Hindawi Publishing Corporation
Journal of Sensors
Volume 2015, Article ID 621326, 14 pages
http://dx.doi.org/10.1155/2015/621326

Research Article

Autonomic Context-Aware Wireless Sensor Networks

Nídia G. S. Campos,[1,2] Danielo G. Gomes,[1] Flávia C. Delicato,[3] Augusto J. V. Neto,[4] Luci Pirmez,[3] and José Neuman de Souza[1]

[1] *Group of Computer Networks, Software Engineering and Systems (GREat), Federal University of Ceará, Pici Campus, Avenida Mister Hull, s/n, Bloco 942-A, 60.455-760 Fortaleza, CE, Brazil*
[2] *Telematics Department, Federal Institute of Ceará, Fortaleza Campus, Avenida Treze de Maio, 2081 Benfica, 60.040-531 Fortaleza, CE, Brazil*
[3] *CCMN, Federal University of Rio de Janeiro, Avenida Athos da Silveira Ramos, Cidade Universitária, 21.941-590 Ilha do Fundão, RJ, Brazil*
[4] *Informatics and Applied Mathematics Department, Federal University of Rio Grande do Norte, Lagoa Nova Campus, 59.078-970 Natal, RN, Brazil*

Correspondence should be addressed to Nídia G. S. Campos; nidiacampos@great.ufc.br

Received 4 November 2014; Accepted 21 April 2015

Academic Editor: Banshi D. Gupta

Autonomic Computing allows systems like wireless sensor networks (WSN) to self-manage computing resources in order to extend their autonomy as much as possible. In addition, contextualization tasks can fuse two or more different sensor data into a more meaningful information. Since these tasks usually run in a single centralized context server (e.g., sink node), the massive volume of data generated by the wireless sensors can lead to a huge information overload in such server. Here we propose DAIM, a distributed autonomic inference machine distributed which allows the sensor nodes to do self-management and contextualization tasks based on fuzzy logic. We have evaluated DAIM in a real sensor network taking into account other inference machines. Experimental results illustrate that DAIM is an energy-efficient contextualization method for WSN, reducing 48.8% of the number of messages sent to the context servers while saving 19.5% of the total amount of energy spent in the network.

1. Introduction

Smart environment can be defined as a digital one exploiting the integrated use of Information and Communication Technologies (ICTs) for helping people in a sensitive way to meet improved quality of life. Within the scope of smart environments, a city can be defined as "Smart" when investments in human/social capital and traditional/modern ICTs networked infrastructures fuel sustainable economic development, with a wise management of natural resources through participatory action and engagement of citizens [1].

In Smart Cities, applications can exploit multiactors, multisector, and multilevel perspectives taking advantage of the opportunities ICTs offer, such as wireless mobile networking, speech/image recognition, sensors/actuators, and intelligent decision-making systems [2]. For instance, a Smart City system can provide online information from the road authorities about traffic jams and accidents to the users' cars system which will search for the best route, avoiding problematic areas [3].

"*Context is any information that can be used to characterize the situation of entity*" [4]. Smart City applications are context-sensitive in nature; hence one of the main roles on such environments regards the potentials in producing key information (context) from the environment (entity) to enable applications with proactive decisions (instead to reactive ones of traditional cities) for anticipated problem solutions.

To achieve this goal, a Smart City system must support a context-aware approach, embedding mechanisms for gathering sensor data, processing sensor data to turn raw data into powerful insights (i.e., context), and making them available with easy access. The contextualization [5] process encompasses data analytics towards provisioning context

information that semantically describes complex environment insights, such as situations, behaviors, conditions, and preferences.

The sensing systems in the urban environment deploy integrated telecommunications (including mobility), a large variety of smart sensors, electronics, and information technologies able to gather data from multiple sensing sources of diverse approaches, and also to efficiently automate industrial processes up to a self-regulating optimization. In this scope, Wireless Sensor Networks (WSN) attract special attention by their ability to cope with the herein above requirements, especially in terms of gathering diverse environmental data. For instance, raw data of temperature, humidity, and light sensors can be fused in order to produce high-level context information that semantically contextualizes a high fire risk insight in the targeted environment. Without context-awareness, the task to provide vehicular situational and conditional information is very hard [6].

The contextualization process is a hard task, since it must be deployed appropriately to allow achieving meaningful information with high precision and accuracy, thus enabling effective application reactions. The contextualization can be achieved by means of statistical methods, machine learning, and/or Artificial Intelligence (e.g., fuzzy logic inference rules [7], ASP (Answer Set Programming) and ontologies [8], natural language inference rules [9], and OWL (Ontology Web Language) and CLIPS (C Language Integrated Production System) facts and rules [10]).

Typically, the contextualization process involves the presence of a context provider. The context provider is the entity system in charge to make available context information for end systems (users and applications) of the smart environment, being a key issue in context-aware approaches. Centralized context provider (a.k.a. context server) approach is widely adopted; however it imposes several performance issues [11, 12].

On the one hand, context servers overload exponentially with the increasing number of sensors gathering data which is a central problem in current big data scenario. On the other hand, the heterogeneity of sensor data (different types of information) requires adopting more emerging and complex processing techniques for achieving improved contextualization, which can limit the efficiency of mission critical systems due to the increased latency and response time [9, 10].

One possible and rarely employed solution consists of pushing the contextualization task to WSN as a distributed context provider approach, using a lightweight computational intelligence technique to overcome the performance issues of centralized context providers. Fuzzy logic can be used in resource constrained networks like WSN, seeking to handle raw data of faulty and/or imprecise sensors, since these values are transformed into meaningful information by the contextualization task [13].

In such distributed context provider approach, the computational restrictions (low memory, processing, storage, and communication capacities) of WSN nodes require optimizations to fulfill the performance demands of Smart City applications. Moreover, WSN nodes have very limited battery capacities, and they can be placed in areas with difficult access for humans. The self-management feature of the Autonomic Computing [14] can be adopted to allow the WSN to work as a biological system, that is, without administrators (human) intervention as much as possible.

An autonomic sensor element (ASE) [15] brings the self-management feature to WSN when it is embedded in WSN nodes. ASE makes WSN nodes able to keep and adjust their own operations based on their external and internal behavior at runtime, to implement the required corrective administrative actions without endangering or corrupting ongoing operations, as an Autonomic Computing system [16]. ASE can still help sensor node in its relationship with other ones to increase its lifetime and autonomy.

Furthermore, fuzzy logic can assist ASE in self-configuration decisions of WSN, which can be confirmed by its adoption in computing routing metrics and MAC layer back-off delays [17], and to adapt message dissemination interval to loss rates combined with data transmission from a WSN node to an actuator in a wireless sensor and actuator network (WSAN).

We claim that an efficient context-aware WSN for Smart Cities must provide the contextualization task taking into account aspects focused on optimizing WSN resource utilization for improved sustainability, in order to achieve system availability and performance. Motivated by the performance issues of centralized context-aware WSN systems, associated with the severe performance demands of Smart City mission critical applications, this work proposes the distributed autonomic inference machine (DAIM). DAIM is an ASE embedded in each WSN node which uses fuzzy logic in contextualization task based on local and neighboring raw data, while it controls message dissemination and sensing intervals and the best times to start or stop dispatching messages by fuzzy rules.

The DAIM performance is evaluated in a real testbed and compared to ASE with crisp rules [15] and a distributed inference machine without self-management [13]. The results show that DAIM allows saving WSN energy consumption by 19.55%, decreases the number of sent messages in general by 29.87%, and decreases the number of context messages sent to the context provider by 48.83%. Context messages contain the result of contextualization done by WSN; therefore, if fewer context messages are sent to the context server, then there is less overload in servers.

This paper is structured as follows. In Section 2, we examine works about Context Aggregation and Autonomic System in WSN. Section 3 shows how DAIM functions are associated with the model of ASE with fuzzy rules for aggregating context information and self-management. Section 4 details an application used by DAIM that is responsible for the contextualization task based on temperature and relative humidity sensor of Crossbow [18]. In Section 5, there is a description of the testbed and other inference machines which have been implemented in nesC language to TinyOS 1.X and whose performance has been compared to the performance of DAIM. In Section 6, the obtained results are analysed and our acknowledgments are in Acknowledgments. Section 7 makes some concluding remarks and recommends further work in this area.

2. Related Work

We did not find in the literature related work addressing both the issues regarding the overload of centralized context providers nor WSN self-management. Thus, this section is subdivided to analyze the most relevant work related to contextualization and self-management applied in WSN.

2.1. Contextualization in WSN. In this section, 4 works exploiting fuzzy inference technique for contextualization in WSN are analyzed. A distributed fuzzy logic engine for rule-based WSNs (DFLER) is embedded in each WSN node to infer fire risk through fuzzy rules [13]. DFLERs' inputs are local temperature and smoke data and a neighboring consensual opinion about these two pieces of data. The neighboring consensual opinion is estimated through a majority quantifier calculated by WSN nodes with neighboring fuzzified data. Simulation results show that DFLER is more robust in cases with false positives and it has a lower error rate than an engine based on crisp rules or fuzzy rules with only local data as engine input.

WSN can recognize car assembler's activities through fuzzy logic with an algorithm of temporal sequence order embodied in each WSN node [19]. The algorithm inputs are the local and neighboring 3D accelerometer data, while the output represents an assembling activity. Upon contextualizing that assembly workers are not carrying out an operation correctly at a car assembly point, the WSN sends such context information to warn the car monitoring system.

Canada-Bago et al. [20] propose a fuzzy rule-based system composed of two fuzzy logic engines to process both local and neighboring data. Each WSN node embeds this system seeking to contextualize the presence of olive tree plagues based on processing temperature and humidity sensor data. In order to achieve this, the engine for local data is used first and its output represents a local alert status sent to its WSN neighbors nodes. The second engine receives, as its inputs, local and neighboring alerts status to compose a general alert status. This fuzzy system is incorporated in a distributed system which allow users to model the knowledge used in WSN nodes and transfer it to WSN [21].

Both works [13, 19] are not concerned about neither the WSN battery usage nor the context provider overload. Moreover, [20, 21] perform experiments testing the battery consumption to estimate WSN lifetime with their fuzzy system. The authors claim that their proposals allow for saving energy, as well as sending/processing lesser data than [13]; however they do not confirm these hypotheses in the works. Then, we implement the work of [13] and we put it to the same experiments/tests as our work to possibly evaluate their performance analyzing WSN lifetime and number of messages sent to the environment system.

2.2. WSN Self-Management. In this section, 3 works about self-management on WSNs are analyzed. Braga et al. [15] propose an ASE for WSN which is based on the generic model of Kephart and Chess [14] and it is embedded in each WSN node. ASE takes as inputs the local and neighboring information, such as intervals of sensing and message dissemination, and the amount of residual energy used in a correlation algorithm based on crisp rules (IF-THEN) to infer about the current performance of the WSN node. ASE executes an action plan to increase/decrease the intervals of sensing and message dissemination and to activate or not WSN nodes with redundant information. It was simulated in a WSN consisting of a hundred nodes and it saves approximately 40% of WSN energy when compared to a WSN without ASE. This work does not take into account the importance of contextualization to decrease the overload of inference process in context providers.

Cuevas-Martinez et al. [22] develop a management application protocol to a multiagent structure embedded in WSN nodes. There is a management agent that turns up and down sensor services and controls the sensor sleep-awake cycle. These sensor services take data measurements, infer a fuzzy system output, and so forth. An application control agent manages execution of fuzzy rule-based applications to control the interval of fuzzy system executions and the update of the knowledge base. The management application protocol establishes communication between the WSN node and its neighbors and with a management station. The system does not allow the WSN node to change itself, the executions, and sleep-awake intervals values of the application. The intervals can only be updated by the management station. The authors assure that the system can decrease the number of messages sent by WSN; however they do not examine such behavior in their experiments. The battery consumption is evaluated but it is not compared with other systems results.

The same authors of [22] adapt the management application protocol so that WSN node can infer itself the best sleep mode interval (SMI) using a fuzzy system [23]. The system takes as inputs sound pressure value (their use case), the variation of sound pressure, and battery level. This work is compared with a differential system which calculates the next SMI based on the difference of data measured and its previous value and the battery level [24]. Experiments in a real WSN show there is no significant difference in energy consumption between the systems.

Our analysis in relevant work on WSN contextualization and self-management evidences in which none of the considered proposals fulfill the requirements pointed out in Section 1, in terms of taking into account available context for optimizing WSN resource utilization to meet improved sustainability and performance (see Table 1). In our proposal, the fuzzy logic is adopted to allow context-driven WSN self-managing its resources without human intervention as much as possible, while it does contextualization over classic solutions based on sensor raw data (Table 1). In order to compare another work performance towards contextualization, we implement ASE [15] and DFLER [13] besides our proposal.

3. DAIM Description

Before we describe how the proposal works, we make some considerations. The WSN topology is flat; that is, all sensor nodes are equal in their roles that they perform and send messages to a sink node connected to a context provider.

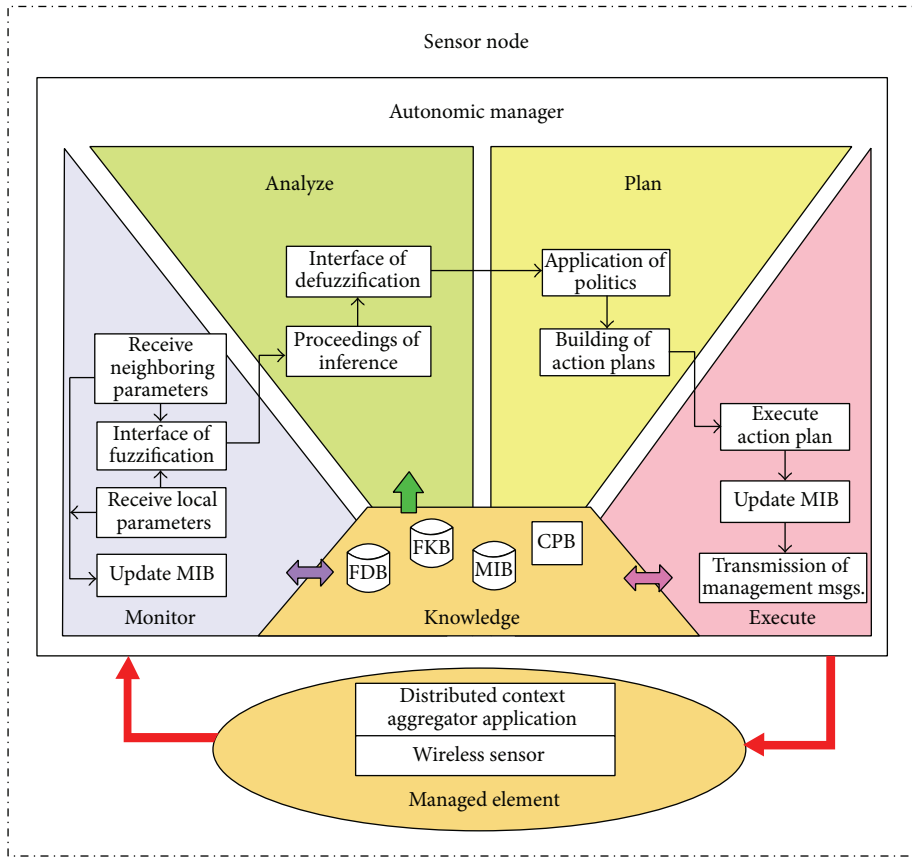

Figure 1: Distributed autonomic inference machine (DAIM).

Table 1: Main characteristics of related works.

Work	Contextualization method	WSN self-management method
Marin-Perianu et al. [13]	Fuzzy logic/distributed	None
Marin-Perianu et al. [19]	Fuzzy logic/distributed	None
Canada-Bago et al. [20, 21]	Fuzzy logic/distributed	None
Braga et al. [15]	None	ASE with an correlation algorithm based on crisp rules
Cuevas-Martinez et al. [22]	Fuzzy logic/distributed	None
Cuevas-Martinez et al. [23]	Fuzzy logic/local	Fuzzy logic
Our solution	Fuzzy logic/distributed	ASE with an correlation algorithm based on fuzzy rules

The context provider stores and processes sensor data delivering such data to clients applications through public interfaces representing synchronous queries or subscriptions.

Each sensor node embeds a DAIM instance in order to enable both self-management and contextualization operations. These operations are based on the fuzzy logic in order to achieve increased WSN's lifetime while optimizing the performance rates on the context provider in terms of processing overhead. Figure 1 shows DAIM architecture, highlighting its main parts, namely, the managed element (ME) and the autonomic manager (AM).

The ME consists of a distributed context aggregator application similar to a distributed fuzzy logic engine for rule-based WSN (DFLER) [13]. The context generated in DAIM is a high-level value-added information with potentials to semantically describe changes in the targeted environment, such as fire risk (more details in Section 4).

The following subsections describe the AM which is responsible for the self-configuration of sensor nodes. We start explaining the AM general overview in Section 3.1 and its components' details and interactions in Sections 3.2, 3.3, 3.4, and 3.5.

3.1. Autonomic Manager Overview. The AM is designed following the MAPE-K model [14], thus encompassing the well-defined services of Autonomic Computing, namely, Monitor Service, Analyze Service, Plan Service, and Execute Service, which query the Knowledge during the autonomic cycle. The autonomic cycle is the repeated execution of MAPE-K engine

in a time interval previously defined (we configure this time value in Section 5.2).

In our proposal, we use a distributed fuzzy system to infer appropriate intervals for both data sensing and message dissemination as well as to determine if such information is redundant for context processing in the context provider. Figure 1 shows where the basic components of a fuzzy control system are placed in a MAPE-K engine. Monitor Service has an interface of fuzzification. Analyze Service has proceedings of inference and an interface of defuzzification. Knowledge has a fuzzy data base (FDB) and a fuzzy knowledge base (FKB).

At the beginning of autonomic cycle, the Monitor Service (see Figure 1) provides the self-knowledge feature which depends on monitoring software and hardware parameters (e.g., sensor node configuration). The Monitor Service makes the sensor node learn its own local configuration and its sensor neighbors' one (through the management messages). In this way, DAIM allows the sensor node to be aware of internal functions and behavior, as well as the environment where it is placed (read Section 3.2 for more details).

The Analyze Service (described in Section 3.3) processes data forwarded by the Monitor Service to make the WSN self-diagnosis for the sensor node analysis and decision based on its Knowledge, such as increasing or not the value of sensing interval. The self-diagnosis is useful information (context) which represents the node's current state of sensing and message dissemination interval and the level of redundant information collected by WSN. Thus, DAIM grants WSN the context-awareness.

The Plan Service (described in Section 3.4) processes the self-diagnosis to decide whether it is necessary to adapt the system to attend the goals defined in its Knowledge. The Plan Service builds an action plan which is a set of adaptation values aiming at the performance optimization of the sensor node. Therefore, DAIM triggers context-driven actions to optimize WSN resources.

The Execute Service (described in Section 3.5) implements the desired self-configuration feature. This service uses the action plan forwarded by the Plan Service to increase/decrease the sensor node's sensing and message dissemination intervals. The Execute Service can even stop sending messages to the sink node aiming to reduce the problem of overload in context provider. This is the end of autonomic cycle.

The Knowledge keeps the management information base (MIB) which stores the current values of

(i) sensor node and its neighbors configuration (see Section 3.2);

(ii) maximum and the minimum thresholds of sensing and message dissemination intervals;

(iii) self-management interval, defined as a period of time in which the sensor node sends management messages to its neighbors;

(iv) autonomic cycle interval.

We set the value of these intervals in Section 5.2. Knowledge also keeps a communication protocol base (CPB) which has the types and formats of messages sent by DAIM: management (Section 3.2) and fuzzy and context (Section 4) messages. We describe with more details each autonomic service in the following subsections.

3.2. Monitor Service. As illustrated in Figure 1, the Monitor Service retrieves the sensor node configuration from MIB of Knowledge. The sensor node configuration is a set of parameters which describes local information of the sensor node. These parameters used are the percentage of relevant data, the values of sensing and message dissemination intervals, the number of dispatched message bytes, and the percentage of residual energy. Relevant data are raw data values which indicate important changes in environment, so they depend on type of sensors used (we show its values in Section 5).

The Monitor Service also retrieves management messages from MIB. A management message has a neighbor configuration and its last raw sensed data. The neighbor configuration is the sensor node configuration of a sensor node at a hop distance from a given node (neighbor). Management messages are sent by the Execute Service at the end of autonomic cycle and periodically in each self-management interval (see Section 3.1).

Then, the Monitor Service computes the neighboring percentage of relevant data and the percentage of its neighbors with similar parameters values (sensor node configuration). A neighbor parameter value is similar to a local parameter value when 60% of the information received through management messages is different in 20% of its own available information. This is a convention used by [15].

At the beginning of the autonomic cycle, the interface of fuzzification receives, as input, the local and neighboring parameters stored in the MIB (Figure 1). The interface of fuzzification transforms an input value in instances of its respective linguistic variable.

A linguistic variable has a universe of discourse U which contains all possible input numeric values of the interface of fuzzification and grammar G which defines a set of linguistics terms (fuzzy set). Each linguistic term is represented by a pertinence/membership function that receives the input numeric value and returns a pertinence degree.

The pertinence degree is a number between 0 and 1 and represents how much the input numeric value is linked to that linguistic term of the linguistic variable [25, 26]. Pertinence/membership functions are placed at FDB of Knowledge component.

In this way, self-configuration fuzzification process uses the following linguistic variables:

(i) Percentage of relevant data and percentage of residual energy are defined by $G1 = \{s0 = \text{Low}, s1 = \text{Medium}, s2 = \text{High}\}$, $U1 = [0, 100]$, and the membership functions of the fuzzy set of Figure 2(a).

(ii) Number of dispatched message bytes is defined by $G1$, $U2 = [0, 20000]$, and the membership functions of the fuzzy set of Figure 2(b).

(iii) Sensing interval is defined by $G1$, $U3 = [0, 120]$, and the membership functions of the fuzzy set of Figure 3(a).

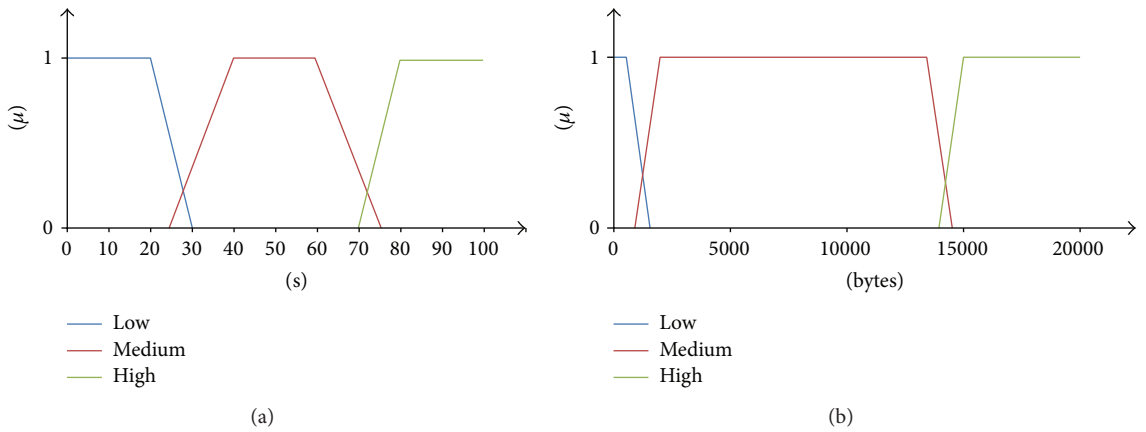

Figure 2: (a) Fuzzy set of percentage of relevant data and (b) fuzzy set of number of dispatched messages bytes.

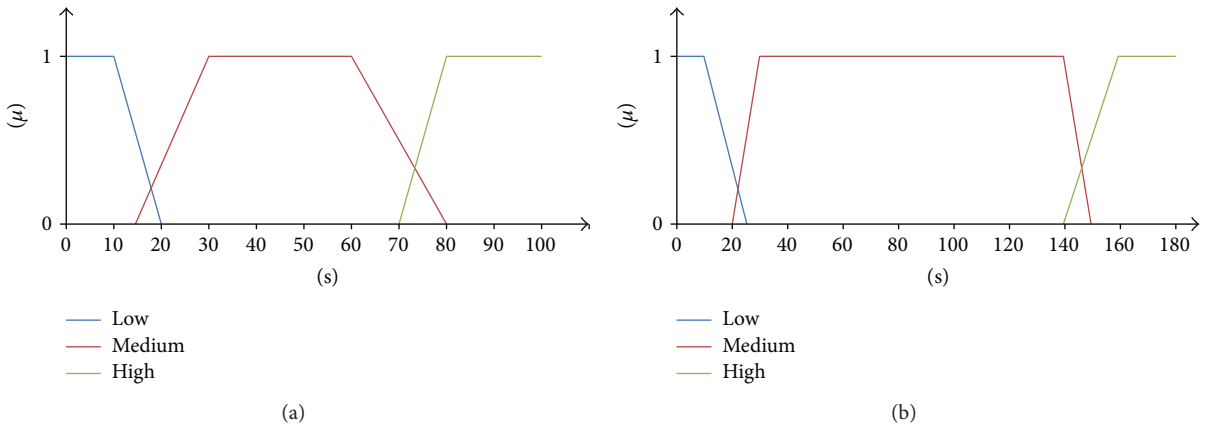

Figure 3: (a) Fuzzy set of sensing interval and (b) fuzzy set of message dissemination interval.

(iv) Message dissemination interval is defined by $G1$, $U4 = [0, 180]$, and the membership functions of the fuzzy set of Figure 3(b).

(v) Percentage of neighbors with similar configuration is defined by $G1$, $U1$, and the membership functions of the fuzzy set of Figure 4.

Trapezoidal functions are more suited to the processing restrictions of sensor nodes because they have lower computational cost than sine functions and they are easy to implement.

3.3. Analyze Service. During autonomic cycle, the Monitor Service sends the fuzzified parameters to the Analyze Service. These parameters are then applied in 74 fuzzy rules of FKB (Knowledge). In the inference process, the active rules determine fuzzy values of the WSN self-diagnosis. The self-diagnosis is represented by the current state of sensor node sensing and message dissemination intervals and WSN information redundancy level.

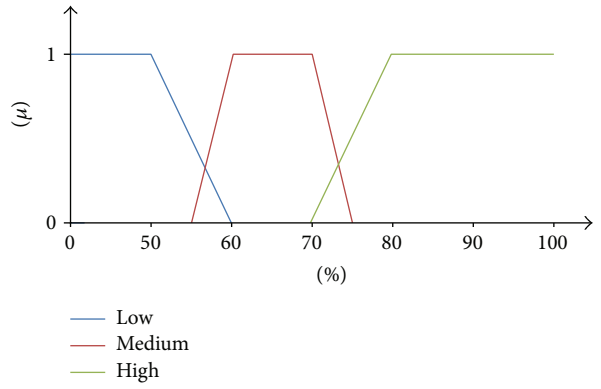

Figure 4: Fuzzy set of percentage of neighbors with similar configuration.

The following fuzzy rules are examples of WSN self-diagnosis:

(i) If the percentage of the relevant local data is Low AND the relevant neighboring data is Low AND the sensing interval is High AND the dissemination interval is Medium, THEN the current state of sensing interval is

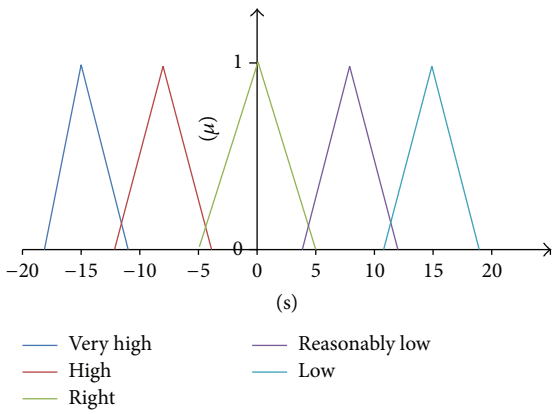

Figure 5: Fuzzy set of sensing interval self-diagnosis.

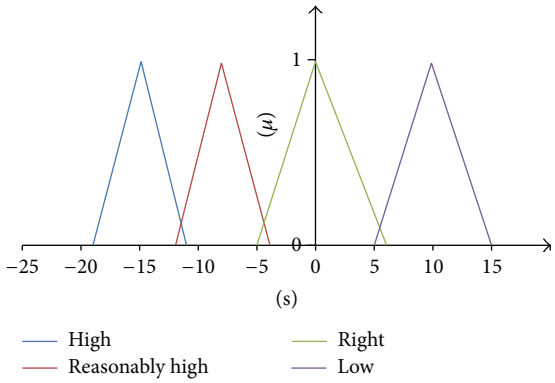

Figure 6: Fuzzy set of message dissemination interval self-diagnosis.

Low AND the current state of message dissemination interval is Low.

(ii) If the percentage of neighbors with a similar configuration is High AND the percentage of relevant data is Low, THEN the information redundancy level is High.

All the rules use "AND" operator to carry out the intersection (min operation) between the fuzzy sets of linguistic variables inputs. The inference model is Mamdani where the rules are activated whenever the result of "AND" operation is higher than zero. Several rules can be activated during the inference process and each one makes a contribution to express the current state of intervals and information redundancy level (WSN self-diagnosis). Rules contributions are combined with the union operator by using the "MAX" operation [25, 26].

The interface of defuzzification receives the inference fuzzy result. It uses the Centroid method to transform the result into nonfuzzy value to be used in the self-configuration actions. For intervals, the defuzzification result is the number of seconds to be added to or subtracted from the current value of interval (Figures 5 and 6). When a defuzzification value of intervals is positive, it represents "Add seconds" action and when it is negative, it represents "Subtract seconds." In the case of information redundancy, the defuzzification result is the level of how much sensor configuration and irrelevant data are similar to neighbouring.

In this way, the following linguistic variables are used by defuzzification process of WSN self-diagnosis:

(i) Sensing interval self-diagnosis is defined by $G2 = \{s0 = \text{Very High}, s1 = \text{High}, s2 = \text{Right}, s3 = \text{Reasonably Low}, s4 = \text{Low}\}$, $U5 = [-18, 19]$, and the membership functions of the fuzzy set of Figure 5.

(ii) Message dissemination interval self-diagnosis is defined by $G3 = \{s0 = \text{High}, s1 = \text{Reasonably High}, s2 = \text{Right}, s3 = \text{Low}\}$, $U6 = [-19, 15]$, and the membership functions of the fuzzy set of Figure 6.

(iii) Information redundancy level self-diagnosis is defined by $G1$, $U1$, and the membership functions of the fuzzy set of Figure 4.

3.4. Plan Service. The Plan Service uses the defuzzification result of self-diagnosis in the policies regarding the execution of sensor node self-configuration. The policies give the probabilities of increasing or decreasing intervals and allow the sensor node whether or not to stop dispatching messages to the context provider. The probabilities are included to prevent all the sensor nodes of the same region abruptly reducing their sensing and/or dissemination rates which could let the region without WSN monitoring. According to [15], they also prevent the rates rising at the same time; thus they are able to reduce collisions and the loss of messages.

Following are the policies of the Plan Service:

(1) Add 80% of probability to execute an addition of 12 to 19 seconds in the current value of sensing interval.

(2) Add 20% of probability to execute an addition of 4 to 11 seconds in the current value of sensing interval.

(3) Add 80% of probability to execute a subtraction of 4 to 11 seconds of the current value of sensing interval.

(4) Execute immediately a subtraction of 12 to 18 seconds of the current value of sensing interval.

(5) Add 80% of probability to execute an addition of 5 to 15 seconds in the current value of message dissemination interval.

(6) Add 80% of probability to execute a subtraction of 12 to 19 seconds of the current value of message dissemination interval.

(7) Add 20% of probability to execute a subtraction of 4 to 11 seconds of the current value of message dissemination interval.

(8) Add 40% of probability to stop dispatching messages to the context provider when the information redundancy level is between 55% and 75%.

(9) Add 60% of probability to stop dispatching messages to the context provider when the information redundancy level is higher than 75%.

If the number of seconds to increase/decrease the intervals and/or information redundancy level (defuzzification

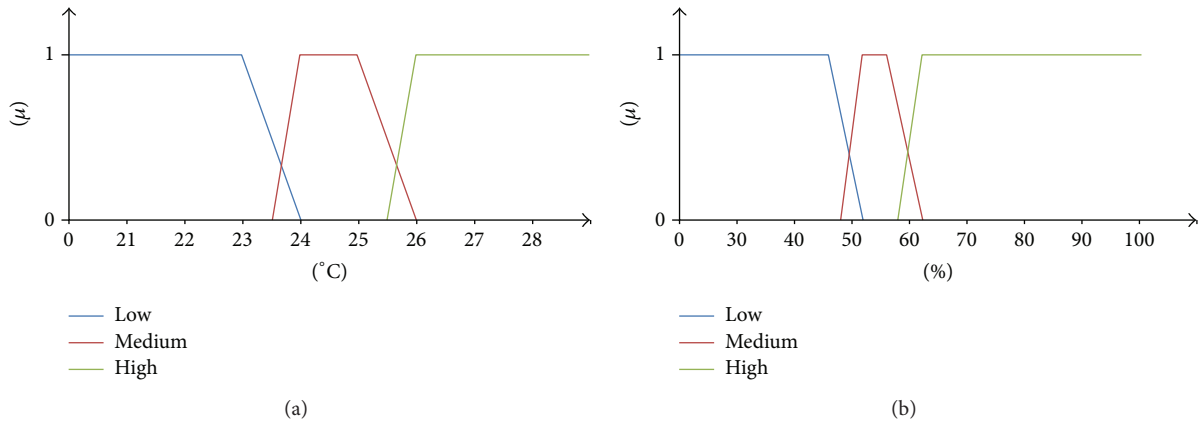

Figure 7: (a) Fuzzy set of temperature and (b) fuzzy set of relative humidity.

results of the WSN self-diagnosis) are not included in the policies, the self-configuration activities will not occur. Consequently, the values of the sensing and message dissemination intervals will not be changed and the sensor node will not stop sending messages to the context provider, but if it has already stopped, it will stay stopped.

Following are some examples to demonstrate how an action plan is built relating the defuzzification values of WSN self-diagnosis to a policy applied to the intervals and information redundancy level:

(i) If the defuzzification value for the sensing interval is "+13" then policy 1 will be applied; that is, 13 seconds will be added with 80% of probability to the value of the sensing interval.

(ii) If the defuzzification value for the dissemination interval is "−7" then policy 7 will be applied; that is, 7 seconds will be subtracted with 20% of probability of the value of dissemination interval.

(iii) If the defuzzification value for the information redundancy level is "50" then no policy will be applied; that is, the sensor node will not be stopped from sending messages to context provider.

3.5. Execute Service. The action plan is executed by this service during the autonomic cycle. The values of the sensing and message dissemination intervals are only updated if a new value does not exceed the minimum and maximum limits in MIB. The Execute Service sends management messages with the sensor node configuration and its last raw sensed data. This is the end of autonomic cycle.

4. DAIM Prototyping Evaluation

The contextualization done by WSN takes the raw data of different sensors embedded in a WSN node to generate a set of context information, such as changes in monitored environment. For the implementation of this work we used Crossbow MTS400 sensorboards [18] embodying temperature, relative humidity, light, pressure, and accelerometer sensors. The contextualization task follows environment specialized knowledge to process raw data for context generation. Thus, we consider a knowledge base in which variables have the same type of the available sensors, in order to make possible the classification of the environmental conditions with two or more types of available sensors.

As we have not found such required knowledge base, we deployed fire risk information based on the combination of temperature and relative humidity raw data, with the contribution from a meteorologist of the Ceará Foundation of Meteorology and Water Resources (FUNCEME) (http://www.funceme.br). He said that it could be possible to compose fire risk information using temperature and relative humidity data. After this, several experiments involving environment temperature and relative humidity changes were carried out to model a fuzzy knowledge base of fire risk with 36 rules.

First of all, in each sensing interval, the temperature and relative humidity raw data are taken as the input of a distributed context aggregator application similar to DFLER [13]. Then, the inputs are fuzzified using the fuzzy sets of Figures 7(a) and 7(b), respectively. Secondly, the fuzzified data are sent to the neighbors' nodes to calculate the most quantifier defined by the fuzzy sets in Figure 8(a). The most quantifier is the consensual opinion about the neighboring data.

In this way, the following linguistic variables are used by the fuzzification on the contextualization task:

(i) Temperature is defined by $G1, U7 = [15, 100]$, and the membership functions of the fuzzy set of Figure 7(a).

(ii) Relative humidity is defined by $G1$, $U8 = [20, 100]$, and the membership functions of the fuzzy set of Figure 7(b).

(iii) Most quantifier is defined by $G4 = \{s0 = \text{Low}, s1 = \text{High}\}$, $U9 = [0, 1]$, and the membership functions of the fuzzy set of Figure 8(a).

The fire risk inference utilizes the local fuzzified data and the most quantifier in fuzzy rules which model this problem. For instance, rule 20 says: "IF the local temperature is High

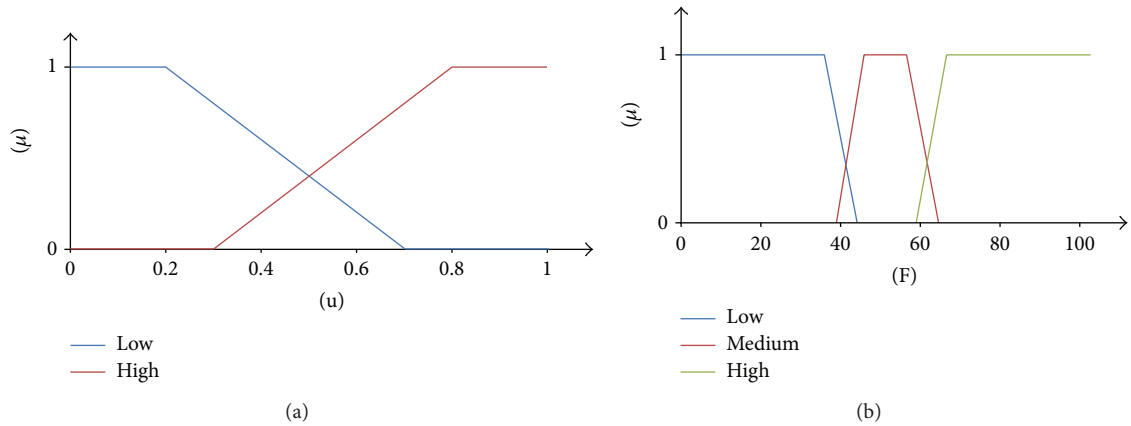

Figure 8: (a) Most functions and (b) fuzzy set of fire risk.

AND the local relative humidity is Low AND temperature of the most quantifier is High AND the relative humidity of the most quantifier is High, THEN the fire risk is High." The operator "AND" and Mamdami method are used to activate the fire rules of inference fire risk. The defuzzification method is Centroid to detect fire risk levels. The linguistic variable Fire Risk is defined by $G1, U1$, and the membership functions of Figure 8(b).

The Micro Sensornet (μSONG) ontology proposed by Hauifeng and Xingshe [27] defines the context messages of fire risk dispatched to the context provider in each message dissemination interval. μSONG expresses the WSN context as having only 3 principal attributes:

(i) *ContextName*: this is the particular name of the context and, in this work, it is FIRE_RISK.

(ii) *ContextValue* is used with *ContextName* and, in this work, it is VERY_HIGH.

(iii) *Fidelity* shows the fidelity of the *ContextValue* and, in this work, it is the defuzzification result; according to Figure 8(b), it can be a value between 0 and 100.

5. DAIM Prototyping Evaluation

This section describes the materials and methods used to evaluate if the self-management function can really help save WSN resources and if the WSN contextualization function can reduce the processing overload of the context provider. We have implemented two related work systems. They were embedded in WSN nodes and tests were carried out to estimate WSN storage, energy, and communication as well as the overload of the context provider.

5.1. Description of Testbed. The WSN of our experiments is composed by 6 MICAz motes with MTS400 sensorboards and a sink node, that is, a MICAz mote with a sniffer application embedded which is connected to a MIB520 [18] that forwards the WSN messages to a notebook by USB connectivity. The context provider is a Java SerialForwarder application running in the notebook to receive and store in a data base the WSN messages. Figure 9 shows the testbed.

Figure 9: Testbed.

Each system is embedded in the WSN nodes to run for 1 hour. With the purpose of analyzing the system reaction during a possible fire risk, the WSN is under action of a hair dryer of 1700 W for 10 minutes. In this testbed, the hot air event starts after the first 100 seconds of system execution in the WSN nodes. The hot air event modifies both environment temperature and relative humidity of a room which is cooled at 18°C. This proceeding is repeated 10 times to each system. Each WSN node uses two AA alkaline batteries which are replaced when another system is evaluated.

Following are the parameters analyzed to evaluate the systems:

(1) ROM memory consumed.

(2) RAM memory consumed.

(3) Percentage of consumed energy by WSN.

(4) Number of dispatched messages.

(5) Number of lost messages.

(6) Number of data messages dispatched.

(7) Number of relevant data messages dispatched.

(8) Number of context messages dispatched.

(9) Number of relevant context messages dispatched.

(10) Number of messages bytes dispatched to the context provider.

(11) Mean of sensing and message dissemination intervals.

(12) Mean of last value of sensing and message dissemination intervals.

(13) Mean of variance coefficient of sensing and message dissemination intervals.

WNS storage resource is measured by items (1) and (2). WSN lifetime is determined by item (3). WSN communication is deduced from items (4) to (13). The overload of context provider is analyzed through items (8) and (9).

Health messages are sent by all systems every minute with information about number of dispatched messages, number of lost messages, and residual energy of a WSN node. Residual energy is given by the batteries voltage value of a WSN node.

Health messages and their functions inserted in systems code are not included in calculation of neither parameters (1), (2) nor parameters (4) to (10) mentioned before, because health messages help only in proceedings to evaluate the individual performance of each system implemented.

In this work, temperature data is considered relevant when it is equal to or higher than 30°C and relative humidity data is relevant when it is equal to or lower than 40%, since both indicate high heating in the environment. Context of a very high fire risk is relevant when *Fidelity* value is equal to or higher than 70 (see Section 4).

5.2. Description of Tested Systems. The systems are implemented in nesC language and run under TinyOS Operating System (http://www.tinyos.net), version 1.1. NesC is an extension of C Language which incorporates the execution model of TinyOS, an event-based operating system designed for WSNs. TinyOS provides a set of software components, including components that implement the communication protocol stack for WSNs, which were used in our implementation. No routing protocol was used, but, at link layer, Zigbee protocol (IEEE 802.15.4) was used by WSN nodes.

The first system evaluated is ASE which is described and simulated in NS-2 in Braga et al. [15] with a carbon monoxide (CO) sensing application as managed element. The ASE autonomic manager is based on a MAPE autonomic engine with crisp rules to provide self-configuration of WSN nodes. The managed element is adapted to the sensorboard available for this work and it consists of a sensing application of temperature and relative humidity that dispatches messages with local raw data to the context provider.

The second system tested is DFLER which is a distributed sensing application simulated in OMNet++ Marin-Perianu et al. [13]. DFLER uses temperature and smoke raw data and fuzzy logic to infer fire risk. In our work, DFLER is adapted to sensorboard available to contextualize fire risk based on temperature and relative humidity raw data available. The application is similar to the managed element described in Section 4. Its sensing and message dissemination intervals are both fixed in 40 seconds.

The third system tested is DAIM described in Section 3. It contextualizes fire risk using local and neighboring temperature and relative humidity while it makes the WSN node self-configuring intervals for both sensing and message dissemination. It decreases redundant context information sent to context provider. Therefore, DAIM uses fuzzy rules to both functions: environmental conditions inference and WSN self-management.

Table 2: Messages types.

Type	Length (bytes)	System
Data	10	ASE
Management	22	ASE, DAIM
Fuzzy	42	DFLER, DAIM
Context	8	DFLER, DAIM

Table 3: Mean of analysed parameters.

Parameter	ASE	DFLER	DAIM	DAIMsec
Consumed ROM memory (bytes)	23870	54140	65716	74692
Consumed RAM memory (bytes)	1395	1268	1912	2108
Consumed energy (%)	1.91	2.30	1.85	2.04
Total of sent messages	499.8	2520.7	1767.6	1034.5
Total of lost messages	0	0	0	0
Total of data messages	69.8	—	—	—
Total of relevant data messages	37.4	—	—	—
Total of context messages	—	480.4	245.8	204.7
Total of relevant context messages	—	91.1	70	43.1

The fourth system tested is DAIMsec which is DAIM with authentication resources of TinySec framework [28]. It uses the MICAz version code of Stajano et al. [29]. DAIMsec uses the component *TinySecC* which is responsible for commands of receiving and sending messages with authentication. Security is an important feature to Internet of Things systems because they can be attacked in many ways, like pushing erroneous data into the WSN. Authentication resources ensure data integrity and authenticity [30], so we choose to evaluate our work with this function.

Table 2 shows types of messages dispatched by the 4 implemented systems for TinyOS. Sensing and message dissemination intervals start at 20 and 40 seconds, respectively, for both ASE and DAIM. The minimum and maximum threshold are 10 and 120 seconds, respectively, for the sensing interval. For message dissemination interval, the minimum and maximum threshold are, respectively, 15 and 180 seconds. For DFLER, these intervals are fixed in 40 seconds. ASE and DAIM have an autonomic cycle which is done by their autonomic manager and it runs every 100 seconds and the management messages are also sent every 200 seconds (self-management interval).

6. Result Analysis

In this section, we show the results obtained in experiments of system evaluation according to the analysis of the 13 parameters listed in Section 5.1. Table 3 shows the mean of

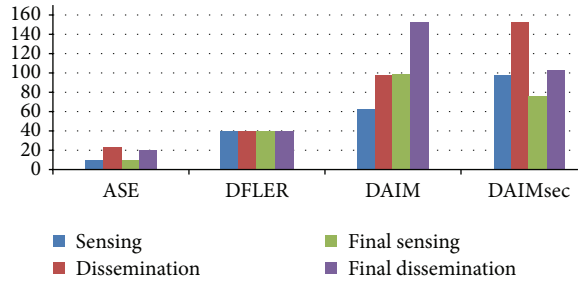

Figure 10: Means of sensing and message dissemination intervals.

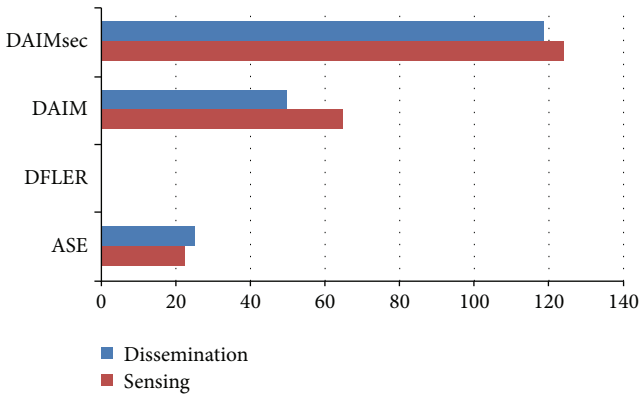

Figure 11: Covariance of sensing and message dissemination intervals.

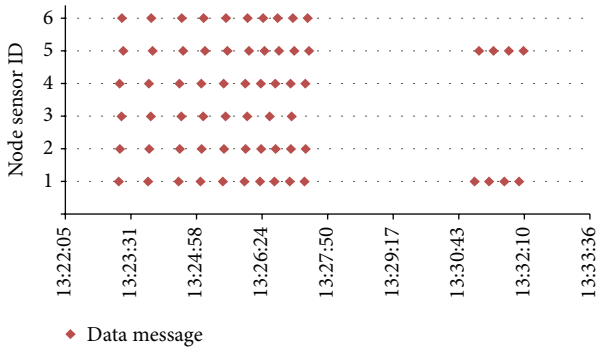

Figure 12: Data messages dispatched by ASE to the context provider.

the nine first parameters analyzed and the other ones are illustrated in figures of this section.

6.1. WSN Storage Resources. We notice DAIM consumes more sensor storage resources than the other systems. DAIM occupies approximately 50.13% of a 128 KB ROM and 46.68% of a 4 KB RAM of a MICAz mote. DAIMsec code consumes 57% of ROM and 46.7% of RAM of a sensor. It is expected because DAIM does both self-management and contextualization tasks. The free memory remaining is enough to employ a routing protocol with DAIM.

6.2. WSN Lifetime. DAIM spends less WSN energy than the other systems: it saves approximately 3.24% more energy than ASE and 19.55% more than DFLER. DAIMsec wastes 9.31% more energy than DAIM, but it saves 11.3% more than DFLER which represents a good match between DAIM and TinySec framework. DAIM self-management fulfills WSN optimization resource demands than other systems evaluated.

6.3. WSN Communication. Looking at the number of sent messages (see Table 3), DFLER sends more messages than other systems and ASE is the one which sends less. In general, DAIM sends 3.5 more messages than ASE, but it dispatches 29.87% less messages than DFLER.

Figure 10 shows the mean of sensing and message dissemination intervals and the mean of their last values (parameters 11 and 12, see Section 5.1) which help us analyze the self-configuration function of systems. The intervals of DFLER are constants because it does not change them through self-configuration capabilities.

ASE and DAIM change the intervals because they have an autonomic manager. However, DAIM modifies them more than ASE as Figure 10 shows us. In fact, the means of DAIM sensing and message dissemination intervals are, respectively, 5.84 and 4.10 higher than the mean of ASE intervals. The means of DAIM intervals last values are, respectively, 9.85 and 7.55 higher than the ASE ones. The mean of ASE sensing and message dissemination intervals and its last values are little more higher than the half of its initial values, 20 and 40 seconds, respectively.

Figure 11 shows the results of self-configuration through the mean of variance coefficient of sensing and message dissemination intervals. DFLER does not self-configure its intervals; then, its variance coefficient of intervals is zero. DAIM variance coefficient is higher than the ASE one.

The means of the intervals indicate that, at the end of execution, ASE could be collecting and sending data messages without relevant data, that is, without data of significant changes of environment temperature and relative humidity (hot air event). It means that autonomic manager with crisp rules does not work well.

To confirm this hypothesis, we analyse an ASE traffic pattern of data messages. Figure 12 shows data messages dispatched by WSN node with ASE to context provider during a round test between 13 h22 and 14 h22. We observe that ASE decreases message dissemination interval, so it augments the dispatching rate of data messages during hot air event. However, ASE makes WSN nodes stop sending data messages before the event finishes.

WSN with ASE stops monitoring the area where it is placed and, consequently, it causes starvation of data messages in context provider system which can not inform any more its users about the current environment conditions. Therefore, ASE is the system which sends less messages in general.

TinyOS sensing components continue to be called every 10 seconds (the minimum threshold of sensing interval) until the final of test. Then, we conclude that ASE spends the major part of WSN energy in this activity.

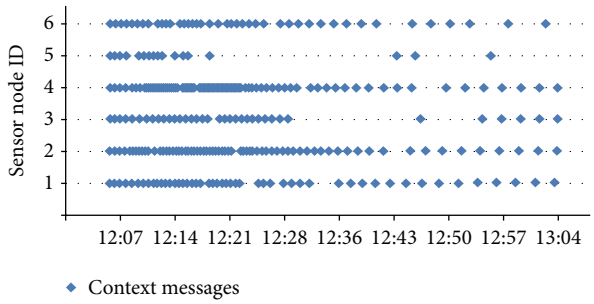

Figure 13: Context messages dispatched by DAIM to the context provider.

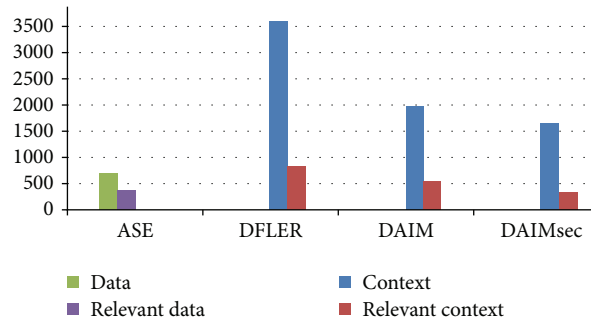

Figure 14: Mean of messages bytes dispatched by system to the context provider.

We analyze DAIM traffic data pattern as well. Figure 13 shows context messages dispatched to the context provider by each WSN node with DAIM during a round test between 12 h05 and 13 h05. DAIM sends several context messages at the beginning of execution which is a reaction due to hot air event.

When the event finishes and the environment is cooled continuously, WSN node 5 stops dispatching context messages for approximately 40 minutes. WSN node 3 stops for 30 minutes, but it starts dispatching context messages again with a large message dissemination interval. It happens to save energy because there is no more fire risk and the information collected by these WSN nodes is redundant (i.e., very similar to the other WSN nodes).

The results demonstrate that DAIM autonomic manager augments sensing and message dissemination rates during hot air event and it decreases these rates when the event disappears. DAIM autonomic manager can stop message dissemination when configuration of WSN nodes is very similar to its neighbours and when there is no relevant information to a very high fire risk. Moreover, it saves more WSN node energy than the other and it decreases the number of messages sent by WSN without causing starvation of messages in the context provider.

6.4. Overload of Context Provider. Context messages have the state of environment which is deduced from two or more types of data collected by WSN. In our use case, DAIM and DFLER aggregate local and neighbouring temperature and relative humidity data using a fuzzy system. So, these systems make WSN work as distributed system to do the contextualization task.

ASE does not do contextualization and it sends data messages with two pieces or more of collected information. Therefore, the context provider can be overloaded because it has to aggregate WSN data by itself, according to [9, 10].

In our tests, DAIM sends 6.57 more relevant messages (i.e., which alert about fire risk) than ASE (see Table 3). This result is expected because ASE causes starvation of data to the context provider, so the number of data messages sent is reduced.

DAIM sends 48.83% less context messages and 23.15% less relevant context messages than DFLER. DAIMsec sends 16.72% less context messages than DAIM and 38.42% less relevant context messages, but there is not starvation.

It is a desirable result of autonomic self-management. DAIM autonomic manager decreases sensing and message dissemination intervals at the moment of hot air event and it augments them after event finishes and/or local data collected is very similar to other WSN neighbour nodes.

This DAIM functionality reduces considerably the number of bytes messages sent by WSN (Figure 14) and consequently it saves WSN energy. In addition, the context provider detects the current conditions of the environment with less context messages. Context messages are smaller than data messages and they have a high semantic value because fire risk context is more meaningful than temperature and relative humidity raw data.

7. Conclusion

Autonomic Computing paradigm and context-awareness (with Artificial Intelligence) techniques can really help WSN to self-manage its resources more efficiently in order to increase its operational lifetime. During context generation task (i.e., contextualization), a context provider is overloaded, because it has to correlate various types of raw sensor data received through WSN messages towards providing context (a value-added meaningful information) semantically indicating relevant changes in the surrounding environment. We show that Autonomic Computing and fuzzy logic allowed optimized overload rates in the context provider introduced by the contextualization task.

With an autonomic element based in a fuzzy logic system, DAIM is embedded in each WSN node to manage sensing and message dissemination rates according to the environment conditions. If DAIM detects relevant changes in the monitored environment, it increases the rates to send information messages to the context provider. If there is no relevant change and the information collected by the WSN node is very similar to that collected by its neighbors, then DAIM decreases the rates and/or, occasionally, it makes the WSN node stop sending information messages.

Our experiments using MICAz motes demonstrate that DAIM adoption can save up to 19.55% of WSN energy consumption because, in general, it sends 29.87% less messages even though WSN sends 6.57 more relevant information messages. DAIMsec, DAIM version with the TinySec authentication resources, can save 11.3% of WSN energy showing

that DAIM can be used with security communication components. It is important to mention that DAIM does not cause the problem of starvation of messages in the context provider as it was reported in ASE [15].

DAIM context messages contain the context/condition of the environment. DAIM makes the WSN work as a distributed fuzzy system which takes local and neighboring raw data of two or more types of sensors in a process to compose relevant context. We believe that bringing the contextualization task within the WSN can reduce the overload that the context provider would suffer if it has to compose the environment context itself correlating all WSN data messages. The self-management feature also decreases the overload because DAIM sends 48.83% less messages to the context provider process than DFLER [13]. A context message has also the advantage of being smaller than data message with two types of raw data, therefore promoting additional energy saving fewer bytes since fewer bytes need to be transmitted.

Although DAIM occupies more WSN node memory than other evaluated related work, there is free space enough to install other services like a routing protocol. The limitation of this work is the testbed because we used only 6 MICAz placed in a closed room which does not represent very well a real place like a Smart City illustrated by Smart Santander [3].

As future work, DAIM and other systems implemented in nesC language are going to be simulated in TOSSIM (http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM), a simulator for TinyOS applications, in order to assess our approach in a large-scale WSN, composed of hundreds of nodes. Another future work is including DAIM in a semantic clusterization model [31] where WSN saves energy using fuzzy logic to aggregate similar data and put sensor nodes in low duty cycle reducing messages transmissions. We believe that WSN can detect environment changes more faster than a centralized context provider, so it will be evaluated in another future work.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

Danielo G. Gomes gratefully thanks the financial support from CNPq (Project no. 480872/2012-0). Flávia C. Delicato is partially supported by CNPq (under Grants 307378/2014-4 and 457783/2014-1) and FAPERJ (Grant JCNE E-26/102.961/2012). Luci Pirmez is partially supported by CNPq (under Grants 304941/2012-3, 473851/2012-1, and 477223/2012-5) and INMETRO (PRONAMETRO).

References

[1] A. Caragliu, C. del Bo, and P. Nijkamp, *Smart Cities in Europe*, Serie Research Memoranda 0048, VU University Amsterdam, 2009.

[2] J. C. Augusto, "Ambient intelligence: the confluence of ubiquitous/pervasive computing and artificial intelligence," in *Intelligent Computing Everywhere*, pp. 213–234, Springer, London, UK, 2007.

[3] L. Sanchez, L. Muñoz, J. A. Galache et al., "SmartSantander: IoT experimentation over a smart city testbed," *Computer Networks*, vol. 61, pp. 217–238, 2014.

[4] A. K. Dey, "Understanding and using context," *Personal and Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, 2001.

[5] L. Sokol and R. Ames, *Analytics in a Big Data Environment*, IMB, 2013.

[6] A. Török, P. Laborczi, and B. Mezny, "Context-aware traffic information flooding in vehicular ad hoc networks," *Tamkang Journal of Science and Engineering*, vol. 13, no. 1, pp. 53–61, 2010.

[7] A. Copetti, J. C. B. Leite, O. Loques, and M. F. Neves, "A decision-making mechanism for context inference in pervasive healthcare environments," *Decision Support Systems*, vol. 55, no. 2, pp. 528–537, 2013.

[8] D. Merico, A. Mileo, and R. Bisiani, "Wireless sensor networks supporting context-aware reasoning in assisted living," in *Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments (PETRA '08)*, pp. 1–2, ACM, Arlington, Va, USA, July 2008.

[9] H. Kawashima, Y. Hirota, S. Satake, and M. Imai, "MeT: a real world oriented metadata management system for semantic sensor networks," in *Proceedings of the 3rd workshop on Data management for sensor networks (DMSN '06)*, pp. 13–18, ACM, September 2006.

[10] N. Jabeur, J. D. McCarthy, X. Xing, and P. A. Graniero, *A knowledge-oriented meta-framework for integrating sensor network infrastructures [Master's thesis]*, 2009.

[11] J. Antoniou, C. Christophorou, A. Neto et al., "Session and network support for autonomous context-aware multiparty communications in heterogeneous mobile systems," *International Journal of Handheld Computing Research*, vol. 1, pp. 1–24, 2010.

[12] J. Simoes, S. Sargento, J. Antoniou et al., "Context-aware control for personalized multiparty sessions in mobile multihomed systems," in *Proceedings of the 5th International ICST Mobile Multimedia Communications Conference (Mobimedia '09)*, M. G. Martini and C. Politis, Eds., ACM International Conference Proceeding Series, 2009.

[13] M. Marin-Perianu and P. J. M. Havinga, "D-FLER—a distributed fuzzy logic engine for rule-based wireless sensor networks," in *Ubiquitous Computing Systems*, H. Ichikawa, W. D. Cho, I. Satoh, and H. Y. Youn, Eds., vol. 4836 of *Lecture Notes in Computer Science*, pp. 86–101, Springer, Berlin, Germany, 2007.

[14] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 4–50, 2003.

[15] T. R. M. Braga, F. A. Silva, J. M. S. Nogueira, A. A. F. Loureiro, and L. B. Ruiz, "A tiny and light-weight autonomic element for wireless sensor networks," in *Proceedings of the 4th International Conference on Autonomic Computing (ICAC '07)*, p. 17, IEEE Computer Society, June 2007.

[16] P. Lalanda, J. A. McCann, and A. Diaconescu, *Autonomic Computing—Principles, Design and Implementation*, Undergraduate Topics in Computer Science, Springer, 2013.

[17] D. Acharjee and P. Patel, "Multipath routing sensor network for finding crack in metallic structure using fuzzy logic," *SOURCEProceedings of World Academy of Science: Engineering & Technolog*, vol. 36, pp. 36–324, 2008.

[18] Crossbow Technology, *Crossbow Mote-View 1.2 User's Manual*, 2006, http://www.datasheetarchive.com/dl/Datasheets-SW21/DSASW00417438.pdf.

[19] M. Marin-Perianu, C. Lombriser, O. Amft, P. Havinga, and G. Tröster, "Distributed activity recognition with fuzzy-enabled wireless sensor networks," in *Distributed Computing in Sensor Systems: Proceedings of 4th IEEE International Conference, DCOSS 2008 Santorini Island, Greece, June 11–14, 2008*, vol. 5067 of *Lecture Notes in Computer Science*, pp. 296–313, Springer, Berlin, Germany, 2008.

[20] J. Canada-Bago, J. A. Fernandez-Prieto, M. A. Gadeo-Martos, and J. R. Velasco, "A new collaborative knowledge-based approach for wireless sensor networks," *Sensors*, vol. 10, no. 6, pp. 6044–6062, 2010.

[21] M. A. Gadeo-Martos, J. A. Fernandez-Prieto, J. Canada-Bago, and J. R. Velasco, "An architecture for performance optimization in a collaborative knowledge-based approach for wireless sensor networks," *Sensors*, vol. 11, no. 10, pp. 9136–9159, 2011.

[22] J. C. Cuevas-Martinez, M. A. Gadeo-Martos, J. A. Fernandez-Prieto, J. Canada-Bago, and A. J. Yuste-Delgado, "Wireless intelligent sensors management application protocol-WISMAP," *Sensors*, vol. 10, no. 10, pp. 8827–8849, 2010.

[23] J. C. Cuevas-Martinez, J. Canada-Bago, J. A. Fernandez-Prieto, and M. A. Gadeo-Martos, "Knowledge-based duty cycle estimation in wireless sensor networks: application for sound pressure monitoring," *Applied Soft Computing Journal*, vol. 13, no. 2, pp. 967–980, 2013.

[24] J. C. Cuevas-Martínez, J. Canada-Bago, J. A. Fernández-Prieto, and M. A. Gadeo-Martos, "Propagation of agent performance parameters in wireless sensor networks," in *Highlights in Practical Applications of Agents and Multiagent Systems*, J. B. Pérez, J. M. Corchado, M. N. Moreno et al., Eds., vol. 89 of *Advances in Intelligent and Soft Computing*, pp. 213–221, Springer, Berlin, Germany, 2011.

[25] F. A. C. Gomide and R. R. Gudwin, "Modelagem, controle, sistemas e lógica fuzzy," *SBA Controle & Automação*, vol. 4, pp. 97–115, 1994.

[26] F. Herrera and E. Herrera-Viedma, "Linguistic decision analysis: steps for solving decision problems under linguistic information," *Fuzzy Sets and Systems*, vol. 115, no. 1, pp. 67–82, 2000.

[27] Q. Hauifeng and Z. Xingshe, "Context aware sensornet," in *Proceedings of the 3rd International Workshop on Middleware for Pervasive and Ad-Hoc Computing*, vol. 115 of *ACM International Conference Proceeding Series*, pp. 1–7, 2005.

[28] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 162–175, November 2004.

[29] F. Stajano, D. Cvrcek, and M. Lewis, "Cast iron and concrete: security 667 engeneering for real world wireless sensor networks," in *Applied Cryptography and Network Security*, vol. 5037 of *Lecture Notes in Computer Science*, pp. 460–478, Springer, Berlin, Germany, 2008.

[30] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[31] A. R. Rocha, L. Pirmez, F. C. Delicato et al., "WSNs clustering based on semantic neighborhood relationships," *Computer Networks*, vol. 56, no. 5, pp. 1627–1645, 2012.