A Flexible Middleware System for Wireless Sensor Networks

Flávia Coimbra Delicato[1], Paulo F. Pires[1,2], Luci Pirmez[1], Luiz Fernando Rust da Costa Carmo[1]
Núcleo de Computação Eletrônica – NCE[1] & Computer Science Department – DCC[2]
Federal University of Rio de Janeiro
P.O Box 2324, Rio de Janeiro, RJ, 20001-970, Brazil
{ fdelicato, paulopires, luci, rust } @ nce.ufrj.br

Abstract: The current wireless sensor networks (WSN) are assumed to be designed for specific applications, having data communication protocols strongly coupled to applications. The future WSNs are envisioned as comprising of heterogeneous devices assisting to a large range of applications. To achieve this goal, a flexible middleware layer is needed, separating application specific features from the data communication protocol, while allowing applications to influence the WSN behavior for energy efficiency. We propose a service-based middleware system for WSNs. In our proposal, sensor nodes are service providers and applications are clients of such services. Our main goal is to enable an interoperability layer among applications and sensor networks, among different sensors in a WSN and eventually among different WSN spread all over the world.

KEYWORDS: Wireless Sensor Networks, SOAP, WSDL, Web Services.

1. Introduction

Recent advances in micro-electro-mechanical systems (MEMS) technology, wireless communications, and digital electronics have enabled the development of low-cost, low-power, multifunctional sensor nodes that are small in size and communicate over short distances. These tiny sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks based on collaborative effort of a large number of nodes [2].

A wireless sensor network (WSN) is composed of a large number of such sensor nodes, which are densely deployed either inside the monitored phenomenon or very close to it and are interconnected by a wireless network. Sensor networks can play the role of a highly parallel, accurate and reliable data acquisition system.

Typically, sensors are devices with limited energy and processing capabilities, deployed in an ad-hoc fashion and communicating through low bandwidth wireless links. Sensor nodes have to operate unattended, since it is unlikely to service a large number of nodes in remote, possibly inaccessible locations. Therefore, energy saving is a crucial requirement in such an environment.

Examples of sensor networks include military networks for intruder detection, networks for environment monitoring, parking lot networks, surveillance networks and so on.

Sensor tasks usually have high-level descriptions, such as “report the detection of any 10 tons four-legged animal in region X”. However, individual sensor nodes typically provide very simple and low level functionality. Therefore, to meet a complex sensing task, sensor nodes must coordinate among themselves and the individually collected data must be aggregated to provide more accurate and significant results. The coordination among sensor nodes must take into account their heterogeneity and their individual features such as location, sensor type and residual energy.

Sensor 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 to appear significant redundancy among data generated from different sensors. Such a redundancy can be exploited to save transmission energy, through filtering and data aggregation procedures in-network. Also to save energy, the short-range hop-by-hop communication is preferred over the direct long-range communication to the final destination. Therefore, to achieve energy efficiency, applications should be able to dynamically change the network behavior, for example, influencing the way sensor data are routing throughout the network.

Current works [7,19,21,22] consider sensor networks as being designed for specific applications, where data communication protocols are strongly coupled to the application. In fact, the network requirements and organization, as well as the way data should be routed, change according to the application. In spite of the application specific behavior of the current sensor networks, many authors [27] 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 middleware service is needed to provide a layer of abstraction that separates application specific requirements from underlying data dissemination protocols.

A middleware for WSN should support the implementation and basic operation of a sensor network, such as described in [37]. This is a non-trivial task, since WSNs have some unique features, such as the resource constraint of nodes (energy, storage and processing) and the high dynamic and fault prone characteristics of the WSN environment. Furthermore, sensor nodes in the same network can be rather heterogeneous regarding their processing and storage capabilities. To deal with the intrinsic characteristics of sensor networks, some software design principles for WSN have been proposed in [11] and have been used by most of the WSN specific protocols. These principles are the adoption of localized algorithms, data-centric communication and the utilization of application-specific knowledge. A WSN middleware must take into account such design principles.

We propose a distributed middleware system for sensor networks sitting above the data dissemination protocol. Our approach is motivated by the fact that despite of the advantages of the middleware technology, current works on WSN do not consider such a technology in the design of WSNs. The proposed system addresses the specific requirements of WSN and it is based on the concept of services. Services are defined as the data provided by the sensor nodes and the applications (for instance, a filtering program) to be executed on such data. Clients access the sensor network submitting queries to those services.

Services are published and accessed through an XML-based language (Extensible Markup Language [41]) named WSDL language (Web Services Description Language) [39]. WSDL is used for describing services available on the Web, named *Web services*, in a standardized way.

Web Services build on SOAP [45] protocol's capability for distributed, decentralized network communication by adding new protocols and conventions that expose users functions to interested parties over a network from any service reachable from such a network. [8]. Any software component or application can be exposed as a Web service 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).

By adopting the Web Services paradigm, we propose an interoperability layer for sensor networks systems that is generic and flexible, providing the basic functionalities required for any WSN. Such a middleware layer is composed of the SOAP protocol and interfaces provided by WSDL documents. Using specific data dissemination protocols for sensor networks, such as direct diffusion [19] and LEACH [16], among others, and the service-based middleware layer, we intend to offer a flexible and powerful way of manipulating, extracting and exchanging data from sensor networks. Applications access the sensor network and modify the underlying data dissemination behavior through a common and application independent interface provided by the middleware layer.

The middleware interface provides a mechanism through which application specific code (such as programs to data filtering and data fusion) can be injected and triggered inside the network, allowing energy efficiency in data dissemination, thus increasing the WSN performance and time life. The middleware also enables the generation and communication of high level tasks, as well as the coordination of such tasks among nodes, even if the nodes have heterogeneous features. In order to suit to the WSN resource constraint and fault prone, the proposed middleware is designed to be robust and fault tolerant, demanding little processing and storage requirements, and keeping the messages exchanged as short as possible.

Our approach enables the construction of generic sensor networks capable of meeting the requirements of a large range of independently designed applications. Furthermore, the use of

standard protocols in the middleware layer provides the necessary mechanisms to enable the interoperability among different networks.

The present work describes the main features and the components of the proposed middleware service. The paper is organized as follows. Section 2 covers the background concepts. Section 3 presents the components of the proposed middleware system. Next, Section 4 details the system operation and Section 5 presents the related work. Section 6 discusses system features according to the specific requirements of sensor networks. Finally, Section 7 outlines the conclusions and future works.

2. Background

This section presents some background concepts needed for the comprehension of the remaining of the paper. The concepts outlined encompass WSN, middleware systems (generic and specific for WSN) and Web services technology.

2.1 Wireless Sensor Networks (WSN)

Wireless sensor networks represent an increasingly important example of distributed event systems [15]. 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 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 done by nodes performing measurements and relaying data through neighboring nodes to reach some sink in the network. Data sent by different nodes can be aggregated in order to reduce redundancy and minimize the traffic and thus the energy consumption. To enable data aggregation in network in an efficient way, application-specific code, such as data caching and collaborative signal processing should occurs as close as possible to where data is collected. Such a processing depends on attribute-identified data to trigger application-specific code and hop-by-hop processing of data [14].

WSN can be classified in proactive and reactive networks, according to the class of the target application. In proactive WSNs, nodes periodically (in a pre-defined interval) sense the environment and transmit data of interest. In reactive WSNs, nodes react immediately to sudden and drastic changes in the value of a sensed attribute. These classes of WSN are well suited for time critical applications.

Once the type of network is defined, protocols that efficiently route data from nodes to users have to be designed. Several WSN specific protocols have been proposed in the last few years [7,14,16,19]. Some protocols are sender-initiated [22] while others are receiver-initiated [19]. Some protocols are based on a flat network topology [19,22] while others are based on a hierarchical topology [7,16]. In the latter case, protocols adopt a cluster-based approach and make use of some algorithm for cluster formation [36] requiring the coordination among nodes in a cluster.

For large-scale networks, grouping nodes in clusters can be beneficial for a number of reasons [36]. From a routing perspective, clustering allows network protocols to operate in a hierarchical fashion, breaking transmissions into different levels. Such an approach is highly fault-tolerant, providing better isolation and recovery of network problems. Clustering can also be beneficial for data collection algorithms. Some applications do not require the data collection from all nodes during all time. Cluster members can collaborate about recent data measurements and determine how much information should be transmitted to the user application. By averaging data values collected within the cluster, the algorithm can trade data resolution for transmission power [36]. Finally, clustering can help dealing with non-ideal distribution of sensor networks. In areas where there are a redundant number of sensors, a clustering algorithm can be used to select which nodes better represent data

samples for the region and which ones can be put in a power-save mode, thus saving energy and increasing the lifetime of the network as a whole.

Most of WSN protocols rely on localized algorithms and data-centric communication, besides to exploit application-specific knowledge in the data dissemination. Localized algorithms are a special kind of distributed algorithms that achieve a global goal by communicating with nodes in a restricted neighborhood. Such algorithms scale well with increasing network size and are robust to network partitions and node failures [28]. Data-centric communication introduces a new style of addressing in which nodes are addressed by the attributes of data they generate (sensor type) and by their geographical location, instead of by their network topological location. Finally, the use of application knowledge in nodes can significantly improve the resource and energy efficiency, for example by application-specific data caching and aggregation in intermediate nodes [28].

Regardless the specific protocol adopted, all protocols depend on some mechanism for representation of user application queries and of generated sensor data, and for execution of application-specific processing triggered by pre-defined data attributes. Data-centric protocols represent queries and data through high level descriptions (meta-data) and disseminate such descriptions in the network instead of the collected raw data. When a cluster-based approach is adopted, a further mechanism for representation of coordination messages exchanged among nodes is needed.

2.2 Middleware Technology

Middleware technologies free application designers of explicitly dealing with problems related to distribution, such as heterogeneity, scalability, resource sharing, and the like. Middleware provides application designers with a higher level of abstraction, hiding the complexity introduced by distribution. In other words, distribution becomes transparent [34].

The term middleware is widely used to denote a layer comprised of groups of generic services sitting below user applications. Typical middleware services include directory services, service discovery, transactions, persistence and provide different types of transparencies, such as location transparency and fault transparency. CORBA [24], J2EE and J2ME [31], COM [23] and WAE [46] (Wireless Application Environment) are examples of traditional middleware technologies. The use of middleware systems speeds up the development and deployment of new applications, leaving to the developers only the task of designing business specific components.

Traditional middleware technologies have been developed assuming the requirements of fixed distributed systems. Such systems are composed of fixed devices, with high processing and storage capabilities, usually permanently connected to the network through continuous and high bandwidth connections. These distributed systems operate in a relatively static execution context. For static context we mean the bandwidth is high and continuous and the location of the devices and services hardly ever changes.

WSN are a category of ad-hoc networks having all the features of such networks and some further constraints. Devices in WSN have low processing and storage capabilities, can be mobile or not, can be destroyed or suffer battery depletion and are subject to environmental dynamics. Furthermore, they are typically connected through wireless links with low capability and error prone. The adopted communication paradigm is typically asynchronous and event-driven.

The essential requirements for WSN middleware include providing mechanisms that assure the efficient use of communication resources available and that allow the dynamic configuration of user applications. Besides, it must be robust, fault tolerant, lightweight and with short storage requirements, given the WSN low capabilities.

One additional requirement concerns the execution context information. Middleware collects information on the execution context, such as actual location of a device, value of network bandwidth, latency, available remote services, etc. Most of middleware developed for traditional distributed systems adopts the principle of transparency. By transparency, we mean that such a

context information is used privately by middleware and not shown to the applications. For example, middleware may discover a congestion in a portion of the distributed system and therefore redirect requests to access data to a replica residing on another part of the distributed system, without informing the application about this decision [5]. In the other hand, in WSN, applications must be aware of context information, in order to accomplish some strategy for efficient use of the scarce network resource. Such a feature is named principle of awareness. By awareness we mean that information about the execution context (or part of it) is passed up to the running applications, that are now in charge of taking strategic decisions [5].

The next section gives a more detailed view of WSN middleware characteristics.

2.3 WSN Middleware Requirements

The main purpose of middleware for sensor networks is to support the development, maintenance, deployment and execution of sensing-based applications. This includes mechanisms for formulating complex high-level sensing tasks, communicating those tasks to the WSN, coordination of sensor nodes to split the tasks and distribute them to the individual sensor nodes, data fusion for merging sensor readings of individual sensor nodes into a high-level result, and reporting the result back to the task issuer. Moreover, appropriate abstractions and mechanisms for dealing with the heterogeneity of sensor nodes should be provided [5]. All mechanisms provided by a middleware system should respect the special characteristics of WSN, mainly the energy efficiency, robustness, and scalability. The communication style to be adopted should typically be asynchronous, event-driven and data-centric.

Another unique feature of WSN middleware is the application knowledge in sensor nodes. Traditional middleware is designed to accommodate a wide variety of applications without necessarily needing application knowledge. Middleware for WSN, however, has to provide mechanisms for injecting application knowledge into the WSN [28].

A further characteristic addresses the concepts of time and location of sensed events. Since WSNs monitors real world data, time and spatial information are relevant, being key elements for fusing individual sensor readings. Therefore, support for time and location management should be tighly integrated into a middleware for WSN [5].

Finally, it is important to note that the scope of middleware for WSN is not restricted to the sensor network alone, but also covers external networks connected to the WSN (such as Internet) as well as the applications interested in querying sensor data through such external network.

Despite of the advantages of the middleware technology, current works on WSN are not considering such a technology in the network design. WSNs have been built with a high degree of dependency between the applications and the underlying communication protocol. Such a dependency generates rigid systems, with sensor networks being specifically designed to particular applications.

In fact, WSN applications should be able to access the network and modify the underlying data dissemination behavior in order to achieve energy efficient. The adoption of a middleware service provides a flexible, application independent layer that allows the interaction among different applications and the WSN, separating the data communication functionalities from the application specific processing.

In this work, we propose a middleware layer for sensor networks that aims to meet their specific requirements. Our proposal is based on the concept of service, and on the Web services technology. The next section gives an overview on the Web services technology.

2.4 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 component-based middleware systems, Web services expose an interface that can be reused without worrying

about how the service is implemented. Unlike current component-based middleware [23,24,31], 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[12]) and XML [41], which are vendor independent.

The *Web Services Description Language* (WSDL) [39] 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 [39]. 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.

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 [45]. 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's standard communication model is the asynchronous model, however, it can be mapped to represent more complex communication models such as RPC-like (solicit and response) or broadcast communication models. The RPC communication model is the last part of the SOAP specification.

Since the Web services technology uses XML as the encoding system, data is easily exchanged between computing systems with incompatible architectures and incompatible data formats. WSDL completely describes the Web service interface, while SOAP completely describes parameters, data types and exceptions included in a message being exchanged between Web services.

The Web services technology is based on a flexible architecture named SOA (service-oriented architecture [13]). In a service-oriented architecture three roles are defined: 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 [13].

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 [13].

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 [13]. 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 [13].

Besides to comply to the SOA pattern, the Web service technology can be factored into three protocols stacks [13]: the wire stack (or exchange format), the description stack and the publish and discovery stack. Next, we present a brief description of each stack.

- 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. Such a stack is composed of three levels. The first level is the network protocol. Web services can be based on a variety of standard, Internet wire protocols, such as HTTP [12] or FTP [26], as well as sophisticated enterprise-level protocols such as RMI/IIOP [24]. 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, that acts as a wrapper to XML messages, guaranteeing a solid, standard-based foundation for Web services communication.

- The Description Stack

The main goal on service description is to provide aspects of a service that are important to the service requestor. In Web services, XML is the basis of service description. The XML Schema specification (XSD) [42] defines the canonical type system and all service description technologies in the description stack are expressed using XML. Besides the level of canonical data type definition, 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. With WSDL, a developer describes the set of operations supported by a Web service, including the 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.

As being an XML language, WSDL is a very flexible model for services descriptions but it is also rather verbose. For most applications the verbosity of XML 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 representing the data is needed. One example of such a mechanism is the WAP Binary XML Content Format (WBXML [40]). This format defines a compact binary representation for XML [41], intended to reduce the size of XML documents for transmission and to simplify parsing them. WBXML was designed to be used as part of the WAP protocol [47]. The binary XML content format was designed to allow more effective use of XML data on narrowband communication channels with no loss of functionality or semantic information.

- 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, while service requestors need well-defined find APIs for using such Web services. The UDDI standard [35] is the proposed technology for Web services directory.

3. Proposed Middleware Service

Our work proposes a distributed middleware system for sensor networks sitting above the data dissemination protocol and basing on the Web services technology. Such a middleware aims to

provide a generic and flexible interoperability layer allowing different user applications to access and extract data from sensor networks.

The main goal of our middleware is to provide an interoperability layer:

- among user applications and the WSN, allowing the execution of data queries and of application specific processing in-network;
- among different sensors in the same WSN, allowing data communication and sensors coordination according to an underlying protocol;
- eventually, among different sensor networks.

The proposed system is based on the Web services technology. Web services are built according to a pattern called service-oriented architecture (SOA) and they can be described by a trio of interoperability stacks [13] (see Section 2.4).

Sections 3.1 describes the sensor network physical components considered in the proposed system. Section 3.2 describes the roles played by the middleware components in agreement with the SOA pattern, while Section 3.3 describes such components according to the Web services interoperability stacks.

3.1 Sensor Network Physical Components

In our system, we consider a sensor network as comprising of two main physical components: sensor nodes and sink nodes. Our distributed middleware runs in both sensor and sink nodes above the data dissemination and the location services. Furthermore, a proxy provides the communication interoperability between user applications and the sensor network (Figure 1). Its important to note that this proxy is not coupled to our middleware design, neither it is required to be built with any specific technology. It is actually a generic proxy responsible for generating SOAP messages to be exchanged between the user application and the WSN.

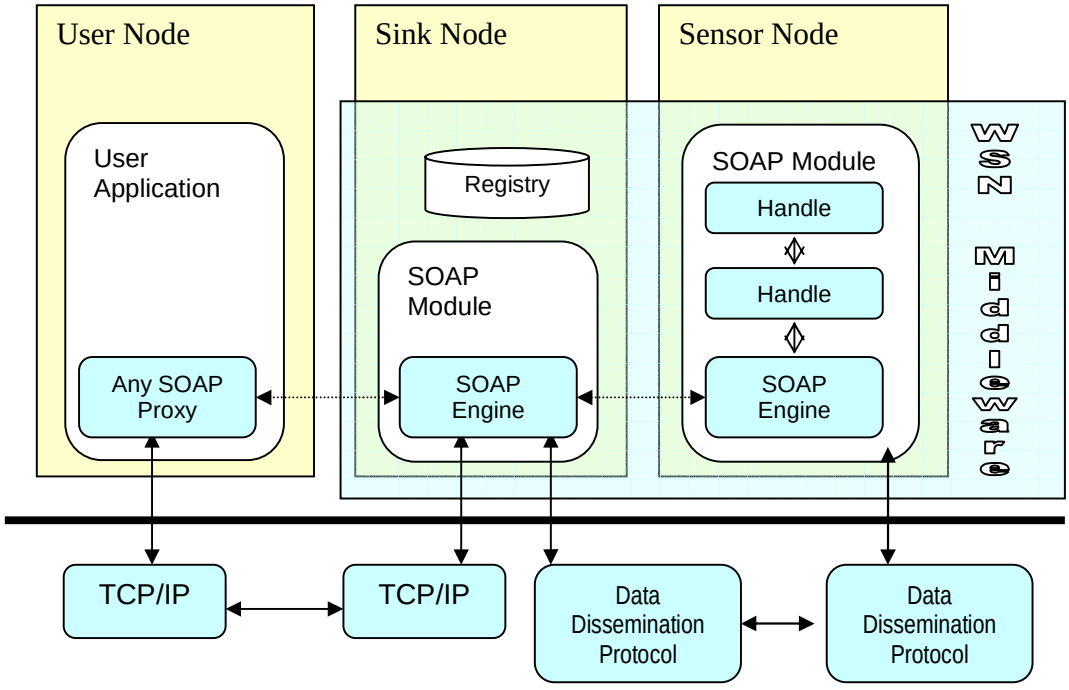

Figure 1 – System Architecture.

A **sensor node** can contain one or more specialized sensing devices. Furthermore, it can have routing and data aggregation capabilities. Thus, the routing function is distributed among all nodes. We assume that all the sensor nodes have enough processing and storage capacities to store and execute aggregation programs.

Sink nodes provide an interface through which external systems can obtain the information collected by the sensor network. Such interfaces can be accessed locally or remotely (i.e., through the Internet). Sink nodes can also aggregate data, but they do not have sensor devices. We assume that they are more powerful regarding to processing and communication capabilities than sensor nodes.

3.2 System Components According to the Service-Oriented Architecture Pattern

The proposed system is based on the concept of service-oriented architecture (SOA) [13] (see Section 2.4). 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 the sensor nodes, requesting their specialized services, in order to meet the user application needs. Sensor nodes are **service providers**, providing data and programs (for application-specific processing). Sensor nodes send their services description to sink nodes, thus executing the basic publish operation. Sink nodes also act as **registries**, keeping a repository with services descriptions of each sensor type existing 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 3.3.2).

Our system groups the functionalities described by the operations find and bind in one single operation. Sink nodes provide the services description interface and, at the same time, provide access to such services. The user application interacts only with sink nodes, and sink nodes in its turn access sensor nodes services passing the resulting data to the application. In fact, the operation find is only accomplished internally by the sink nodes, which consult their repositories of services descriptions. When an application submits a query to the sensor network, it is actually executing a bind to the services supplied by the sensor nodes. However, the application only interacts with the sink. The operation `Subscribe_Interest` is translated by the sink to a **find** operation followed by a **bind** to the sensor nodes that can meet the application request.

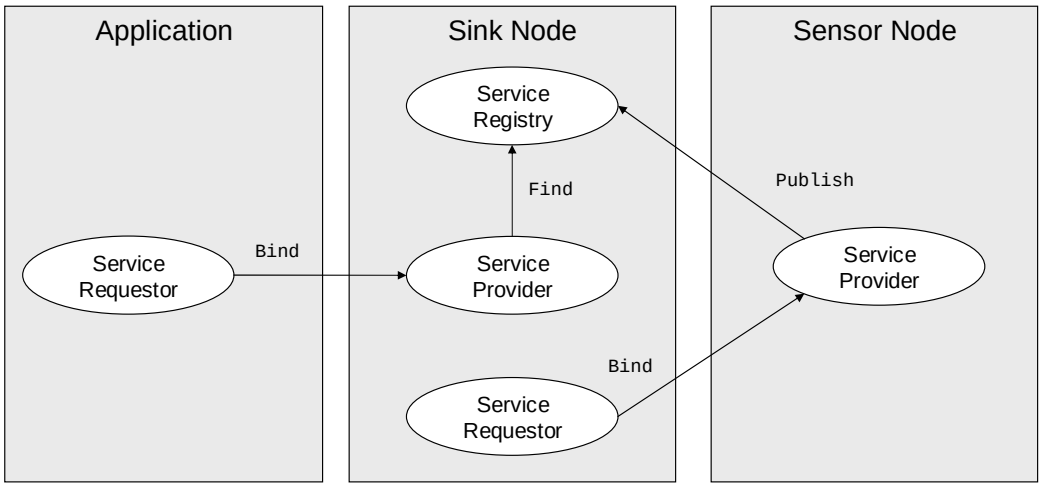

Figure 2 – SOA roles of the componentes of the proposed WSN middleware.

3.3 Interoperability Stacks

In our system, the **wire stack** is composed of the SOAP protocol and an underlying data dissemination protocol. We do not make assumptions about the underlying protocol. Instead, we provide a generic interface for a class of protocols. The **description stack** has all of its levels based on WSDL documents, in the document-centric approach [13]. The functionalities of the **publish and discovery stack** are accomplished by a software module executing in sink nodes. Sink nodes act as service registry agents. During the network configuration, sensor nodes send messages publishing their services and sink nodes keep a repository with such descriptions. Besides such functionalities, sink nodes act as interceptors for network services requests. External applications access the network via sink nodes. Sinks receive requests and direct such requests to sensor nodes according to the information stored in the sink repository.

Sections 3.3.1 and 3.3.2 detail the wire and description stacks. We do not describe the discovery stack in detail since it is not relevant to this work.

3.3.1 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 describing the user interests. Such a message is generated based on the sensor network service descriptions stored in the sink repository. Services descriptions are written in WSDL language. Since WSDL is an open and ubiquitous standard for services description, there are many tools [18,32] for automatic generation of SOAP proxies. Proxies build SOAP messages and receive back query results thus, they represent the software interface among applications and sink nodes. The proposed WSN middleware provides a service interface allowing user applications to interact with the sensor network system in an application-to-application communication style, offering more flexibility than a direct user interface. Instead of submitting queries in a proprietary and pre-defined format, specified through the user interface, applications are free to choose the way they want to view and receive data.

All the communication inside the sensor network is accomplished using the underlying data dissemination protocol 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 using a variety of underlying protocols. SOAP Version 1.2 Part 2: Adjuncts [43] includes the specification for a binding to HTTP. Additional bindings can be created by specifications that conform to the binding framework introduced in [44]. Specific bindings for each data dissemination protocol should be defined as needed.

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 [40] is adoted for SOAP messages exchanged inside the sensor network

The SOAP module, as well as a module representing the data dissemination protocol 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 coordinate the SOAP message's flow through the various handles and for ensuring that the SOAP semantics are followed. Handles are the basic building blocks inside the SOAP module and they represent the messages processing logic, including the marshalling/unmarshalling of messages transport specific processing. Three kinds of handles are defined: common handles, transport handle and specific handle. Common handles are responsible for marshalling/unmarshalling of messages, header and attachments processing, serialization,

conversions of data type to the types supported by the local software, among any other basic functions. The transport handle `Matching_Data` is specifically built for sending and receiving messages through the underlying protocol. The handle `Matching_Filter` is a sensor's specific handle which is built for representing the activation of application-specific programs inside the network. More details about the use of specific handles are described in Section 4.

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`.

3.3.2 The Services Description Stack: WSDL Documents

The generic services provided by a sensor network are described through a WSDL document. In that document, port types elements (see Sectio 2.4) contain two types of service descriptions: descriptions of services provided by sensor nodes and descriptions of services provided by sink nodes. Each service port type contains operations, that can be thought as system APIs. Those operations contain parameters, defined in the document through messages. Bindings of operation definitions to their concrete implementation should 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. A port identification, indicating the place containing the operation implementation, can be done through any unique identifier, as a device address.

The operations defined for the Web services specified in our system address the requirements of a generic sensor network. Despite of the data dissemination protocol adopted, a WSN needs mechanisms to: represent user queries and sensor data; represent and trigger application specific code; and to represent coordination messages in cluster-based approaches. The following operations aims to provide such mechanisms.

`Publish_Content:` used by the sensor node to create and disseminate a SOAP message containing its service descriptions. Services include types of sensing data and filters existent in the sensor node.

`Publish_Data:` used by sensor nodes to create SOAP messages communicating generated data.

`Subscribe_Interest:` used by an application to submit a query to a sink node. The query includes the interest description and the filters to be activated.

`Subscribe_Filter:` used by an application in a sink node to inject a new filter in the network. A filter contains the attributes to be matched for its execution and the syntax to invoke the filter program.

`Join_Cluster:` used by sensor nodes to declare their intention to join in a cluster.

`Advertising_Leader:` used by the elected leader node to announce its identity to the others cluster members.

4. System Operation

Sensor networks have an initial setup stage comprising of four different phases: deployment, activation, local organization and global organization [**37**]. Deployment is the physical placement of sensors in the target area. In order to reduce energy consumption, sensor nodes reside in a sleep state until the deployment. Therefore, sensors need to undergo an activation phase after they are scattered in the region of interest. 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 is supposed to know and distinguish the nearby nodes. Any unique identifier can be used as a node identifier, as for example, its MAC address or a device serial number. When adopting a hierarchical, cluster-based protocol, besides the phases just described, the WSN initial organization includes a phase for clusters formation, in which nodes

group themselves in clusters with a chosen leader or cluster-head responsible for the management of the communication among cluster members.

Our middleware system operates according to four different steps: intial setup, interest advertisement, data advertisement, and (optionally) cluster formation. We discuss each one of those steps in the next sections. Figure 3 presents a sequence diagram describing the system operation according to such steps.

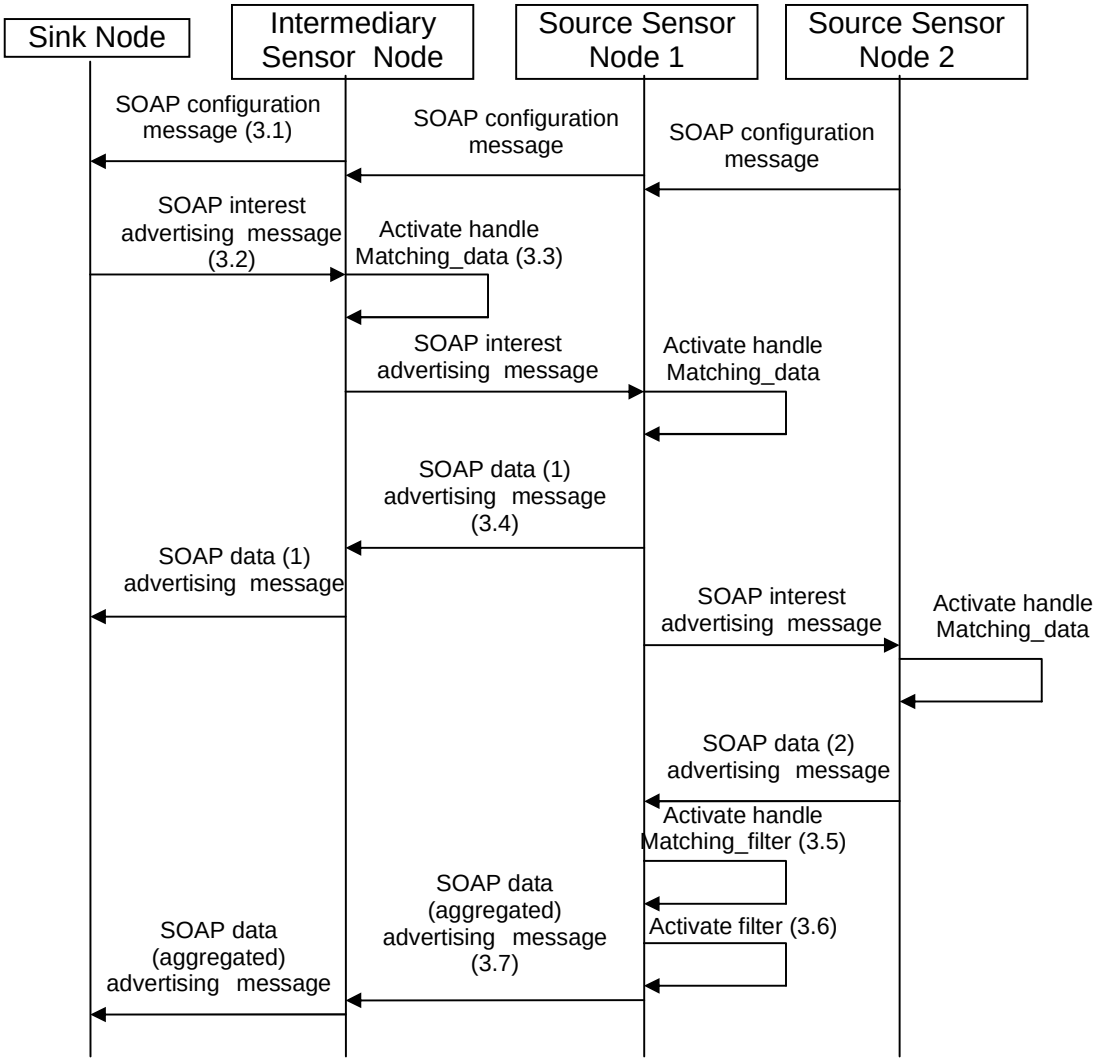

Figure 3 – Sequence diagram describing the system operation.

4.1 Step 1 - Initial Set Up

In our system, during the local and global organization phases, nodes exchange SOAP configuration messages (Figure 43.1), describing the services (data and filters) supplied by them. Such messages include the node and network identification (the latter 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), filters that exist in the node and specific information of each sensor type. The SOAP configuration message is broadcasted in the network using the functionality of the

underlying data dissemination protocol. When a sensor node receives a configuration message, it can decide to transmit it or not. If the message describes a sensor type matching its own features or if a similar message has already been sent before, the node does not need to transmit it again. Sinks keep entries for each different sensor type, therefore their repositories scale with the number of sensor types.

```
<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 message describing configuration messages.

Sink nodes store the content of received configuration messages in a local repository. Such a repository is based on soft-state, since active sensors in a particular instant of time can be inactive in a subsequent instant. It is important that every sink in the network has the complete knowledge on all existent sensor types. Sinks may periodically exchange messages, so that all sinks contain the same information.

Since configuration messages traverse intermediary nodes until reaching a sink, such nodes can also store messages exploiting their content, for example, extracting geographic and energy information when disseminating interests through the network. The information about sensor geographical location can be used when the underlying protocol implements some kind of location-based routing optimization [49]. The data dissemination protocol adopted 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 application-specific programs in the network, and are executed only when the application asked for it.

4.2 Step 2 - Interest Advertisement

Applications requesting 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 the sensors must inform data, regardless the current data rate.

Applications can request the activation of application-specific programs existent in nodes. Furthermore, new programs can be injected in the network. A program description contains an identifier and a list of data types with their respective values. The identifier is used to trigger the execution of the appropriate program already existent in the sensor node when such a node receives data matching the values specified in the program description. When injecting a new program, it is transported as a SOAP message attachment [45].

SOAP messages advertising interests (Figure 5 – SOAP message advertising interests. are disseminated in the sensor network using the underlying data dissemination protocol (Figure 3.2).

A handle responsible for matching data to interests, named `Matching_Data` handle (Figure 3.3) is provided as part of the middleware layer.

```
<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 message advertising interests.

4.3 Step 3 - Data Advertisement

A sensor generates data in an initial rate specified in its configuration message. The sensor only sends SOAP data advertisement messages if it had received a previous interest message advertising interests matching its own data type. Sensors change their acquisition interval according to the received SOAP interest messages. When detecting data for which they have received an interest, sensors issue data advertisement messages.

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.

The message dissemination (Figure 3.4) involves a matching stage among data and interests, and the possible execution of filters. The matching data to interest stage is accomplished by the handle `Matching_Data` (Figure 3.3). The handle `Matching_Filter` (Figure 3.5) matches data to programs and dispatches programs execution (Figure 3.6) whenever it is necessary. The resulting (possibly aggregated or filtered) data are delivered to the dissemination layer as a new SOAP data advertisement message to be sent along the network (Figure 3.7).

```
<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:PublishData 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>
        </m:PublishData>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 6 - SOAP message advertising data.

4.4 Step 4 - Cluster Formation

Cluster-based protocols have additional cluster formation and cluster-leader election phases according to a specific algorithm [36]. After nodes are relatively confident that they are aware of their neighbors (organization phase), the next task is to form relationships with nearby nodes resulting in clusters. Clusters should contain a manageable number of nodes that are close [organization tech]. Usually cluster formation algorithms include a step in which nodes declare their interest in joining in a particular cluster as a leader and a further step of deciding which node will be the leader, advertising the chosen node to the other cluster members. Specific code representing the algorithm must be injected in the network in a interest advertising message. Two messages are needed to accomplish the functionality of a generic cluster algorithm: the `Join_cluster` message (Figure 7) is used by nodes advertising their desire of joining in a cluster and the `Advertising_Leader` message (Figure 8) announces the elected cluster leader. `Join_cluster` messages contain the node identification, a timestamp and the node current energy amount. The node energy can be considered or not for the cluster algorithm being used. `Join_cluster` messages can be multicasted or broadcasted in a target area, according to the underlying data dissemination protocol. `Advertising_Leader` messages contain a timestamp and the elected leader node identifier.

```
<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:JoinCluster xmlns:m="http://namespace.example.com">
            <parameter>
                <m0:NodeID>NODE_MAC_ADDRESS</m0:DataValue>
                <m0:TimeStamp>01:16:50</m0:TimeStamp>
                <m0:Energy>
                    <m0:Unit>J</m0:Unit>
                    <m0:Value>0.8</m0:Value>
                </m0:Energy>
            </parameter>
        </m:JoinCluster>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 7 – SOAP message for building a cluster.

```
<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:AdvertisingLeader xmlns:m="http://namespace.example.com">
            <parameter>
                <m0:LeaderID>NODE_MAC_ADDRESS</m0:DataValue>
                <m0:TimeStamp>01:16:50</m0:TimeStamp>
            </parameter>
        </m:AdvertisingLeader>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 8 – SOAP message advertising leader.

5. Discussion

In this section we discuss the features of the proposed middleware system according to the specific requirements sketched for WSN presented in Sections 2.2 and 2.3.

- Efficient Usage of WSN Resources

The content of SOAP messages includes information on node energy and geographical location. Both information are parameters for resource usage optimization algorithms. Application specific code implementing such algorithms is deployed in sensor nodes and triggered by SOAP messages containing some pre-defined data values.

- Robustness and Fault Tolerance

The proposed middleware system is fully distributed, with application specific code deployed in every sensor node and with the information on services provided by the WSN being replicated in every sink node. Such a distributed feature naturally increases the system robustness and fault tolerance.

- Lightweight and Short Storage and Energy Requirements

SOAP is a lightweight protocol, with implementations specifically built for embedded systems. The SOAP messages exchanged through the system as well as the WSDL documents stored in repositories are represented with the XML binary compact format, in order to reduce their length. The use of the compact format also reduces the message payload and thus the energy spent in message transmission.

- Statement and Communication of High Level Tasks and Coordination among Nodes

Functionalities of node coordination as well as the communication of high level tasks are accomplished through generic SOAP messages defined by the middleware system (see Section 4).

- Data Fusion and Data Filtering

Data filtering and aggregation programs can be loaded in sensor nodes during the WSN deployment or they can be injected on-the-fly as SOAP message attachments. Such programs are triggered when pre-defined data arrive in sensor nodes containing the programs code. The trigger data are defined by interest advertising SOAP messages.

- Support to Nodes Heterogeneity

Such a support is a central feature of our proposal. Since our middleware system is based on the ubiquitous XML technologies, we naturally address the interconnection among different sensor nodes in a WSN, or even among different WSNs throughout our middleware layer.

- Awareness and Application Knowledge

User applications and the middleware layer exchange execution context information, such as nodes energy and location, in order to carry out optimization strategies for the efficient use of WSN resources.

6. Related work

There are some projects addressing the development of middleware for WSN, such as [10, 11, 13,14, 15, 16, 17, 19]. The Smart Messages Project [17] is based on agent-like messages containing code and data, which migrate throughout the sensor network. NEST [10] provides microcells as a basic abstraction. They are similar to operating system tasks with support for migration, replication, and grouping. SCADDS [14] is based on a paradigm called Directed Diffusion, which supports robust, data-centric and energy-efficient delivery and in-network aggregation of sensor events. Most of these projects are in an early stage focusing on developing algorithms and components for WSN [12], which might later serve as a foundation for future middleware systems.

In [4] a distributed sensor network middleware service is presented whose purpose is power conservation. Such a service sits on top of the network routing layer and performs data placement and caching as a strategy to conserve battery power. That work does not address the representation of user queries and sensor data.

The Intentional Naming System is an attribute-based name system operating 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 a middleware for sensor networks, INS was designed for more generic mobile networks, offering a sophisticated hierarchical attribute matching procedure. However, they do not address the specific requirements of WSN, nor provide mechanisms which deal with interoperability issues.

Our proposal has some similarities with [48], a database approach for WSN systems. Such a work exploits the sensor computation capabilities to execute part of the query processing inside the network, using query proxies. In their distributed approach, relevant data is extracted from the sensor network, when and where it is needed. The primary difference from our work is that they adopted a relational data base approach, based on XML an SQL queries optimization. Their system performs aggregations in the network as specified by a centrally computed query plan. We propose a totally distributed service approach, based on the ubiquitous standards WSDL and SOAP.

7. Conclusions and Future Works

In this paper, we have presented a middleware service for sensor networks. We claim that the future wireless sensor networks should provide a ubiquitous, standardized access through a common and application independent interface. The contributions of this work are three-fold. First, we propose an interoperability layer separating the data dissemination functionality from the application-specific processing. Second, we have defined an ubiquitous middleware architecture for WSN based on the Web services technology, where sink nodes are modeled as Web Services that expose services provided by the network 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 used by the underlying communication protocol.

We do not couple our proposal to any particular underlying data dissemination protocol. Instead, we provide a generic interface between the middleware layer and the underlying protocol layer.

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 to provide 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, sharing a common interest in a specific application area. Since energy saving is a key element in WSN design, our proposal makes an effort to keep the amount of spent energy in the same level as current WSN systems. It is important to note that energy consumption in data processing in WSNs is assumed to be order of magnitude smaller than in data transmission [19]. 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 exchanges inside the WSN.

Currently, we are working on the implementation of the SOAP module as described in this paper. Our implementation is based on the ESOAP [10], a SOAP implementation version especially designed for embedded systems. We have already defined the WSDL documents for describing the WSN services and the SOAP messages format and content. We expect that the experimental results prove the system feasibility and beside, the total energy spent in transmission and processing do not overcome the values found in current WSN protocols.

8. References

1. Adjie-Winoto, W., Schwartz, E., Balakrishnan, H., Lilley, J., 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):186–201, Dec. 1999
2. Akyildiz, I. et. al. Wireless sensor networks: a survey. Computer Networks, 38(4):393–422, March 2002.
3. Beigl, M., Gellersen, H. and Schmidt, A.. MediaCups: Experience with Design and Use of Computer-Augmented Everyday Objects. *Computer Networks, Special Issue on Pervasive Computing*, 25(4):401–409, March 2001.
4. Bhattacharya, S., Abdelzaher, T. Data Placement for Energy Conservation in Wireless Sensor Networks. Department of Computer Science, University of Virginia. Available in: http://www.andrew.cmu.edu/~weizhang/wsn/documents/fin_sagnik_journal.pdf. Submitted to ICDCS 2002.
5. Capra, L., Emmerich, W. and Mascolo, C. Middleware for Mobile Computing (A Survey). UCL Research Note RN/30/01. Available in: http://www.cs.ucl.ac.uk/staff/L.Capra/publications.html. July 2001.
6. Capra, L., Emmerich, W. and Mascolo, C. Middleware for Mobile Computing (A Survey). UCL Research Note RN/30/01. Available in: http://www.cs.ucl.ac.uk/staff/L.Capra/publications.html. July 2001.
7. Choksi, A., 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 .
8. Coyle, F.P., XML, Web Services, and the Data Revolution. Addison-Wesley Information Technology Series, Addison-Wesley Press, 2002.
9. Czerwinski, S., Zhao, B., Hodes, T., Joseph, A. and Katz, R.,An Architecture for a Secure Service Discovery Service. In *Proc. ACM/IEEE MOBICO*M, pages 24–35, August 1999.
10. ESOAP: Easy Interoperability for Embedded Systems . Available in: http://www.embedding.net/eSOAP/english/brochure/brochure.shtml.
11. Estrin, D. et.al. Next Century Challenges: Scalable Coordination in Sensor Networks. In MobiCom 99, Seattle, USA, Aug. 99.
12. Fielding, R. et al. RFC 2616. Hypertext Transfer Protocol -- HTTP/1.1. June, 1999. Available in: ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt.
13. Graham, S. et al. Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI. Sams Publishing, 2002.
14. Heidemann, J., et.al., Building Efficient Wireless Sensor Networks with Low-Level Naming. In Proceedings of the Symposium on Operating Systems Principles, pp. 146-159. Chateau Lake Louise, Banff, Alberta, Canada, ACM. October, 2001. Available in: http://www.isi.edu/~johnh/PAPERS/Heidemann01c.html.
15. Heidemann, J., et.al.., Diffusion Filters as a Flexible Architecture for Event Notification in Wireless Sensor Networks – USC/ISI Technical Report 2002-556

16. Heinzelman, W., Chandrakasan, A. and Balakrishnan, H, Energy-Efficient Communication Protocol for Wireless Microsensor Networks. Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS '00), January 2000.
17. Howard, A., Mataric, M. and Sukhatme, G.. Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. In *DARS 02*, Fukuoka, Japan, June 2002.
18. IBM White Paper, Web Services Toolkit. Available in: http://www.alphaworks.ibm.com/tech/Webservicestoolkit, April 2002.
19. Intanagonwiwat, C., Govindan, R., Estrin, D. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000), pages 56-67, Boston, MA, USA, Aug 2000.
20. Jini ™. Available in: http://java.sun.com/products/jini/, 1998.
21. Krishnamachari, B., Estrin, D., Wicker, S., Modelling Data-Centric Routing in Wireless Sensor Networks. Available in: http://www2.parc.com/spl/members/zhao/stanford-cs428/readings/Networking/Krishnamachari_infocom02.pdf
22. Kulik, J., Heinzelman, R. B., Balakrishnan, H. Negotiation-based protocols for disseminating information in wireless sensor networks. ACM Wireless Networks 2000. Available in: http://citeseer.nj.nec.com/335631.html, 2000.
23. Microsoft Corporation and Digital Equipment Corporation, "The Component Object Model Specification". Available in: http://www.opengroup.org/pubs/catalog/ax01.htm, October 1995.
24. OMG (Object Management Group). The Common Object Request Broker: Architecture and Specification. Revision 2.0., July 1995.
25. Perkins, C., Service Location Protocol White Paper. Available in: http://playground.sun.com/srvloc/slp_white_paper.html, May 1997.
26. Postel, J., Reynolds, J. RFC 959.FILE TRANSFER PROTOCOL (FTP). Available in: ftp://ftp.rfc-editor.org/in-notes/rfc959.txt, October 1985
27. Qi, H., Kuruganti, P. T. and Xu, Y., The Development of Localized Algorithms in Wireless Sensor Networks, Invited Paper - Sensors 2002, 2, pp. 286-293, 2002.
28. Römer, K., Kasten, O., Mattern, F., Middleware Challenges for Wireless Sensor Networks – ACM SIGMOBILE Mobile Computing and Communications Review, Vol. 6, Number 2, 2002.
29. Silva, F., Heidemann, J., Govindan, R. Network Routing Application Programmer's Interface (API) and Walk Through 9.0, 2002. Available in: http://citeseer.nj.nec.com/silva02network.html.
30. Subramanian, L., Katz, H., An Architecture for Building Self-configurable Systems. Avaliable in: http://www.cs.berkeley.edu/~lakme/sensor.pdf
31. SUN Microsystems, "Enterprise JavaBeans Specification 2.0. Sun Microsystems". [http://java.sun.com/products/ejb/docs.html], August 2001.
32. SUN Microsystems, Implementing Services On Demand and the Sun Open Net Environment (Sun ONE). Available in: http://www.sun.com/software/sunone/wpimplement/wp-implement.pdf, 2001.
33. The 29 Palms Experiment: Tracking Vehicles with a UAV-Delivered Sensor Network. Available in: http://www.eecs.berkeley.edu/_pister/29Palms0103.
34. The Advanced Network Systems Architecture (ANSA). Reference manual, Architecture Project Management, Castle Hill, Cambridge, UK., 1989.
35. UDDI.org, UDDI Technical White Paper. Available in: http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.PDF, September 2000.
36. Ulmer, C. Organization Techniques in Wireless In-situ Sensor Networks. Report. Available in: http://users.ece.gatech.edu/~grimace/research/.
37. Ulmer, C., Alkalai, L. and Yalamanchili, S., 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
38. Universal Plug and Play: Background. Available in: http://www.upnp.com/ resources/UPnPbkground.htm, 1999.
39. W3C (World Wide Web Consortium) Note, "Web Services Description Language (WSDL) 1.1". Available in: http://www.w3.org/TR/2001/NOTE-wsdl-20010315, March 2001.

40. W3C (World Wide Web Consortium) Note, “WAP Binary XML Content Format”. Available in: http://www.w3.org/TR/wbxml/, June 1999.
41. W3C (World Wide Web Consortium) Recommendation, "Extensible Markup Language (XML) 1.0 (Second Edition)”. Available in: http://www.w3.org/TR/REC-xml, October 2000.
42. W3C (World Wide Web Consortium) Recommendation, "XML Schema Part 0: Primer”. Available in: http://www.w3.org/TR/xmlschema-0/, May 2001.
43. 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/.
44. 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/
45. W3C(World Wide Web Consortium) Note on Simple Object Access Protocol (SOAP) 1.1, Available in: http://www.w3.org/TR/SOAP/, May 2000
46. WAP-190-WAESpec. Wireless Application Protocol - Wireless Application Environment Specification Version 1.3, 29-March-2000. Available in: http://www1.wapforum.org/tech/documents/WAP-190-WAESpec-20000329-a.pdf.
47. Wireless Application Protocol Forum: *What is WAP and WAP Forum?* Available in: http://www.wapforum.org /what/index.htm
48. Yong Yao and J. E. Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. Sigmod Record, Volume 31, Number 3, September 2002. Available in: http://www.cs.cornell.edu/ johannes/papers/2002/sigmod-record2002.pdf
49. Yu, Y., Govindan, R., Estrin, D., Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks. Available in: http://citeseer.nj.nec.com/461988.html.