Service-oriented Middleware for Wireless Sensor Networks

Flávia C. Delicato[1,3], Paulo Pires[1,2], José Ferreira de Rezende[3], Luiz Rust da Costa Carmo[1], Luci Pirmez[1]

[1] Núcleo de Computação Eletrônica
[2] Computer Science Department
[3]Grupo de Teleinformática e Automação
Federal University of Rio de Janeiro
P.O Box 2324, Rio de Janeiro, RJ, 20001-970, Brazil
{fdelicato,paulopires}@nce.ufrj.br,
rezende@gta.ufrj.br,{rust,luci}@nce.ufrj.br

Abstract. There is a wide range of applications for wireless sensor networks (WSNs) with different needs. The network infrastructure and data dissemination protocol change according to each specific application requirement. To achieve the best network performance, it is important to adapt the network operation to the application needs. We propose a middleware system for WSNs, which provides a layer between user applications and the network. Such middleware offers an automatic choice of the network configuration and data dissemination strategy.

1 Introduction

The area of Wireless Sensor Networks (WSN) is an emergent research field with wide implications to connect the digital realm to the physical world. WSNs bring new and wide perspectives for monitoring environmental variables. However, they also bring new problems and challenges.

A WSN is composed of dozens to thousands of devices, which have low cost and reduced size, and are able to accomplish sensing tasks and data transmission through wireless connections. Sensors are constrained in terms of energy, storage and processing resources. They act collaboratively, extracting environmental data and transmitting them to user applications.

Since sensors are equipped with limited battery power and usually have to operate unattended for long periods of time, energy consumption is a crucial issue in WSNs. As the largest source of energy consumption in sensors is data transmission, the communication protocols for WSNs propose solutions to minimize the number and range of transmissions, thus extending the network lifetime. Most of the solutions proposed by those protocols are based on multi-hop, short-range communication and adopt some aggregation mechanism to reduce the amount of data to be transmitted.

WSNs can be applied to a wide range of applications with different requirements. Several works [4,11] highlight the importance of the application to participate in the communication process in WSNs. Application-specific optimizations greatly reduce communication costs by replacing communication with computation in the network [4].

Each application class has specific needs and is best assisted by a particular type of communication protocol. Thus, the selection of the most suitable protocol to each application class becomes an important issue. In general, it is difficult for application developers to choose the protocol that better meets their application needs. However, in [4] it was observed that a careful choice of the protocol could reduce the global amount of traffic in the network in up to 50% thus, increasing the network energy efficiency.

Our work is motivated by two main facts: (i) great performance improvements can be reached by the right choice of the communication protocol according to the application requirements and such choice is hard to both express

and accomplish; and (ii) future WSNs probably will serve to different applications. Therefore, we propose a middleware for WSN that acts as a mediator between applications and the WSN and translates application requirements in an efficient choice of network configuration and protocols. The middleware is generic enough to be used over a wide range of applications.

Despite of the advantages of the middleware technology, only few works on WSN are currently considering such technology in the network design [5,8]. WSNs have been built with a high degree of dependency between applications and the underlying communication protocol. Such dependency is justified as necessary to achieve energy efficiency. However, it generates rigid systems, with WSNs specifically designed to particular applications.

The conception of our middleware was leveraged by the recent researches in Web services [3]. The middleware adopts a service approach, in which the WSN is seen as a service provider for user applications. The service provided by the WSN is data collection and delivery. The main services provided by the middleware system are the interpretation of the application needs and the selection of the best network protocol and configuration based on those needs. Besides, since data aggregation is a basic operation, required by all classes of current WSN applications, we propose to leverage the aggregation as a first class operation supplied as one of the middleware services.

The design of the proposed middleware assumes the adoption of a data-centric naming scheme for routing packets in the network [7]. In this scheme, nodes are identified by attributes describing their capabilities or geographical location. Queries issued by applications are also described based on those attributes. Node attributes are used to describe the services supplied by the network. The supplied services are published and accessed through an XML [15] based language. XML messages exchanged inside the network and with the application are formatted and encapsulated through the SOAP protocol [13].

The benefits of the proposed middleware can be appraised under two points of view: the user and the WSN. From the user's point of view, the system offers an interoperability layer between the different applications and the WSN. The middleware creates the abstraction of a generic WSN that provides services for several applications. Those services can be accessed in a flexible way and through any high level language. Furthermore, through the use of XML language and SOAP protocol, both *de facto* web standards, the proposed middleware naturally provides interoperability between the WSN and the Internet. Moreover, SOAP is a lightweight protocol, incurring a low communication and processing overhead. From the network point of view, the system provides mechanisms to obtain the best match between communication protocols and application requirements.

This paper presents the architecture of the proposed middleware and the description of its operation. Besides, the implementation of a prototype is described and some simulation results are presented. The remaining of this work is organized as follows. Section 2 presents the grounding concepts used in our proposal. Section 3 contains the description of the proposed system. Section 4 describes the system evaluation. Section 5 presents the related works. Section 6 contains the conclusion and future work.

2 Background Concepts

This Section presents the grounding concepts used in our proposal, which include the general WSN architecture and a brief discussion on communication and data delivery models adopted in WSNs.

2.1 WSN Architecture

A WSN is composed of three main organizational components: the infrastructure, the protocol stack and the application [11]. The infrastructure consists of the WSN nodes and their current deployment status in the environment. In general, a WSN has several sensor nodes and one or more sink nodes that play the role of exit points from the network. Sink nodes have more processing power than sensor nodes and they are not energy

constrained. They act as an entry point for submission of user interests and as a concentrator of the information sent by sensors. Sensor nodes contain one or more sensing devices and have limited processing and storage capabilities. Their function is to collect, aggregate and transmit their own data and their neighboring data. The protocol stack used by sink and sensor nodes consists of the application, transport, network, data link and physical layers. The application is responsible for issuing queries or interests, describing the physical phenomena to be studied by the user.

In general, optimization procedures in the WSN cut across these three organizational levels. For example, application features may influence on the network infrastructure and on the adopted protocols. Therefore, the network might use the application level knowledge to obtain larger energy efficiency, thus extending its lifetime.

2.2 Communication Types

Usually, a WSN supports two communication types [11]: infrastructure and application. The infrastructure communication is responsible for the communication needed for WSN configuration, management and optimization. It is generated by the network and link protocols in response to the application requirements or to events occurred in the network.

The application communication relates to both the transfer of sensed data, from data sources to the user, and the transfer of the user interests to the WSN. The user interest is the starting point from which the data collection takes place. Ideally, the interest is specified in terms of the monitored phenomenon. The adoption of a data-centric naming scheme allows interests to be described by a set of attributes, such as the sensor type, the geographical area to be monitored, and the data rate.

2.3 Data Delivery Models and Data Dissemination Protocols

The data delivery model used by the network governs the generation of the application traffic. [11] classifies WSNs in four types according to the data delivery model required by the application: continuous, event-driven, observer-initiated and hybrid.

In the continuous model, sensors communicate their data continually at a predefined rate. In the event-driven model, the sensors notify data only if an event of interest occurs. In the observer-initiated model, the sensors only notify their results in response to an explicit request from the application. Finally, the three approaches can coexist in the same network, generating a hybrid delivery model. The data delivery model dictates the best type of data dissemination protocol to be adopted.

In general, the data dissemination protocols can be classified according to their dissemination approach in: broadcast-based, unicast and multicast [11]. The interaction between the data delivery model and the type of dissemination protocol to be adopted might have a significant impact on the network performance, in terms of data loss, delay and energy consumption [11]. One of the main goals of the proposed middleware is to promote such interaction, in a flexible and transparent way for the user.

3 System Description

The term middleware is widely used to denote a layer comprised of generic services sitting below user applications. A middleware seeks primarily to hide the underlying complexity of the networked environments by insulating applications from dealing with explicit protocol handling, network faults, and parallelism issues.

In particular, a middleware for WSN has to insulate applications from the specific features of the underlying infrastructure and network protocols. It has to include mechanisms for formulating high-level sensing tasks,

communicating those tasks to the WSN and coordinating sensors to perform them. Besides, it must be robust and fault tolerant, and it must have short computational requirements.

The evidences that the WSN adaptation to the application requirements promotes the best network performance, as well as the belief that future WSNs will serve to different applications, motivated us to propose a middleware for WSNs. Such middleware offers a layer that interacts both to the different applications and to the WSN. We argue that the needed interaction between applications and the WSN to achieve energy efficiency should be performed through an application independent interface, which translates application requirements to network configuration without breaking the layered network structure.

Our middleware resides between the application and networking layers and it provides a high level representation for: (i) application interests; (ii) sensor generated data, and (iii) in-network aggregation functions. Such representation is based on the concept of services and it adopts XML as the interface description and encoding language. SOAP is adopted as the protocol for formatting XML messages and transporting invocations between peers by using a data-centric underlying protocol.

3.1 System Architecture

In this work, we assume a WSN composed of several sensor nodes and one or more powerful sink nodes. The interaction between applications and the WSN occurs in the sink nodes. The basic design of our middleware is shown in Figure 1. We separate its functionality into two main components: an adaptation module and a SOAP module. The adaptation module runs only in sink nodes, and it is responsible for receiving application requests and for establishing the WSN mission. WSN mission basically consists of determining the data delivery model according to the application needs. The adaptation module is in charge of defining the logical topology (flat or hierarchical) and the data dissemination strategy to be adopted for the current mission. This module communicates with the SOAP module to receive the application input messages, which are delivered to a decision unit. Also, it has an XML-based API to communicate with the underlying dissemination protocol in order to activate its execution according to the current WSN mission.

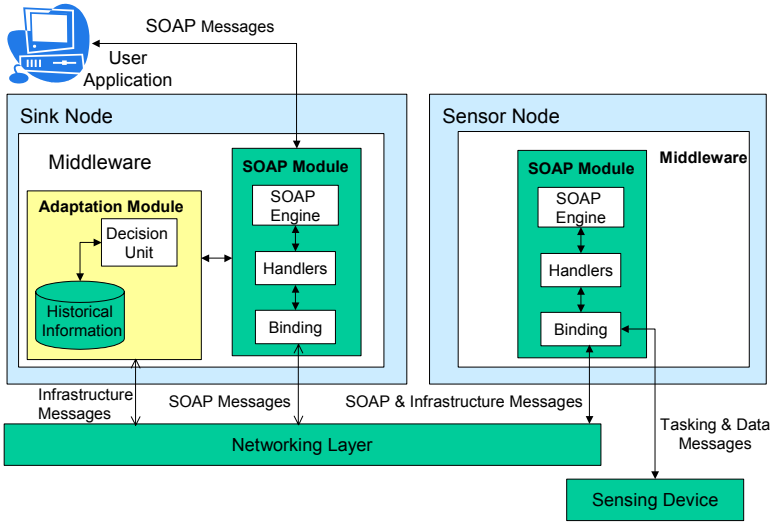

Fig. 1. Sink and Sensor Node components

Once being aware of the current WSN mission, the underlying communication sub-layer generates all the infrastructure communication needed to set the network topology and to establish the data dissemination strategy.

Control messages generated by the infrastructure communication are not represented in XML. Only application communication is XML-based.

The SOAP module runs in both sink and sensor nodes and it is responsible for converting messages to/from XML and for coordinating all operations for publishing and invoking services. The SOAP module is composed of three main components: the SOAP engine, a set of handlers and a binding with the underlying protocol. The SOAP engine acts as the entry point into the SOAP module. It coordinates the SOAP messages flow through the several handlers and ensures that the SOAP semantics is followed. Handlers are the basic building blocks inside the SOAP module and represent the message processing logic. Four handlers are defined: basic handlers, `Parse_Interest`, `Matching_Data` and `Matching_Aggregation` handlers. Basic handlers are responsible for XML parse and composition, header processing, and data type conversions. The `Matching_Data` handler is specifically built for sending and receiving messages through an underlying interest-based, data-centric protocol. It performs a matching function between sensor generated data and a previous received user interest. The `Parse_Interest` handler performs a matching function between a received interest and the sensor description (type and location). The `Matching_Aggregation` handler represents data for the activation of in-network aggregation functions. An aggregation function is triggered whenever a data, generated or received in a sensor node, matches a set of specified attributes.

Bindings act as SOAP drivers, translating messages in protocol specific formats to SOAP/XML. Since the middleware is designed to work above different network infrastructures, a binding has to be implemented for each protocol. There are two different kinds of bindings. If the underlying protocol uses the message content in its routing decisions, the most efficient binding consists of translating the SOAP message to the protocol format and vice-versa. We call this a **type one** binding. If the protocol conveys the payload without considering its content, the binding only sets the header fields and inserts the XML/SOAP message directly into the packet payload (**type two** binding).

3.2 System Operation

The proposed middleware allows a WSN to be tailored for a mission according to the application needs. Such approach provides greater flexibility in the WSN usage.

The middleware adopts a service-based approach. A user application querying data from a WSN plays the role of a service requestor. Sink nodes act as service providers to the external environment, providing the service descriptions of the whole WSN. At the same time, they act as requestors to the sensor nodes, by requesting their specialized services to meet the application needs. Sensor nodes are service providers. They send their service descriptions to sink nodes.

In the service oriented approach, operations are defined for each service provider. According to the operations, messages are exchanged to invoke such operations. In the proposed system, the service providers are Sink Nodes and Sensor Nodes. The system operations and messages used to invoked them are described in Section X.

Applications submit interests to the WSN through sink nodes. Interests are represented in XML language and packetized as SOAP messages. There are tools [6] that automatically translate method calls in the application language to SOAP/XML messages according to a predefined format. Interests are used to determine the data delivery model dictated by the application and they may be classified in synchronous or asynchronous. A synchronous interest corresponds to a simple query on the current state of a monitored phenomenon. Asynchronous interests correspond to long-running queries or queries on the occurrence of a particular event of interest.

It is important to point out that the choice of XML language to represent application interests and sensor data was motivated by two main reasons. First, XML is recognized as a standard language for information exchanged in the web. Second, a WSN needs a language which is able to represent both queries and sensor tasks. The XML language offers flexibility enough to achieve this goal.

Our system operates according to four main stages: (i) publication of sensor description; (ii) submission of application interests; (iii) establishment of the WSN mission and configuration; and (iv) data advertisement.

Stage 1: Publication of Sensor Description. After the WSN deployment, nodes exchange SOAP messages describing their status ("ready"/"malfunction") and the services supplied by them. SOAP *Configuring* messages include node identification, sensor type, geographical location, residual energy, maximum and minimum confidence degrees, maximum and minimum acquisition intervals.

The adaptation module needs to know the current available sensors and their features to make its configuration decisions. At the same time, applications need to know which services the WSN offers to access them. A multi-hop protocol, which is not still an optimized solution to the specific application requirements, is used to forward these messages towards sink nodes. In large WSNs, more than one sink need to be active, each one responsible for a sub-set of sensor nodes, and they exchange control messages to keep their database consistent and to coordinate their work.

Stage 2: Submission of Application Interests. Once the application is aware of the available WSN services, it is able to submit its interests, through the different kinds of SOAP *Subscribe* messages. Messages representing synchronous interests contain the sensor type and the geographic coordinates of the target area. The ones representing long-running queries indicate the sensor type, the target area, the data acquisition duration and rate. SOAP messages representing event-driven queries indicate the target area, the sensor type and the event to be monitored.

Applications may request the use of aggregation functions. A SOAP *Aggregation Request* message contains an identifier and a list of data types with their respective values. The identifier is used to trigger the execution of the appropriate aggregation service in the middleware layer of the node, when such node receives data matching the values specified in the request.

Stage 3: Establishment of the WSN Mission and Configuration. The WSN user is often an expert in his/her own application domain, not in computer networks. Therefore, it is difficult for this user to choose the network protocol and organization more suitable for his/her requirements. So, the proposed middleware offers an automatic decision process, by using the application inputs to guide the choice of the best solution for each application. Such choice is based on results of previous works in the field, reported by researchers. The adaptation module keeps a database with historical information of the different WSN configuration choices for each set of application requirements. The database also holds average values for WSN performance metrics, such as delay and accuracy, for each executed application. The decision unit of the adaptation module uses information from the database as inputs for its decision algorithm. Since the decision algorithm has many parameters which are subjective, such as "if there are **a lot of** data sources", we are currently exploring a fuzzy model to ease its design and improve the results.

Concerning the choice of the data delivery model, an observer-initiated model best supports synchronous interests. Long-running queries are best served by a continuous model, while event-driven queries indicate an event-driven model. Sometimes, an asynchronous interest specifies the need for a continuous monitoring while at the same time it requests for the notification of a particular event. Such requirements indicate the need of a hybrid model.

The query type, the size of the target area and the estimated number of data sources and sinks are used for the decision algorithm to decide among six different strategies of data dissemination and network topologies: (i) flat topology with dissemination based on directed diffusion pull mode [4]; (ii) flat topology with dissemination based on directed diffusion push mode (with or without geographical forwarding) [4]; (iii) flat topology with unicast communication directly to sink nodes; (iv) unicast with cluster formation and one-level of cluster-heads hierarchy; (v) unicast with cluster formation and two-levels of cluster-heads hierarchy; and (vi) multicast.

Currently, the decision unit of the proposed middleware adopts the algorithm below.

```
Receive interest
If interest is synchronous (simple query)
   Verify target area
   If target area is near the Sink node
      If there is a lot of sensors in the target area
```

```
        Use diffusion
      Else
        Use unicast with direct communication with the sink node
  Else
    Use directed diffusion with geographical forwarding (GEAR [18])
If interest is asynchronous
  If it is a long-running query (periodic data sent)
    If there is a lot of sensors in the target area
      Use a cluster-based protocol
      Determine the number of clusters
      Determine the maximum number of nodes inside each cluster
      Elect the cluster leaders - CHs (consider the residual energy and location)
      If all cluster leaders are in the range of the Sink Node
        Use a one-level hierarchy of cluster-heads (CHs transmit direct to Sink Node)
      Else
        Use a two or more levels hierarchy of cluster-heads (function of node density and transmission range)
    Else
      Use a plain topology with greedy forwarding (for example Pegasis [10])
  If it is a event-driven query
    If the target area has a lot of sensors (many sources)
      If there is less than N sink nodes
        If the sources are near (clustered sources) and there is no geographical information
          Use a cluster-based protocol
        Else
        Use two phase pull diffusion with GEAR [18])
        Else
        Use a cluster-based protocol
    Else (few sources)
      Use push diffusion [4]
```

Fig. 2. Middleware Decision Algorithm

The decisions taken by the middleware are passed to the network protocol that triggers the needed infrastructure communication. After using the information extracted from the *Subscribe* message to decide about the WSN configuration, such message is propagated to sensor nodes, according to the chosen strategy for data dissemination.

Stage 4: Data Advertisement. Since the proposed system adopts an interest-based approach, a sensor only sends SOAP *Data Advertisement* messages if it had received a previous message advertising interests that match its own data type.

When a sensor generates data, this data is passed via binding to the SOAP module. The handler responsible for the XML composition converts the data into a SOAP *Data Advertisement* message and delivers it to the `Matching_Data` handler. This handler verifies if the data matches with some received interests. If there is a match, the data is passed to the `Matching_Aggregation` handler. Such handler verifies if the data attributes match with the attributes specified in any aggregation request. If there is a match, the data is delivered to the service that implements the corresponding aggregation function. The resulting aggregated data is passed to the dissemination protocol, via binding, as a new SOAP *Data Advertisement* message to be sent along the network.

The dissemination protocol checks incoming packets to verify if they contain data or control message. If it is a control packet, it is directly processed by the data dissemination protocol. If it is a data packet, it is passed to the SOAP module. **Type two** binding SOAP messages are handled by the XML parser handler, which verifies the message type, that is, if it is a *Subscribe* or a *Data* message. If it is a *Subscribe* message, it is sent to the `Parse_Interest` handler, which verifies if the received interest matches with the node description. In the affirmative case, the handler stores the interest in a local repository. Otherwise, it passes the message to the dissemination protocol that decides about the interest forwarding. If it is a *Data* message, it has the same processing of a local generated data.

3.3 Operations and SOAP Messages

In the proposed system the operations defined for Sink Nodes are listed below.

Query_Sensors - used by an application in a sink node to know the available sensor types in a WSN. This operation is represented by two SOAP messages: an input message without parameters and an output message with the response.

Subscribe_Asynch_Interest - used by an application to submit a query which represents an asynchronous interest to a sink node.

Subscribe_Synch_Interest - used by an application to submit a query which represents a synchronous interest to a sink node.

Subscribe_Hybrid_Interest - used by an application to submit a query which represents a hybrid data delivery model. It is not implemented in the current version of the system.

Subscribe_Triggering_Interest - used by an application to submit a triggering task, such as the on demand activation of specified sensor types, when a given event occurs.

The operations defined for Sensor Nodes are below.

Publish_SensorDescription - used by the sensor node to create and disseminate a SOAP message containing its service descriptions.

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

The corresponding SOAP message generated are shown in the figures below.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" >
  <SOAP-ENV:Body>
    <m:SubscribeAsynchInterest xmlns:m="http://namespace.example.com">
      <parameter>
        <m:SensorType>Temperature</m:SensorType>
        <m:DataRate unit="mSeconds">20</m:DataRate>
        <m:Duration unit="Seconds">20</m:Duration>
        <m:Accuracy>0.1</m:Accuracy>
        <m:Area>
          <m:PointA unit="LatLong">
            <m:x>35.00</m:x> <m:y>-23.00</m:y>
          </m:PointA>
          <m:PointB unit="LatLong">
            <m:x>35.02</m:x> <m:y>-23.03</m:y>
          </m:PointB>
        </m:Area>
        <m:Constraint>
          <m:value>35.00</m:value>
          <m:operation>GT</m:operation>
        </m:Constraint>
      </parameter>
    </m:SubscribeAsynchInterest >
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 3. SOAP message advertising an Asynchronous Interest (Subscribe_Asynch_Interest operation).

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" >
  <SOAP-ENV:Body>
    <m:SubscribeSynchInterest xmlns:m="http://namespace.example.com">
      <parameter>
        <m:SensorType>Temperature</m:SensorType>
        <m:Area>
          <m:PointA unit="LatLong">
            <m:x>35.00</m:x> <m:y>-23.00</m:y>
          </m:PointA>
```

```
            <m:PointB unit="LatLong">
               <m:x>35.02</m:x> <m:y>-23.03</m:y>
            </m:PointB>
         </m:Area>
         <m:Accuracy>0.1</m:Accuracy>
         <m:MaxDelay unit="mSeconds">0.5</m:MaxDelay>
         <m:Constraint>
            <m:value>35.00</m:value>
            <m:operation>GT</m:operation>
         </m:Constraint>
      </parameter>
   </m:SubscribeSynchInterest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 4. SOAP message advertising a Synchronous Interest (Subscribe_Synch_Interest operation)

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" >
   <SOAP-ENV:Body>
      <m:InterestTrig xmlns:m="http://namespace.example.com">
         <parameter>
            <m:Area>
               <m:PointA unit="LatLong">
                  <m:x>35.00</m:x> <m:y>-23.00</m:y>
               </m:PointA>
               <m:PointB unit="LatLong">
                  <m:x>35.02</m:x> <m:y>-23.03</m:y>
               </m:PointB>
            </m:Area>
            <m:mainSensor>
               <m:SensorType>Motion</m:SensorType>
               <m:DataRate unit="mSeconds">40</m:DataRate>
            </m:mainSensor>
            <m:DataType>Four Legged Animal</m:DataRate>
            <m:secundarySensor>
               <m:SensorType>Imagery</m:SensorType>
            </m:secundarySensor>
         </parameter>
      </m:InterestTrig>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 5. SOAP message advertising an Interest for triggering task (Subscribe_Triggering_Interest operation).

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <m:QuerySensorIn xmlns:m="http://namespace.example.com">
         <parameter> </parameter>
      </m:QuerySensorIn>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 6. SOAP message Query_SensorsIn: input message for a Query_Sensors operation.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <m:QuerySensorOut xmlns:m="http://namespace.example.com">
         <parameter>
            <m:SensorType>Motion</m:SensorType>
            <m:Confidence> <m:Max>1.0</m:Max>    </m:Confidence>
            <m:DataRate unit="mSeconds"> <m:Max>10</m:Max>  </m:DataRate>
```

```
        </parameter>
      </m:QuerySensorOut>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 7. SOAP message Query_SensorsOut: output message for a Query_Sensors operation.

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

Fig. 8. SOAP configuration message (used to invoke the Publish_SensorDescription operation).

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
   <SOAP-ENV:Body>
      <m:PublishData xmlns:m="http://namespace.example.com">
         <parameter ID="NODE_MAC_ADDRESS">
            <m:DataValue>Elephant</m:DataValue>
            <m:Location unit="LatLong">
               <m:x>35.00</m:x> <m:y>-23.00</m:y>
            </m:Location>
            <m:Intensity>0.6</m:Intensity>
            <m:Confidence>0.85</m:Confidence>
            <m:Energy>0.9</m:Energy>
            <m:TimeStamp>08:16:40</m:TimeStamp>
         </parameter>
      </m:PublishData>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 9. SOAP message for advertising data (Publish_Data operation)

4 System Evaluation

We evaluated the proposed system according to two different approaches: a qualitative and a quantitative. The goal of the qualitative evaluation was to implement the main building blocks of the middleware architecture and to analyze the interaction between applications and the system. We also analyzed the impacts of adopting the SOAP/XML representation. In the quantitative evaluation we performed simulations to show that the adoption of a network protocol that meets application requirements can improve the WSN overall performance. The WSN performance can be evaluated according to different metrics, such as average delay, average dissipated energy and event delivery ratio [7]. In this first stage of work we are interested in measuring the average dissipated energy, which is a ratio of the total dissipated energy per node in the network to the number of distinct events delivered to the sink node, and it is directly related to the WSN lifetime [7].

Qualitative Evaluation. As a proof of concept for our system, the proposed middleware was implemented using Java Language and the Apache Axis implementation of the SOAP protocol [16]. Sensor and sink node services were created as Java classes implementing all defined operations. WSDL (Web Services Description Language [12]), an XML extension specific for Web Services, was used for creating a document describing services provided by the WSN. To access the WSN services, applications get the WSDL document describing them. There are tools [16] to read WSDL documents and automatically generate method calls in the application native language to invoke the available operations. An event-based application was implemented in Java to issue queries to the WSN sink node and to get back the results. The main goal of building the application was to validate the high level interaction between applications and the WSN and to check the operation invocation and the message delivery to the respective SOAP handlers. The Java source code and the WSDL document can be downloaded from: http://www.nce.ufrj.br/labnet/research/networksensors/software.htm.

To prove the feasibility of adopting our middleware system in a real WSN, it is important to evaluate its impact in the WSN global energy spent. The more prominent source of energy consumption in WSN is data transmission. Energy spent in processing is considered negligible. From Section 3.1, we saw that when adopting the type one binding, data messages are sent throughout the WSN in the original network protocol format thus, not incurring extra communication overhead. When using the type two binding, SOAP messages are directly encapsulated into the packet of the underlying protocol. Therefore, depending on the original message format according to the protocol, the use of XML/SOAP may generate an increase in the packet size. A SOAP message for advertising data, independent on the data content, has 260 Bytes. There are protocols that adopt message sizes smaller than 260 Bytes, and in those cases the use of XML can generate an increase in the dissipated energy. To overcome that problem, when using the type two binding, we adopt the XML compact binary format inside of the WSN [14]. Such format reduces the message size in up to 50%, to the price of a smaller portability, what is not a problem in the controlled environment of WSNs. Applications continue to adopt standard XML.

Quantitative Evaluation. In this stage, we addressed the improvements that can be reached when using a network protocol chosen according to the application needs. Directed diffusion (DD [7]) is a broadcast-based protocol specific for WSN that uses the content of data messages to configure gradients that drives the data forwarding. We implemented a type one binding to be used with the directed diffusion protocol. The binding converts XML messages to the format adopted by DD, which consists in attribute vectors in C++ language, and vice-versa. We implemented the sub-part of the decision algorithm responsible for deciding between two modes of DD: pull and push. In pull diffusion, sink nodes periodically send interests and generated data are propagated in response to such interests. Since interests are flooded into the WSN, with a large number of sinks the protocol tends to have a high communication overhead. In push diffusion, sinks become passive, with interest information kept local to the node subscribing to data. The main factors that influence the choice between pull and push diffusion are the number of sink and source nodes and the data rate requested by the application. We varied these parameters through simulations and the results were used to refine the decision algorithm.

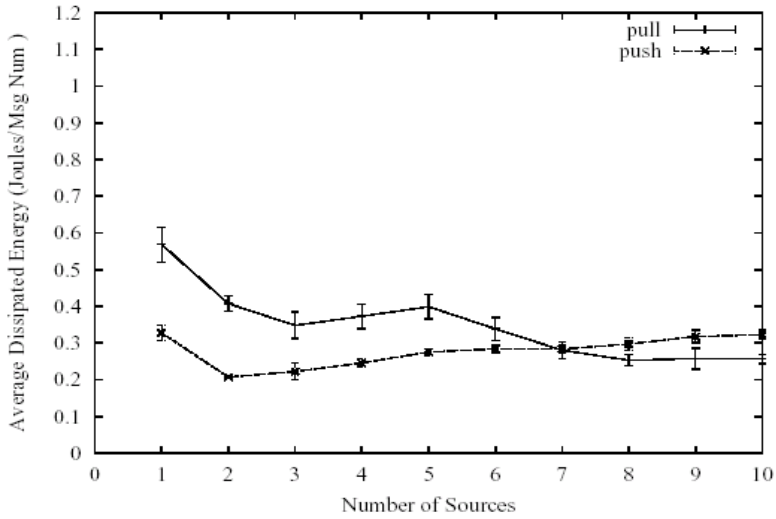

Fig. 10. Average Dissipated Energy for Different Number of Sources.

We generated scenarios with 50 and 100 nodes. The adopted energy dissipation model was as described in [7]. The radio range was set to 40m. The error bars shown in the figure represent a confidence interval of 95%. The variation of the number of sink nodes did not present a significant difference between pull and push. For number of source nodes varying from 1 to 7, the push mode dissipated less energy. Starting from 7 source nodes, pull mode presented the lowest values of dissipated energy (Figure 10). To evaluate push and pull modes against the data rate, the interest and exploratory data sending rates were kept as the original protocol, respectively 30 and 60 seconds [7]. The data message size is 64B and the interest message size is 36B. In such conditions, pull and push dissipated the same energy amount in high data rates (1 data each 5 seconds). With lower rates, pull dissipated about 25% more energy than push, in other words, the WSN works more to deliver useful information to the application (Figure 10).

To sum up, the middleware decision algorithm should base its choice between pull and push on two main factors: (i) if there are **many** potential source nodes, use pull diffusion; (ii) if the data rate requested by the application is **very low**, use push diffusion, or adjust the sending rates of control messages according to the data rate.

5 Related Work

MILAN [5] is a middleware for WSNs that receives a description of application requirements and choose the best sensor and network configuration that meets such requirements while maximizing the network lifetime. MILAN incorporates state-based changes in application needs and it manages the network conditions along the time. It does not address the issue of providing a standard representation for application needs and sensor generated data. In its first version, MILAN adopts a totally centralized approach, which is not appropriate for large WSNs.

In [17] the sensor computation capabilities are exploited to execute part of the query processing inside the network, by using query proxies. In [1], an SQL-like declarative language is proposed for users who submit queries to sensor networks. In [2], a sensor network architecture based on concepts of virtual databases and data-centric routing is proposed. The main difference between such works and ours is that we propose a service-based approach, which is totally distributed and relies on the ubiquitous XML and SOAP standards. By exposing sensor functionalities as services we offer a more flexible architecture when comparing to SQL queries. Besides that, SQL

is basically a query language, while XML can be used both as a query and a task language, making it a more appropriate representation mechanism for WSNs. Moreover, to the best of our knowledge, our proposal is the first that adopts a mission-oriented approach aiming to adapt the WSN configuration and protocols to each particular application requirements.

6 Conclusion and Future Work

We presented a middleware for wireless sensor networks that adopts a service-based approach, allowing a mission oriented usage of WSNs. This work gives the following contributions.

First, a layer is proposed between applications and the WSN. This layer hides from applications detailed information about the WSN infrastructure and protocols, and it allows the WSN to be tailored to the different applications. This layer offers an abstract view of the network as a service provider and it provides access to such services.

Second, an automated decision mechanism is supplied that establishes the best network configuration according to the current WSN mission dictated by the application. Such mechanism has the primary goal of taking decisions about low-level mechanisms that are often outside the scope of the application developer. Its decisions aim to provide an efficient usage of the WSN and to extend its lifetime.

Third, the adoption of XML language and SOAP protocol, both Internet standards, is proposed as the mechanisms for representing all the application communication.

The middleware was implemented in Java as a proof of concept of the proposed architecture. Simulations to evaluate the behavior of the system were performed and the results were satisfactory. Since some parameters adopted in the decision algorithm are very subjective, we are developing a fuzzy model to handle a large range of possible values of each parameter.

One interesting advantage of the proposed system appears when more than one protocol is simultaneously available and the best suitable option is transparently chosen, based on information from the interest posed by the user.

This work takes a first step towards the building of flexible, generic, and energy efficient WSNs.

References

1. Bonnet, P., Gehrke, J. E., Seshadri, P.: Towards Sensor Database Systems. In Proc. of the 2nd International Conference on Mobile Data Management. Hong Kong, Jan 2001.
2. Govindan, R. et. al: The Sensor Network as a Database. Available in http://www-robotics.usc.edu/~gaurav/CS546.
3. Graham, S. et al.: Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI. Sams Publishing, 2002.
4. Heideman, J., Silva, F., Estrin, D.: Matching Data Dissemination Algorithms to Application Requirements. ISI-TR-571. 16 April 2003.
5. Heinzelman, W. et al.: Middleware to Support Sensor Network Applications. IEEE Network Magazine Special Issue. Jan 2004.
6. IBM White Paper. Web Services Toolkit. Available in: http://www.alphaworks.ibm.com/tech/ Webservicestoolkit. April 2002.
7. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: a scalable and robust communication paradigm for sensor networks. In Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking, pp. 56-67, MA, USA. Aug 2000.
8. Liu, T. and Martonosi, M.: Impala: A Middleware System for Managing Autonomic Parallel Sensor Systems. PPoPP'03 - ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, CA, USA. June 2003.
9. NS-2 (The Network Simulator version 2). Available in: http://www.isi.edu/nsnam/ns/.

10. Tan, H, Körpeolu, I.: Power efficient data gathering and aggregation in wireless sensor networks. ACM SIGMOD Record, Vol. 32 , Issue 4, SPECIAL ISSUE: Special section on sensor network technology and sensor data management. Pages: 66 – 71, ISSN:0163-5808. Dec 2003.
11. Tilak, S., Abu-Ghazaleh, N., Heinzelman, W.: A taxonomy of wireless micro-sensor network models. Available in http://citeseer.nj.nec.com/tilak02taxonomy.html. 2002.
12. 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.
13. W3C Note on Simple Object Access Protocol (SOAP) 1.1. Available in: http://www.w3.org/TR/SOAP/. May 2000.
14. W3C Note. WAP Binary XML Content Format. Available in: http://www.w3.org/TR/wbxml/. 1999.
15. W3C Recommendation. Extensible Markup Language (XML) 1.0 (Second Edition). Available in: http://www.w3.org/TR/REC-xml. Oct. 2000.
16. Web Services Project Apache. Available in: http://ws.apache.org/.
17. Yao, Y., Gehrke, J. E.: The Cougar Approach to In-Network Query Processing in Sensor Networks. Sigmod Record, Volume 31, Number 3. Sept. 2002.
18. Yu, Y., Govindan, R. and Estrin, D.: Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks. UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023. May 2001.