelopes between peers using an underlying protocol for transport. The SOAP *encoding rules* defines a serialization mechanism that can be used to exchange instances of application-defined data, arrays, and compound types. The SOAP *RPC representation* defines a convention that can be used to represent remote procedure calls and responses.

The Web services technology is based on a flexible architecture named SOA (service-oriented architecture [8]), which defines three roles: a service requestor, a service provider and a service registry.

A service provider is responsible for creating a service description, publishing that service description to one or more service registries, and receiving Web services invocation messages from one or more service requestors. A service requestor is responsible for finding a service description published to one or more service registries and for using service descriptions to invoke Web services hosted by service providers. Any consumer of a Web service is a service requestor [8].

The service registry is responsible for advertising Web service descriptions published to it by service providers and for allowing service requestors to search the collection of service descriptions contained within the service registry. The service registry role is to be a match-maker between service requestor and service provider [8].

Besides the roles just described, three operations are defined as part of SOA architecture: publish, find and bind. These operations define the contracts between the SOA roles.

The publish operation is an act of service registration or service advertisement. When a service provider publishes its Web service description to a service registry, it is advertising the details of that Web service description to a community of service requestors.

The find operation is the logical dual of the publish operation. It is the contract between a service requestor and a service registry. With the find operation, the service requestor states a search criteria, such as type of service. The service registry matches the find criteria against its collection of published Web services descriptions.

The bind operation embodies the client-server relationship between the service requestor and the provider [8]. It can be sophisticated and dynamic, such as on-the-fly generation of a client-side proxy based on the service description used to invoke the Web service, or it can be a static model, where a developer hand-codes the way a client application invokes a Web service [8].

Besides complying with the SOA pattern, the Web service technology can be factored into three protocols stacks [8]: the wire stack (or exchange format), the description stack and the publish and discovery stack. To follow there is a brief description of each stack. It is important to note that any given Web service does not require the presence of all those stacks in order to be considered a Web service.

- The Wire stack
  The wire stack represents the technologies that determine how a message is sent/received from the service requestor to the service provider. The stack is composed of three levels. The first level is a network protocol, which can be an Internet wire protocol, such as HTTP [6] or FTP [21], or sophisticated enterprise-level protocols such as RMI/IIOP [19]. The second level is the data encoding mechanism. Web services use XML for data encoding. The third level refers to XML messaging layers. For XML messaging, Web services use SOAP [37], which acts as a wrapper to XML messages, in order to guarantee a solid, standard-based foundation for Web services communication.
- The Description Stack
  The key element of SOA is the service description. The service description is published by the provider through the publish operation and it is retrieved by the requestor as a result of the find operation. The service description informs the requestor everything it needs to know in order to invoke the Web service. The service description also indicates what information (if any) is returned to the requestor as a result of the Web service invocation.

  The main goal on service description is to provide information about a service that are important to the service requestor. In Web services, XML is the basis of service description. The XML Schema specification (XSD) [34] defines the canonical type system. Besides this level, the next levels of the stack are the descriptions of the service interface, the service concrete mapping and the service endpoint. All of those levels use WSDL [31]. With WSDL, a developer describes the set of operations supported by a Web service, including the kinds of objects that are expected as input and output of such operations, the various bindings to concrete network and data encoding schemes. An endpoint defines the network address where the service itself can be invoked.

  Due to being an XML language, WSDL is a very flexible model for services descriptions but it is also rather verbose. For most applications the XML verbosity is not a problem. Sensor networks applications, however, are different. A typical sensor device has very limited processing power and memory capacities, and, most importantly, has a very slow communications channel available. Therefore, a more compact mechanism for data representation is needed. One example of such a mechanism is the WAP Binary XML Content Format (WBXML [32]). This format defines a compact binary representation for XML [33], intended to reduce the size of XML documents for transmission and to simplify parsing them.

- The publish and discovery stack
  This stack corresponds to the directory service for Web services. Service providers need a publication mechanism so that they can provide information about the Web services they offer and service requestors need well-defined find APIs for using such Web services. The UDDI standard [28] is the proposed technology for Web services directory.

3. Related work

Several works on data centric communication are based on localized algorithms in order to reduce redundancy, saving energy. Localized algorithms are distributed algorithms that achieve a global goal by communicating with nodes in some neighborhood only [23]. Directed diffusion [13] and SPIN [16] are two representative localized algorithms specifically designed for sensor networks. Both are data-centric protocols and assume a strong coupling between the components of data dissemination and application-specific features.

We propose a generic architecture for sensor networks based on well-known standards for data description. By using an underlying data dissemination protocol, such as SPIN or directed diffusion, our architecture provides a flexible and application-independent solution for sensor network design. We promote a clear distinction between application and data communication functionalities, while still enabling the application to dynamically change the underlying network behavior. The interaction between the application and the communication protocol will be achieved through well defined interfaces.

Recent works address naming and service discovery for heterogeneous networks of devices. Most of these works rely on IP-based communication, and do not consider dynamic and resource constraint environments such as sensor networks. Universal Plug-and-Play [30] uses a subset of XML to describe resources provided by devices. It is limited to TCP/IP networks. Service Location Protocol (SLP) [20] facilitates the discovery and use of heterogeneous network resources using centralized Directory Agents. The Berkeley Service Discovery Service (SDS) [5] extends this concept with secure, authenticate communications and a fixed hierarchical structure for wide area operation. Centralized repositories and fixed hierarchies do not fit well to sensor networks. Our proposal is totally distributed and based on lightweight protocols.

The Intentional Naming System is an attribute-based name system which operates in a overlay network over the Internet [1]. It provides a method based on late binding to cope with dynamically located devices. Despite of having several features desirable for sensor networks, INS was designed for more generic mobile networks, offering a sophisticated hierarchical attribute matching procedure.

Our proposal has similarities with [7,17,38], which are database approaches for sensor networks. In [38] the sensor computation capabilities are exploited to execute part of the query processing inside the network, using query proxies. In [7] a SQL-like declarative language is proposed for users who submit queries to a sensor networks. In [17], a sensor network architecture based on the concepts of virtual databases and data-centric routing is proposed. The main difference between such works and ours is that we propose a totally distributed service approach, based on ubiquitous standards. Exposing sensor functionalities as services offers a more flexible architecture when comparing to SQL queries. Besides that, our proposal addresses interoperability among different systems, which is not easily achived through a database approach.

4. System architecture

Our work proposes a generic and flexible architecture for sensor networks based on the Web services technology. Web services are built according to the service-oriented architecture (SOA pattern) and they can be described by a trio of interoperability stacks [8]. Sections 4.1 and 4.2 describe the physical components of the proposed system and the roles played for those components according to the SOA pattern. Section 4.3 describes the system elements according to the Web services interoperability stacks.

4.1. System physical components

In a generic sensor network, the component nodes can have different functionalities, which are:

- Specialized sensor devices of different types (seismic, temperature, light, motion): detect and collect specific environmental data.

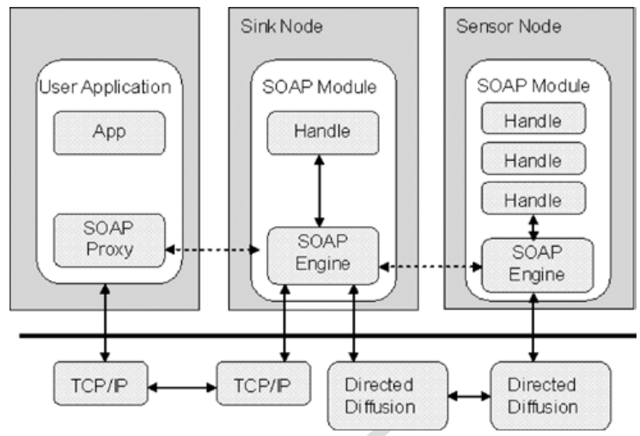

Figure 1. System architecture.

- Routing nodes: receive and transmit data from/to neighbor nodes.
- Aggregator nodes: gather received data before transmission, in order to save transmission energy. The aggregation is accomplished through specific filters for each sensor type and for each application.
- Sink nodes: receive queries from applications and extract information from the sensor network to meet the queries. In general, they are nodes with larger processing power and storage capacity.

In our system, the two main physical components are the sensor nodes and the sink nodes (figure 1). A sensor node contains one or more specialized sensing devices. In addition to it, it has routing and aggregation capabilities. Thus, the routing function is distributed among all nodes. The work also assumes that all sensor nodes have enough processing and storage capacities to store and execute aggregation filters.

Sink nodes provide application interfaces through which external systems can obtain sensor network collected information. Such interfaces can be accessed both locally or remotely. Sink nodes can also aggregate data, but they do not have sensor devices. We assume that they are more powerful, regarding processing and communication capabilities, than sensor nodes.

4.2. System architecture according to SOA

The proposed system architecture is based on the concept of service-oriented architecture [8]. A user application querying data from a sensor network plays the role of a service requestor. Sink nodes act primarily as service providers to the external environment. They provide the service descriptions of the whole sensor network, and they offer access to such services. At the same time, sink node act as requestors to sensor nodes, that request their specialized services, in order to meet the user application needs. Sensor nodes are service providers which provide data and filters. Sensor nodes send their service descriptions to sink nodes, thus they execute the basic

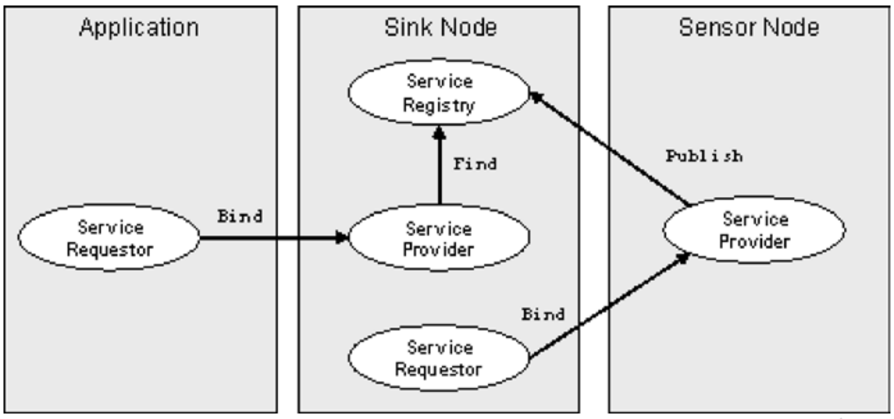

Figure 2. SOA roles in the proposed architecture.

publish operation. Sink nodes also act as registries, that keep a repository with service descriptions of each sensor type that exists in the sensor network (figure 2).

In our system, the functionality of the publish operation is accomplished through the `Publish_content` operation, and the functionalities of find and bind operations are both accomplished through the `Subscribe_interest` operation (see Section 4.3.2). Such operation is translated by the sink to a find operation followed by a bind with the sensor nodes that can meet the application request.

4.3. Interoperability stacks

In our system, the **wire stack** is composed of the SOAP protocol and an underlying data dissemination protocol, the directed diffusion [13] protocol. The **description stack** is based on WSDL documents. The functionalities of **publish and discovery stack** are accomplished by a software module that is executed in sink nodes. Sections 4.3.1 and 4.3.2 detail the wire and description stacks. We do not describe the discovery stack in detail since it is not relevant for our current work.

4.3.1. The wire stack: The communication framework

Users applications interested in submitting queries to the sensor network must access some sink node. The communication between user applications and sink nodes can be accomplished through conventional TCP/IP sockets. Applications must generate a SOAP message which describes the user interests. Such a message is generated based on network WSDL documents stored in the sink repository. Since WSDL is an open and ubiquitous standard for services description, there are many tools [12,26] for automatic generation of SOAP proxies in order to use WSDL descriptions in applications. Proxies build SOAP messages and receive query results, thus representing the interface between applications and sink nodes. The interaction between user applications and the system is one of the kind application-application, providing more flexibility than a direct user interface. Instead of submitting queries in a predefined format, which is specified through the user interface, applications have freedom for choosing the way they want the data is collected and delivered.

All communication inside the sensor network is accomplished by using direct diffusion and formatted as SOAP messages. The sending and receiving of SOAP messages by a SOAP node is mediated by a binding to an underlying protocol. SOAP messages can be transported by using a variety of underlying protocols. SOAP Version 1.2 Part 2: Adjuncts [35] includes the specification for a binding to HTTP. Additional bindings can be created by specifications that conform to the binding framework introduced in [36]. In our system we define a SOAP-Diffusion binding.

The SOAP protocol is responsible for defining exchanging rules and messages format in our system. In order to reduce the messages size, thus saving energy in sending/receiveing, the XML compact binary representation [32] is adoted for SOAP messages exchanged inside the sensor network

Both SOAP module and directed diffusion model must be present in every node in the network.

- SOAP Module

The SOAP module in our system is composed of three main components: the SOAP engine, a set of handles and a binding with the underlying protocol. The SOAP engine acts as the main entry point into the SOAP module. It is responsible for coordinating the way how SOAP messages flow through the handles and for ensuring that SOAP semantics is followed. Handles are the basic building blocks inside the SOAP module and they represent the messages processing logic. Three kinds of handles are defined: common handles, transport handle and Web services specific handle. Common handles are responsible for message marshalling/unmarshalling, header and attachment processing, serialization, data type conversion, and other basic functions. The transport handle `Matching_Data` was specifically built for the message sending and receiving through the directed diffusion protocol. The handle `Matching_Filter` represents the activation of application-specific filters inside the network. More details about the use of specific handles are described in Section 5.

Sink nodes contain common handles only. Sensor nodes contain, besides common handles, the transport handle

`Matching_Data` and the Web services specific handle `Matching-Filter`.

- Directed Diffusion Module

For communication among all sensor network components the directed diffusion protocol [13] is used. We suppose the existence of interest propagation through the network, gradient configuration in nodes and data that match interests being disseminated based on gradients. We also suppose the existence of filters which represent application-specific, in-network processing. With our architecture, we achieve a clear separation between communication and data processing functions. We modified the basic diffusion model in order to reflect such a separation.

The current directed diffusion model [13] consists of a core diffusion layer, a diffusion library and the application layer which includes applications and filters. The core diffusion layer is used to receive/send out packets from/into the network. The library provides an interface for the overlying application classes. Through this interface, applications publish data and subscribe interests [24].

Our system uses the core diffusion layer as its basic data dissemination protocol. Gradients configuration, matching data to interests and matching data to filters functionalities are part of the diffusion. They are kept although the representation model for data, interests and filters is changed. Attributes that describe data, interests and filters are represented through the WSDL language, and the matching functions are carried out by SOAP handles. Gradients and application-specific filters are implemented as software modules.

In spite of being based on the directed diffusion protocol, the proposed architecture relies on well defined and independent functional modules that communicate through well defined interfaces. Therefore, the system could be easily adapted to be used with other underlying data dissemination protocols.

4.3.2. The services description stack: WSDL documents

Generic services that are provided by a sensor network are described through a WSDL document. In that document, port type elements contain the both types of service descriptions: services provided by sensor nodes and services provided by sink nodes. Each service port type contains operations that can be thought as system APIs. Operations contain parameters that are defined in the document through messages and elements. Bindings of operation definitions to its concrete implementation are to be defined according to the underlying protocol. The WSDL language allows a binding to be defined through SOAP or directly to a lower level protocol. The operation implementation place is indicated by a port that can be identified by any unique identifier, such as a device address.

The operations defined for the Web services specified in our system are: (i) **Publish_Content:** used by the sensor node to create and disseminate a SOAP message which contains its service descriptions; (ii) **Publish_Data:** used by sensor nodes to create SOAP messages that communicate generated data; (iii) **Subscribe_Interest:** used by an application to submit a query to a sink node; (iv) **Subscribe_Filter:** used by an application in a sink node to inject a new filter in the network.

5. System description

Sensor networks have an initial setup stage, that comprises four different phases: deployment, activation, local organization and global organization [29]. Deployment can be as diverse as establishing one-to-one relationships by attaching sensor nodes to specific items to be monitored [2], covering an area with locomotive sensor nodes [11], or throwing nodes from an aircraft into an area of interest [27]. Due to their large number, nodes have to operate unattended after deployment. For energy saving, sensor nodes reside in a sleep state until the deployment. Therefore, sensors need to undergo an activation phase after they are scattered in the target area. The local organization phase includes the neighbors' discovery. During the global organization phase, nodes establish the communication path to some sink in the network. It is essential that all nodes reach a sink through some path so that their data can be delivered to the application. After the organization phase, each node might to be able to know and distinguish the nearby nodes. Any unique identifier can be used as a node identificator, as for example, its MAC address or a device serial number.

Our system operates according to three steps, described as follows. Step 1 is the network initial configuration and it occurs during the local and global organization phases that have already been described. Steps 2 and 3 are based on the working stages of directed diffusion protocol [13]. We discuss each one of those steps in the next sections and the figure 3 presents a sequence diagram describing the sytem operation according to such steps.

5.1. Step 1: Initial set up

In our system, during the local and global organization phases, nodes exchange SOAP configuration messages (figure 4), that describe the services (data and filters) supplied by them. Such messages include the node and network identification (the latter is used when there are several interconnected sensor networks), a TTL (sensor time-to-live), sensor type (s), geographical location, current amount of energy, maximum and minimum confidence degrees, maximum and minimum acquisition intervals (data rate), node existing filters and specific information of each sensor type. The SOAP configuration message is broadcasted into the network by using the diffusion core functionality. When a sensor node receives a configuration message, it can decide wheater to transmit it or not. If the message describes a sensor type that matches the sensor node own features or if a similar message has already been sent, the node does not need to transmit it again.

Sink nodes store the content of received configuration messages in a soft-state based local repository. It is

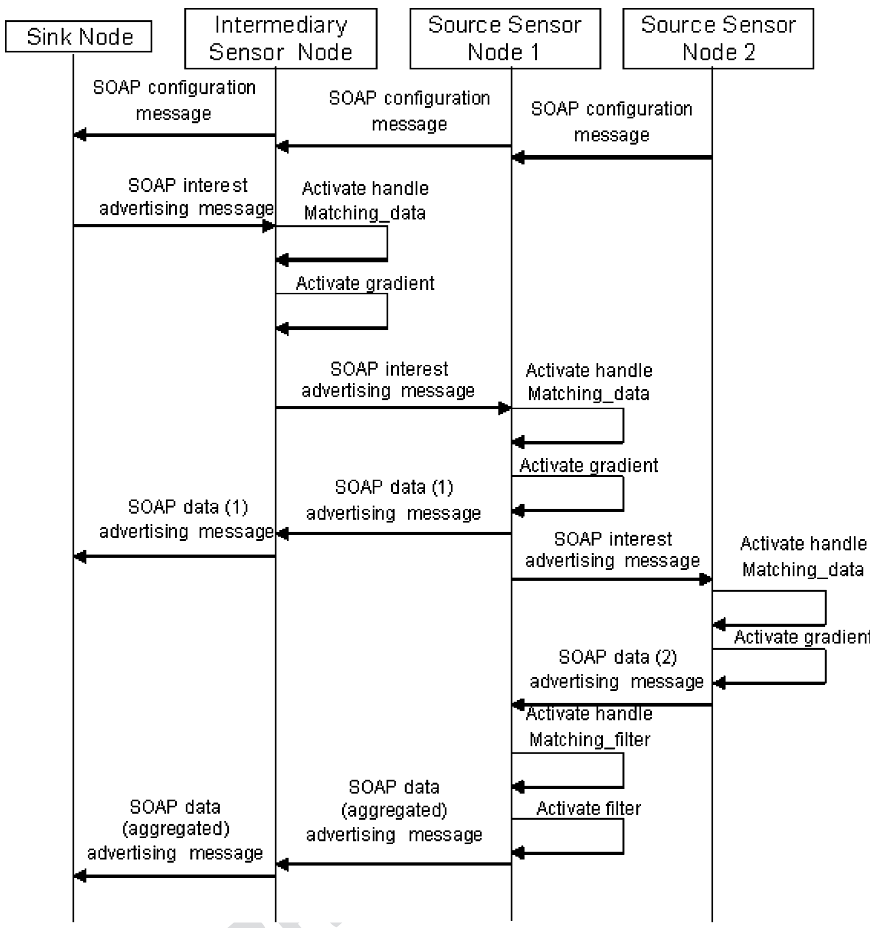

Figure 3. Sequence diagram describing the system operation.

important that every sink in the network has the complete knowledge on all existent sensor types. Sinks exchange messages periodically, so that all sinks contain the same information.

Since configuration messages traverse intermediaries nodes until reaching a sink, such nodes can also store messages exploiting their content, for example, by extracting geographic and energy information when disseminating interests through the network. The information about sensor geographical location can be used when the underlying diffusion protocol implements some kind of location-based routing optimization [39]. The directed diffusion protocol can be further optimized considering the sensor current energy in the decisions about routing. The optimization procedures based on geography location or current energy are included as filters in the network, and are executed only when the application asked for it.

5.2. Step 2: Interest advertisement

Applications that request data from a sensor network should subscribe an interest in some sink. An interest contains the sensor type, the data type, the geographical location of interest, the acquisition interval (data rate) and the acquisition duration. For time critical applications, a threshold value can be included, as a limit from which sensors must inform data, regardless the current acquisition interval.

Applications can request the activation of application-specific filters, which are already deployed in nodes. Furthermore, new filters can also be on-the-fly injected as programs in the network. A filter contains an identifier and a list of data types with their respective values. The filter identifier is used to trigger the execution of an already existing program in the sensor node when such node receives data that match values which are specified in the filter. When injecting a new filter, the program filter itself is transported as an attachment in a SOAP message (SOAP attachment capacity [37]).

SOAP messages that advertise interests (figure 5) are disseminated in the sensor network by using the diffusion core and the diffusion gradient functionalities [24]. When a sensor node receives an interest message, the handle `Matching_Data` in the SOAP module verifies if the interest matches some sensor provided data. The handle extracts from the message all the parameters needed for configuring the gradients, according to the adopted diffusion model [13].

```xml
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:m0="empty">
    <SOAP-ENV:Body>
        <m:PublishContent xmlns:m="http://namespace.example.com">
            <parameter ID="NODE_MAC_ADDRESS" NetworkID="NODE_NETWORK_ID">
                <m0:TTL unit="Seconds">3600</m0:TTL>
                <m0:Type>Motion</m0:Type>
                <m0:DataDomain>
                    <m0:Value>Four Legged Animal</m0:Value>
                    <m0:Value>Two Legged Animal</m0:Value>
                    <m0:Value>Creeper Animal</m0:Value>
                </m0:DataDomain>
                <m0:GeographicLocation unit="LatLong">
                    <m0:x>35.00</m0:x> <m0:y>-23.00</m0:y>
                </m0:GeographicLocation>
                <m0:Energy unit="J">1</m0:Energy>
                <m0:Confidence>
                    <m0:Max>1.0</m0:Max> <m0:Min>0.2</m0:Min>
                </m0:Confidence>
                <m0:DataRate unit="mSeconds">
                    <m0:Max>10</m0:Max> <m0:Min>1000</m0:Min>
                </m0:DataRate>
            </parameter>
        </m:PublishContent>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


Figure 4. SOAP configuration message.

5.3. Step 3: Data advertisement

A sensor generates data in an initial rate which is specified in the configuration message. The sensor only sends SOAP data advertisement messages if there is some active gradient which represents an interest that matches its own data type. The sensor changes the acquisition interval according to the received SOAP interest message. When detecting data for which it has received an interest, the sensor will issue a data advertisement message, which is delivered to the underlying diffusion protocol.

SOAP messages advertising data (figure 6) contain the data type, the instance (or value) of that type that was detected, the sensor current location (sensors can be mobile), the signal intensity, the confidence degree in the accomplished measurement, a timestamp, and the current sensor amount of energy.

Message dissemination involves a matching stage between data and interests, and the possible execution of filters. The matching data to interest stage is accomplished by the handle `Matching_Data` (step 2). The handle `Matching_Filter` matches data to filters and dispatches the filters execution


```xml
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:m0="empty">
    <SOAP-ENV:Body>
        <m:SubscribeInterest xmlns:m="http://namespace.example.com">
            <parameter>
                <m0:SensorType>Motion</m0:SensorType>
                <m0:DataType>Four Legged Animal</m0:DataType>
                <m0:DataRate unit="mSeconds">20</m0:DataRate>
                <m0:Duration unit="Seconds">20</m0:Duration>
                <m0:Area>
                    <m0:PointA unit="LatLong">
                        <m0:x>35.00</m0:x> <m0:y>-23.00</m0:y>
                    </m0:PointA>
                    <m0:PointB unit="LatLong">
                        <m0:x>35.02</m0:x> <m0:y>-23.03</m0:y>
                    </m0:PointB>
                </m0:Area>
                <m0:Threshold>0</m0:Threshold>
            </parameter>
        </m:SubscribeInterest>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


Figure 5. SOAP interest advertisement message.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:m0="empty">
    <SOAP-ENV:Body>
        <mPublishData xmlns:m="http://namespace.example.com/">
            <parameter ID="NODE_MAC_ADDRESS">
                </m0:DataValue>Elephant</m0:DataValue>
                </m0:Location unit="LatLong">
                    <m0:x>35.00</m0:x> <m0:y>-23.00</m0:y>
                 </m0:Location>
                </m0:Intensity>0.6</m0:Intensity>
                </m0:Confidence>0.85</m0:Confidence>
                </m0:Energy>0.9</m0:Energy>
                </m0:TimeStamp>08:16:40</m0:TimeStamp>
            </parameter>
        </mPublishData>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


Figure 6. SOAP data advertisement message.

whenever it is necessary. The resulting (possibly aggregated or filtered) data is delivered to the diffusion layer as a new SOAP data advertisement message to be disseminated along the network.

6. Conclusions

In this paper, we presented a service based architecture for designing sensor networks. We state that the future wireless sensor networks should provide a ubiqutuos, standardized access through a common and application independent interface. The contributions of this work are three-fold. First, we propose generic architecture in which the data communication functionality is separated from the application-specific processing. Second, we have defined a Web services approach for wireless sensor networks, in which sink nodes are modeled as Web Services that expose the services provided by the network by using a standard service interface. Third, we propose the use of the WSDL language and SOAP protocol, already recognized as Internet standards, as the mechanisms for describing services and formatting messages which are used by the underlying communication protocol.

The proposed approach offers high expressiveness and flexibility when designing sensor networks, allowing the interoperability of heterogeneous sensor. In our approach, sensor networks can be used as a system for supplying data for different applications and users. Our main goal is providing the underpinning for building more general purpose networks, instead of strictly task-specific ones, in order to assist a large range of users, possibly spread all over the world, who share a common interest in a specific application area.

Since energy saving is a key element in WSN design, our proposal depply relies on keeping the energy spent amount on the same level as current WSN systems. In particular, communication energy cost is expected to be significantly higher than local computation cost [9]. Therefore, the additional processing needed for parsing SOAP messages should be insignificant to the system. For this reason, our approach addresses energy saving in data transmission by adopting a compact binary XML format in the messages exchanged inside the WSN.

References

[1] W. Adjie-Winoto, et al., The design and implementation of an intentional naming system, 17th ACM Symposium on Operating Systems Principles (SOSP '99). Published as Operating Systems Review 34(5) (1999) 186–201.

[2] M. Beigl, H. Gellersen and A. Schmidt, MediaCups: Experience with design and use of computer-augmented everyday objects, Computer Networks, Special Issue on Pervasive Computing 25(4) (2001) 401–409.

[3] A. Choksi, Hierarchical Routing in Sensor Network, CS-672: Seminar on Pervasive and Peer-To-Peer Computing, Storage & Networking. Term-Paper Submission, Rutgers University. Available in: http://www.cs.rutgers.edu/~achoksi/presentation/CS672_paper_ankur.pdf.

[4] F.P. Coyle, *XML, Web Services, and the Data Revolution* (Addison-Wesley Information Technology Series, Addison-Wesley Press, 2002).

[5] S. Czerwinski, et al., An architecture for a secure service discovery service, in: *Proc. ACM/IEEE MOBICO*M (Aug. 1999), pp. 24–35.

[6] R. Fielding, et al. RFC 2616. Hypertext Transfer Protocol–HTTP/1.1. (June, 1999), Available in: ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt.

[7] R. Govindan, et al., The Sensor Network as a Database (Sept. 2002) Available in: ftp://ftp.usc.edu/pub/csinfo/tech-reports/papers/02-771.pdf, Tech-Rep 02-771 CS Department, University of Southern California.

[8] S. Graham, et al., *Building Web Services with Java: Making Sense of XML* (SOAP, WSDL, and UDDI. Sams Publishing, 2002).

[9] J. Heidemann, et al., Building efficient wireless sensor networks with low-level naming, in: *Proc. Symposium on Operating Systems Principles*, Chateau Lake Louise, Banff, Alberta, Canada, ACM. (Oct., 2001) pp. 146–159. Available in: http://www.isi.edu/~johnh/PAPERS/Heidemann01c.html.

[10] J. Heidemann, et.al., Diffusion filters as a flexible architecture for event notification in wireless sensor networks—USC/ISI Technical Report 2002–556.

[11] A. Howard, M. Mataric and G. Sukhatme, Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem, in: *Proc. DARS 02*, Fukuoka, Japan (June 2002).

[12] IBM White Paper, Web Services Toolkit. (April 2002). Available in: http://www.alphaworks.ibm.com/tech/Webservicestoolkit.

[13] C. Intanagonwiwat, R. Govindan and D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in: *Proc. ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, Boston, MA, USA, (Aug 2000) pp. 56–67.

[14] Jini TM Available in: http://java.sun.com/products/jini/ (1998).

[15] B. Krishnamachari, D. Estrin and S. Wicker, Modelling Data-Centric Routing in Wireless Sensor Networks. Available in: http://www2.parc.com/spl/members/zhao/stanford-cs428/readings/ Networking/ Krishnamachari_infocom02.pdf.

[16] J. Kulik, R.B. Heinzelman and H. Balakrishnan, Negotiation-based protocols for disseminating information in wireless sensor networks (2000), ACM Wireless Networks. Available in: http://citeseer.nj.nec.com/ 335631.html, 2000.

[17] S. Madden, et al., TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks. Available in: http://www.cs.berkeley.edu /∼madden/ madden_tag.pdf.

[18] Microsoft Corporation and Digital Equipment Corporation, The Component Object Model Specification. Available in: http://www.opengroup.org/pubs/catalog/ax01.htm, Oct. 1995.

[19] OMG (Object Management Group), The Common Object Request Broker: Architecture and Specification. Revision 2.0 (July 1995).

[20] C. Perkins, Service Location Protocol White Paper (May 1997) Available in: http://playground.sun.com/srvloc/slp_white_paper.html.

[21] J. Postel and J. Reynolds, RFC 959.FILE TRANSFER PROTOCOL (FTP) (Oct. 1985), Available in: ftp://ftp.rfc-editor.org/in-notes/rfc959.txt.

[22] H. Qi, P.T. Kuruganti and Y. Xu, The development of localized algorithms in wireless sensor networks, Invited Paper—Sensors 2002, 2 (2002) 286–293.

[23] K. Römer, O. Kasten and F. Mattern, Middleware challenges for wireless sensor networks—ACM SIGMOBILE Mobile Computing and Communications Review 6(2) (2002).

[24] F. Silva, J. Heidemann and R. Govindan, Network Routing Application Programmer's Interface (API) and Walk Through 9.0 (2002). Available in: http://citeseer.nj.nec.com/silva02network.html.

[25] L. Subramanian, H. Katz, An Architecture for Building Self-configurable Systems. Avaliable in: http://www.cs.berkeley.edu/∼ lakme/sensor.pdf.

[26] SUN Microsystems, Implementing Services On Demand and the Sun Open Net Environment (Sun ONE) (2001). Available in: http://www.sun.com/software/sunone/wpimplement/wp-implement.pdf.

[27] The 29 Palms Experiment: Tracking Vehicles with a UAV-Delivered Sensor Network. Available in: http://www.eecs.berkeley.edu/_pister/ 29Palms0103.

[28] UDDI.org, UDDI Technical White Paper (Sept. 2000). Available in: http://www.uddi.org/pubs/ Iru_UDDI_Technical_White_Paper.PDF.

[29] C. Ulmer, L. Alkalai and S. Yalamanchili, Wireless distributed sensor networks for in-situ exploration of mars, Work in progress for NASA Technical Report. Available in: http://users.ece.gatech.edu/ ∼grimace/research/reports/nasa_wsn_report.pdf.

[30] Universal Plug and Play: Background (1999). Available in: http://www.upnp.com/ resources/UPnPbkground.htm

[31] W3C (World Wide Web Consortium) Note, "Web Services Description Language (WSDL) 1.1. (March 2001). Available in: http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[32] W3C (World Wide Web Consortium) Note, WAP Binary XML Content Format (June 1999). Available in: http://www.w3.org/TR/wbxml/.

[33] W3C (World Wide Web Consortium) Recommendation, Extensible Markup Language (XML) 1.0 (Second Edition) (Oct. 2000). Available in: http://www.w3.org/TR/REC-xml.

[34] W3C (World Wide Web Consortium) Recommendation, XML Schema Part 0: Primer (May 2001). Available in: http://www.w3.org/TR/xmlschema-0/

[35] W3C (World Wide Web Consortium) Working Draft: SOAP Version 1.2 Part 2: Adjuncts (26 June 2002). Available in: http://www.w3.org/TR/soap12-part2/.

[36] W3C (World Wide Web Consortium) Working Draft: SOAP Version 1.2 Part 1: Messaging Framework (June 2002). Available in: http://www.w3.org/TR/2002/WD-soap12-part1-20020626/.

[37] W3C (World Wide Web Consortium) Note on Simple Object Access Protocol (SOAP) 1.1 (May 2000). Available in: http://www.w3.org/TR/SOAP/.

[38] Yong Yao and J.E. Gehrke, The Cougar Approach to In-Network Query Processing in Sensor Networks. Sigmod Record, 31(3) (Sept. 2002). Available in: http://www.cs.cornell.edu/ johannes/papers/2002/sigmod-record2002.pdf.

[39] Y. Yu, R. Govindan and D. Estrin, Geographical and Energy Aware Routing: A recursive data dissemination protocol for wireless sensor networks. Available in: http://citeseer.nj.nec.com/461988.html.

Flávia Delicato received a M.Sc. degree on computer science in 2000 from the Federal University of Rio de Janeiro, Brazil. Currently she is a doctoral candidate on computer science at the same University. Her research interests include wireless networks, wireless sensor networks, web services and communication protocols.
E-mail: fdelicato@nce.ufrj.br

Paulo F. Pires received a M.Sc. Degree on computer science in 1997 and a D.SC. degree on computer science in 2002 both from the Federal University of Rio de Janeiro, Brazil. During the year 2000 he worked as visiting researcher at the CLIP lab in University of Maryland (USA). Currently he is a collaborator professor at the Department of Computer Science of Federal University of Rio de Janeiro. His current research interests include: transaction models for the Web environment, Web service composition, system integration architectures, and formal modeling tools for distributed systems. Dr. Pires is a member of ACM.
E-mail: paulopires@nce.ufrj.br

Luci Pirmez received a B.Sc degree on computer science in 1981, a M.Sc. degree on computer science in 1986 and the D.SC degree on computer science in 1996 from the Federal University of Rio de Janeiro, Brazil. She is a member of research staff of the Computer Center of Federal University of Rio de Janeiro. Her research interests include wireless networks, wireless sensor networks, networks management and security and formal description techniques for communication protocols.
E-mail: luci@nce.ufrj.br

Luiz Fernando Rust da Costa Carmo received a B.S. degree on electronic engineering in 1984, and a M.Sc. degree on computer science in 1988, both from the Federal University of Rio de Janeiro, Brazil, and the Ph.D. degree on computer science in 1994, from the Laboratory for Analysis and Architecture of Systems French National Organization for Scientific Research (LAAS/CNRS), Toulouse, France. He is a member of research staff of the computer center of Federal University of Rio de Janeiro. His research interests include formal description techniques for communication protocols and communication protocols for distributed systems. Presently, he is spending a sabbatical period at the Research Center of the United Technologies company.
E-mail: rust@nce.ufrj.br