Energy-Driven Reconfiguration of Applications for Wireless Sensor Networks

Vittorio Cortellessa
University of L'Aquila, Italy
vittorio.cortellessa@univaq.it

Antinisca Di Marco
University of L'Aquila, Italy
antinisca.dimarco@univaq.it

Daniele Di Pompeo
University of L'Aquila, Italy
daniele.dipompeo@graduate.univaq.it

Francesco Gallo
University of L'Aquila, Italy
franscesco.gallo@univaq.it

Stefano Pace
Smartly srl, L'Aquila, Italy
stefano.pace@smartlysrl.it

Luigi Pomante
University of L'Aquila, Italy
lugi.pomante@univaq.it

Walter Tiberti
University of L'Aquila, Italy
walter.tiberti@graduate.univaq.it

ABSTRACT

The reconfiguration of Wireless Sensor Networks is a well-known problem that has been tackled in the last few years mostly with approaches operating at the network level. In this paper, we introduce an approach for the automated reconfiguration of WSNs, driven by energy problems, at the level of software application. In particular, we define reconfiguration actions with Proteus, that is a framework to specify and implement reconfiguration plans on top of Agilla, that is a mobile agent middleware for Wireless Sensor Networks. A Reconfiguration Engine acts on agents within a WSN to activate reconfiguration plans when required, e.g. when the batteries of several sensors located in a critical geographical area are exhausted. We have started to experiment the approach on a real application.

CCS CONCEPTS

• **Networks** → **Sensor networks**; *Network performance analysis*;
• **Hardware** → *Power and energy*;

KEYWORDS

Wireless Sensor Network; Energy

ACM Reference Format:
Vittorio Cortellessa, Antinisca Di Marco, Daniele Di Pompeo, Francesco Gallo, Stefano Pace, Luigi Pomante, and Walter Tiberti. 2018. Energy-Driven Reconfiguration of Applications for Wireless Sensor Networks. In *Proceedings of ACM/SPEC International Conference on Performance Engineering (ICPE'18 Companion).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3185768.3186312

1 INTRODUCTION

In the era of the Internet of Things, Wireless Sensor Networks (WSN) are increasing their importance because they enable each object to be connected. A WSN consists of spatially distributed autonomous devices using sensors to monitor physical or environmental conditions [3].

Sensing, processing, and communicating become complex activities with a limited amount of energy. Hence, they require cross-layer design approaches that jointly consider distributed signal/data processing, medium access control, communication protocols and application layer. In this sense, one of the leading open challenges in the WSN context is to develop cross-layer energy-efficient solutions to increase the network lifetime without compromising its operability [8].

The lack of stability in WSN claims for frequent reconfigurations aimed at avoiding service degradation. Recently, WSN reconfiguration has been mostly tackled with approaches operating at the network level. In this paper, we introduce a novel application level energy-driven approach for automated reconfiguration of WSNs that leverages on Agilla [6].

Our framework is made up of a *WSN* of IRIS nodes (also called motes), and a *Refactoring Engine*. The latter is a set of Java objects that aim at processing a plan and sending the code to nodes that are targeted by the plan. The Refactoring Engine communicates with the user through a GUI, and it uses a database, namely a *Reconfiguration plans DB*, which contains every available reconfiguration plan and its code. In our approach, each mote runs *TinyOS* as operating system and *Agilla* as middleware on top of it.

TinyOS is a well-known library that provides useful API to interact with the mote's hardware (i.e. sensors) [1].

Agilla is an agent-driven framework that allows the developer to instruct nodes at runtime, without interrupting their service, by sending and running agents on the WSN node or by stopping their execution [6]. Agilla does not naturally provide application logic to apply automatic adaptations. Hence, to overcome this limit and to provide an automated adaptation of WSN to network events (e.g. energy issues), we introduce the Reconfiguration Engine (RE).

In this paper, the RE is meant to autonomously manage reconfiguration plans based on energy properties to reconfigure the network and to guarantee longer life to it. To prove our approach, we have conducted several experiments in our laboratory. As expected, using Agilla there is a higher consumption of the battery than the case without. However, even if the Agilla middleware consumes more battery, it introduces runtime reconfiguration capabilities that are very useful in case of unsafe or critical network status. The experiments we conducted demonstrate that the battery consumption in presence of Agilla in combination with reconfiguration plans, that reconfigure the nodes to smartly manage their battery, is the same of the nodes with no Agilla agents. Hence, the simple reconfigurations used in the experiments permit to eliminate the energy overhead introduced by Agilla and in some cases, opportunely reducing the rate of temperature sensing, it is even possible to increase the uptime of the whole WSN.

This paper is organized as follows: Section 2 overviews the main background concepts; Section 3 describes our approach; Section 4 introduces a case study; related work is discussed in Section 5, and conclusions are given in Section 6.

2 BACKGROUND

In this section we introduce the main concepts on which our framework is built on, that are the Agilla framework and Proteus.

2.1 Agilla

Agilla is a middleware based on a multi-agent paradigm [6], and it supports star topology WSNs. Motes in a WSN cannot communicate to each other, but they must use a particular node, namely "base station", to transmit data (among them and to the user). The base station and the server are directly connected. Thus, the latter has no battery problems.

To fully support the Agilla framework, a "virtual machine" runs on each node belonging to the network under analysis. The detriment of energy due to the virtualization is fully compensated by the agent-based paradigm. An agent is a snippet of TinyOs [1] code, and its behaviour can be edited by avoiding to restart the node. Hence, this functionality results to be very useful when a system must be alive during a reconfiguration. Furthermore, from its first release, Agilla provides a user-friendly Java-based application, namely "AgillaInjector", which allows the user to directly write her/his snippet of code and, additionally, it provides functionality for interacting with the nodes. The AgillaInjector is in charge of taking the user code and injecting it into the whole WSN or a subset of it (even a single node).

2.2 Proteus

Proteus is a reconfiguration framework that exploits its language, grounded on a proprietary XSD, aimed at building and managing rules for software reconfiguration [12]. Rules are generated and managed using the concept of *virtual membrane*, directly borrowed from biology world. A virtual membrane embraces a subset of resources grouped by shared properties (e.g. battery type, sensor type) and these resources are the only ones that are targeted by the rules. Moreover, a membrane allows users to: i) select a resource or group of resources belonging to the system, and ii) define new interactions within the system, which consequently modifies their behaviour.

The Proteus module deals with obtaining, parsing, interpreting and executing a reconfiguration plans and it uses a socket connection to send a plan into the network. Hence, a reconfiguration contains information about the properties that nodes under reconfiguration must respect, the reconfiguration action to be taken, and the agent snippet code to be injected.

Listing 1 depicts a simple example of a reconfiguration plan, in which the `property` tag and its sub-tags (see lines 3-7) identify which node(s) is involved. `Act` tag, instead, contains the snippet code, i.e. the `body` tag (see lines 11-29). Users can write within the latter, snippet code as complex as they need since Proteus supports the whole grammar of Agilla agent[*]. In particular, that snippet instruments the node whose $battery$ is lower than TH_{BATT} threshold to sense the environment temperature less frequently, that is every 10 second (line 30).By exploiting the `property` tag, we can define new membranes, and, by further exploiting different operators on `<op>` sub-tag, it is possible to define much more dense virtual membranes.

Listing 1: The Reconfiguration Plan for adaptation in case of low battery

```
 1  <?xml version="1.0" encoding="UTF−8" standalone="yes"?>
 2  <program virtualid="Virtual Add Class" persistence="y">
 3    <property propertyid="Prop 1">
 4      <item>battery</item>
 5      <op> < </op>
 6      <val>${TH_BATT}</val>
 7    </property>
 8    <act actionid="Add State 1">
 9      <type>field</type>
10      <action>add</action>
11      <body>
12          pushc 0
13          setvar 0 // set heap[0] = 0 (init. seq. no.)
14        BEGIN pushc 26
15          [...]
16          pushc TEMP
17          sense // sense temperature
18          copy
19          pushc BATTERY // sense battery value (in Volt)
20          pushc 24
21          copy
22          [...]
23      </body>
24    </act>
25  </program>
```

3 APPROACH

In this section, we describe our approach for energy-aware reconfiguration of a Wireless Sensor Network, by introducing a description of our framework.

[*] Agilla Documentation - http://mobilab.wustl.edu/projects/agilla/docs/index.html.

3.1 Framework

In order to overcome the limit introduced by Agilla and to provide an automated adaptation of WSN to network events (e.g. energy issues), we introduce the Reconfiguration Engine (RE) framework, which is illustrated in Figure 1. The most important asset in our RE is Proteus [12], which is a tool that allows us to specify application-level adaptations by writing XML files conforming to a proper XML Schema [14]. It is based on the bio-inspired membrane concept, namely *Virtual Membrane*, that provides logical compartments of a system. For example, in the WSN of Figure 1, the three closed curves depict three different membranes. Each membrane embraces a set of WSN nodes that share some properties (e.g. geo-coordinates, type of battery, kind of sensing).

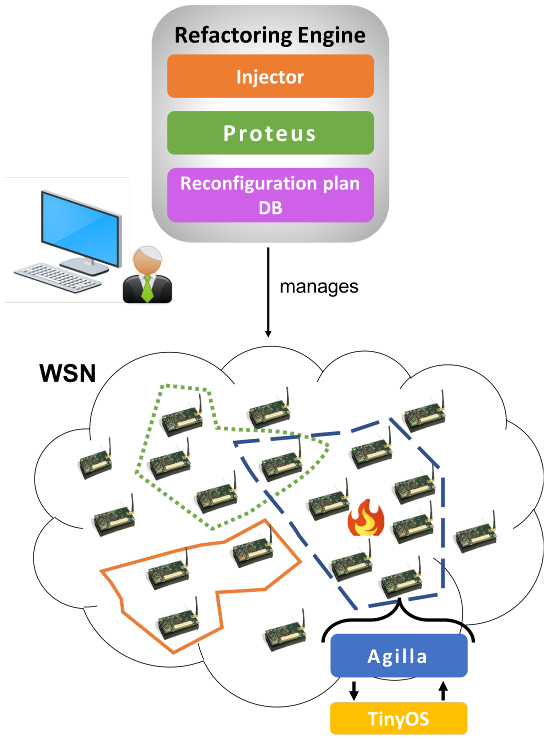

Figure 1: An overview of our framework

To interact with the WSN nodes, we extend the original Agilla agent injector in RE, by enabling it to process Proteus files and to instrument relative motes. The injector is named as WSN Manager in Figure 1. It takes selected Proteus XML files (i.e. reconfiguration plans) as input, and it properly instruments the Base Station. Another important functionality of RE is to provide a GUI that enables the user to interact with the system.

The last asset in RE is the Reconfiguration plans DB, in which all available plans are stored. Autonomously, our reconfiguration engine can select the best plan to react to network issues.

The Base Station, instead, is in charge to send Agilla agents into the WSN motes and to collect, elaborate, and send network information to the WSN Manager. Together with the Proteus asset and WSN Manager, they can autonomously interact with motes in order to properly react to the network bad events.

3.2 Energy-aware membrane

By exploiting the Proteus membrane concept, we introduce here logical predicates with the aim to support the detection of bad network status or unsafe human situation. In order to support an automated network reconfiguration, we introduce logical predicates to describe the current state of the network and to group nodes with same properties in the same membrane.

The proposed approach aims at extending the WSN lifetime by increasing the battery life of each node belonging to it. Hence, the initial problem of making the WSN life longer can be reduced to make the battery life of its nodes longer. The WSN life duration is determined by the time that elapses between the instant the WSN is active (i.e., all its nodes are active) and the time the WSN changes topology due to even a single node stop for battery depletion.

In order to allow automated reactions to network events (i.e. energy issue, high environmental temperature), we introduce a novel methodology for describing network states as logical predicates. The idea is taken from the representation of performance antipattern as logical predicates, introduced by Cortellessa et al. in [5]. Due to space limitation, we only report three new logical predicates that identify three different membranes: i) Battery Lack membrane, ii) High-Temperature Border membrane, and iii) the Safe Neighbours membrane. The latter is composed of nodes that are neighbours of motes with low battery availability (i.e., in Battery Lack membrane) and that are sensing high-temperature (i.e., in High-Temperature Border membrane). To deal with predicates like the last one, RE knows where motes are deployed, saving their geo-position into a database.

Note that the High-Temperature Border membrane is here defined to show how our framework manages application-dependent events (i.e., a dangerous high-temperature). Instead, the Battery Lack membrane is defined by energy-aware predicates aimed at identifying critical and unsafe situations in a Wireless Sensor Network, where nodes typically have a limited battery, and they are not easily accessible. On one hand, we can detect network criticality when a battery of a node is below a threshold; on the other hand, we can additionally detect an unsafe state for users when, for example, a high temperature is sensed (it could be started a fire in the room). Our framework can autonomously select the correct reaction based on the network state.

Finally, Safe Neighbours membrane shows how to combine membranes to deal with 2nd-order complex situations the WSN must autonomously react to.

Battery Lack.

$$IF \quad n.battery \quad \leq \quad \$TH_{BATT} \quad THEN \quad n \quad \in \quad N_b \qquad (1)$$

In Predicate 1 above, the N_b membrane encloses those nodes whose battery level is under a pre-defined TH_{BATT} threshold. This condition supports the identification of those nodes that are in critical status.

In case of no-high-temperature detection, motes must send their temperature value at a pre-defined rate. In this case, when a node has low energy, the system must be able to identify the best reconfiguration plan that increases the uptime of the whole network. In the current version of our framework, we have identified and developed two reconfiguration plans. The first one is a reduction of the sending rate, and the second one consists in turning off a suffering node, by introducing in the membrane a new node that could be a neighbour node or a new one. The latter task requires human interaction. The system is aware of this change, and it reconfigures the new node after some interactions with the base station.

High-Temperature Border.

$$IF\ node.temp\ \geq \$TH_{HT}\ THEN\ n \in N_{ht} \tag{2}$$

Predicate 2 above depicts the logical predicate that the framework uses for identifying a high-temperature in the physical environment (e.g., when a fire occurs). In particular, we define by N_{ht} the subset of motes with an environment temperature higher than a pre-defined TH_{HT} safety threshold. Thus, each node n with environment temperature $n.temp$ greater than TH_{HT} will be part of the "high-temperature virtual membrane" N_{ht}. This condition can be used, for example, to estimate the fire perimeter as narrow as possible.

This definition of such a virtual membrane can be also useful in an ordinary case when the environment temperature should be under control. For example, if a WSN is used for sensing in a botanical garden, it is most important to control the temperature in different areas of the garden.

Safe Neighbours.

$$IF\ \{N_{ht}\ \cap\ N_b\}\ \neq\ \emptyset\ THEN$$
$$\forall n \in\ \{N_{ht}\ \cap\ N_b\}\ \exists k\ \in\ N\ \rightarrow$$
$$F_{dist}(n,k)\ <\ \$TH_{DIST}\ AND\ k.battery >\ n.battery \tag{3}$$

Predicate 3 represents an example of a new membrane specification that uses pre-defined ones. In particular, this predicate leverages the intersection between two membranes to define more complex conditions. When the intersection $N_b \cap N_{ht}$ is not an empty set, we are sure that exists at least a node n that has triggered a high-temperature alarm and has a low battery. In this case, if the predicate is true then for each of these nodes there exists a close node belonging to the whole set N of nodes (i.e., one with $F_{dist}(n,k)\ <\ \$TH_{DIST}$) that has a sufficient amount of battery (i.e., $k.battery >\ n.battery$) to replace the suffering node. We can then apply our planned reconfiguration, which consists to send Agilla agents to the selected node, by exploiting mote's geotagging information, in order to instrument it properly.

4 APPROACH AT WORK

In this section, we present a real-life case study where we have applied our approach, in order to present runtime reconfiguration capabilities of real WSN applications.

Among all Agilla compliant devices, for the sake of our approach validation, we select the IRIS node family on which TinyOS (i.e. the operating system) is running. Every mote has a 8 MHz μ-controller, 128 KB of flash memory and 8 KB of data RAM. We selected a `MIB520CB` device as base station[†], which provides USB connectivity to IRIS family of Motes for communication and in-system programming, and `MDA100` as sensor board that allows us to use different sensors for different scopes[‡].

4.1 The Wildfire Rescue Application

We consider here an extract of Wildfire Rescue real-life application from the VISION ERC project (ERC-240555), see Figure 2. In the following, we show the effectiveness of the reconfiguration occurring upon energy-related alarms.

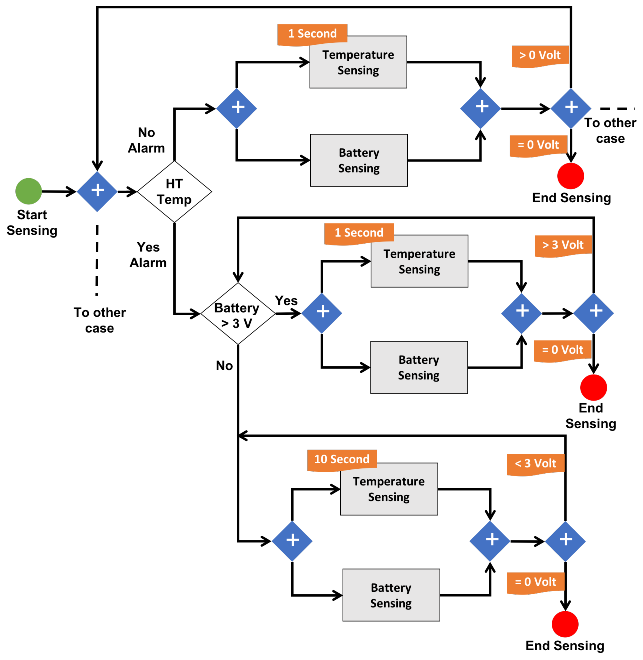

Figure 2: Mote reconfiguration process

Figure 2 depicts the reconfiguration process a single mote can undergo during execution of our case study. In the figure, rectangles depict sensing actions, white diamonds depict decision points, and labels represent rates and battery measurements. For the sake of readability, we do not show in Figure 2 each possible path, but we just report cases of no alarm occurs (upper level of the figure) and of Fire Alarm first occurs (lower part of the Figure).

In the absence of an alarm, every mote senses the environment temperature at every second and sends the battery level to the base station. To prevent a node's death, each node sends its battery level together with a temperature measure. By sending the battery level together with a temperature measure, it should avoid the node's death.

In case of a Fire Alarm, the High-Temperature Border membrane is determined and a reconfiguration plan is sent to all its nodes. Such a plan aims to manage the unsafe situation. If a subsequent low battery alarm is triggered, that is in case of the battery of a node is below 3 V, our framework sends a special agent to that node in order to reduce the rate of the temperature sensing, sensing it every 10 seconds.

[†]http://www.aceinna.com/userfiles/files/Datasheets/WSN/6020-0091-03_A_-DATA_SHEET_MIB520_USB_INTERFACE_BOARD-03_A_MIB520CB.pdf

[‡]http://www.memsic.com/userfiles/files/Datasheets/WSN/mts_mda_-datasheet.pdf

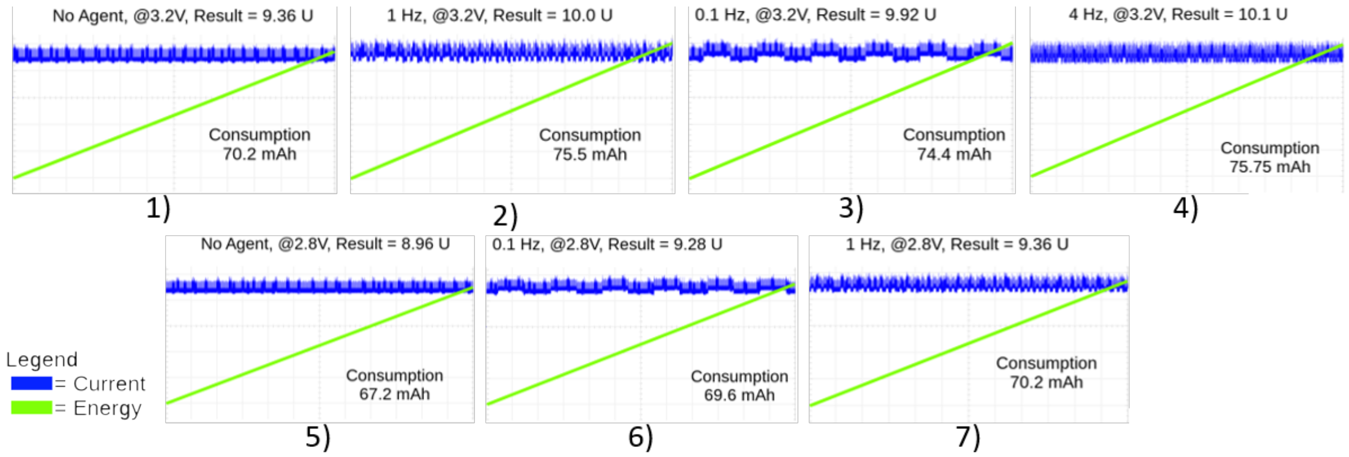

Figure 3: An overview of obtained results

4.2 Energy Consumption Validation

Here, we report our experiments to validate our approach with respect to energy consumption of nodes, by comparing the energy consumption both on nodes without Agilla agent and on nodes augmented with Agilla agents. For the sake of pages limitation, we report only the most interested scenarios. In particular, we classified the scenarios in: *idle mode*, where no agent is running on the node; two *normal modes*, where there is an agent sensing and sending messages (first at 1 Hz frequency, then at 4 Hz frequency); a *slow mode*, where the agent senses and sends messages at a slower rate (i.e. 0.1 Hz) and three *alarm modes* where the battery voltage level is detected to be below the threshold (i.e. $\$TH_{BATT}$) in the case where no agent, an agent sensing at 1 Hz and an agent sensing at 0.1 Hz.

Setup. To simulate on field usage, we use a pair of AA batteries as a power supply line (i.e. V_1) on the IRIS mote, and we use an oscilloscope in order to retrieve measurements. The shunt resistor resistance value is not important (given that the flowing current is enough to power on the node), the only characteristic needed is that the selected value is precise, stable and as linear as possible. Without lack of repeatability, we choice, for our experimentations, a resistor of 8 Ohm.

Results. Here, we report the results of our experiments. In particular, we have performed our measurements in two different cases: i) in case of fully charged node, i.e. Scenarios 1-4„ and ii) in case of no fully charged node, i.e. Scenarios 5-7, as reported below.

In Figure 3, for each scenario, we report the current absorbed by the node and the resulting energy consumption within the 60 seconds time window (i.e., x-axis).

Scenario 1 shows a mote running in *idle mode* i.e. (with no Agilla agent running on it) in the case of a fully charged battery. In scenarios 2, 3 and 4 Agilla agents are added that sense temperature each 1 second, 0.25 seconds, and 10 seconds, respectively, which produces a varying detriment of the battery. It is worth to notice that Agilla agents consume around 7% more energy respect the idle mode, and this detriment is due to the execution of Agilla agents. What we have seen, is that Scenario 1 is the least battery hungry with 70.2 mAh of energy consumption. Instead, by using Agilla agents, i.e. Scenario 2 to Scenario 4, we have noticed an increment of the energy consumption. In particular, with sensing each second and every 0.25 seconds (i.e. Scenario 2 and Scenario 3) the detriment of the battery is the highest, around 75 mAh; instead by sensing every 10 seconds we have seen a reduction of 1 mAh of the energy consumption, i.e. 74 mAh. Though this is a light reduction of the energy consumption, it should be a meaningful reduction when motes are dived in a real context.

The last three scenarios, i.e. Scenario 5, 6, and 7 respectively, represent cases of a not fully charged battery. We have noticed that an Agilla agent consumes less energy with respect to the previous four scenarios thanks to the application of a reconfiguration plan. In fact, in this case the Agilla agent increases the energy consumption by around 4%, instead of 7% that we have obtained before the reconfiguration.

We like to remark that the frequency used to send messages influences the current behaviour which shows "ripples" due the additional consumption by the radio apparatus of the node. However, the experiments demonstrate that the battery consumption in presence of Agilla in combination with reconfiguration plans that reconfigure the nodes to smartly manage their battery, is the same of the nodes with no Agilla agents, that is $70.2mAh$. Hence, the simple reconfigurations used in the experiments permit to eliminate the energy overhead introduced by Agilla and in some cases, opportunely reducing the rate of temperature sensing, it is even possible to increase the uptime of the whole WSN.

5 RELATED WORK

Recently, in order to achieve a formal validation of refactoring changes and for preserving the behaviour of a WSN, some work have been conducted [7, 10, 13, 15, 16].

Liang et al. grounded their approach on a role-based context [9]. By still remaining in a dynamic evolution context, in order to solve an energy-aware issue, they share the load among nodes by leveraging their battery status. In our approach the energy assumes a relevant role, hence in this current version we do not support any sort of sharing information, but we have already planned to introduce such reconfiguration plan. We use the energy status for creating a virtual membrane and we control what action the nodes belonging to that membrane have to take, and thus we can reconfigure the network in order to maintain the whole functionality.

In [18] the authors proposed a middleware solution to efficiently delimit and reconfigure the necessary portion of sensor software instead of updating the full binary image. Besides, in [11], they proposed a biologically inspired node configuration scheme in shared reconfigurable sensor networks that can adapt to the changing environment and efficiently utilize WSN resources.

A dynamically reconfigurable system of agents is described in [4], where the authors have proposed a type of agent that is capable of migrating from hardware to software (or vice versa) based on a decision that is made inside the application or even by an external trigger. Moreover, in their approach a dynamically reconfigurable agent can be executed as either a pure software agent or a hardware one.

To the of best our knowledge, the approach presented by Silva et al. [17] is the most related one to our scope. In fact, they have shown a stable energetic consumption of the Iris sensors in comparison to other competitors with similar characteristics. However, they do not consider dynamic reconfiguration.

Radio transmission is one of the most energy-hungry aspect in a WSN. In order to increase the lifetime of motes in [2], the authors presented a novel idea, that is clustering the network to reduce the required radio power by appointing cluster head nodes, which collect data from owned sub-network. Differently, we create clusters not only by distance but also using different attributes, for example the environment temperature level.

Finally, in [10] a distributed peer-to-peer framework has been introduced for sake of energy efficiency in the surveillance domain. Our approach differs from it because we use a star topology network, and we refer to our engine to identify the reconfiguration plan to apply.

6 CONCLUSION AND FUTURE WORK

We have described a framework for a runtime reconfiguration of a WSN based on energy concepts.

The framework supports some reconfiguration actions, such as the replacement of a battery suffering node, both in a normal case and in a critical status. The framework is fully able to identify when a critical situation is coming up.

Among other, as short-term future work we will experiment the framework on joint performance and energy aspects. Moreover, we plan to extend the reconfiguration plan database by considering other performance parameters. We have also planned to apply our approach in large size Wireless Sensor Networks to analyze its scalability.

ACKNOWLEDGMENTS

This research work has been supported by the GAUSS National Research Project funded by the MIUR under the PRIN 2015 program under Grant No.: 2015KWREMX

REFERENCES

[1] TinyOS Alliance. 2017. TinyOS. (2017). http://tinyos.stanford.edu/tinyos-wiki/index.php/TinyOS_Documentation_Wiki

[2] S. V. Basu, K. P. Ashwin, N. K. Neti, and B. S. Premananda. 2017. Improving the network lifetime of a wireless sensor network using clustering techniques. In *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT).*

[3] Arne BrÃűring, Johannes Echterhoff, Simon Jirka, Ingo Simonis, Thomas Everding, Christoph Stasch, Steve Liang, and Rob Lemmens. 2011. New Generation Sensor Web Enablement. *Sensors* 11, 3 (2011), 2652–2699.

[4] David Cemin, Marcelo Götz, and Carlos Eduardo Pereira. 2014. Dynamically reconfigurable hardware/software mobile agents. *Design Automation for Embedded Systems* 18, 1 (01 Mar 2014).

[5] Vittorio Cortellessa, Antinisca Di Marco, and Catia Trubiani. 2014. An approach for modeling and detecting software performance antipatterns based on first-order logics. *Software and System Modeling* 13, 1 (2014).

[6] Chien-Liang Fok, Gruia-Catalin Roman, and Chenyang Lu. 2009. Agilla: A Mobile Agent Middleware for Self-adaptive Wireless Sensor Networks. *ACM Trans. Auton. Adapt. Syst.* 4, 3 (July 2009).

[7] J. Guevara, E. Vargas, F. Brunetti, and F. Barrero. 2013. Open architecture for WSN based on runtime reconfigurable systems and the IEEE 1451. In *2013 IEEE SENSORS.*

[8] V. C. Gungor and G. P. Hancke. 2009. Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches. *IEEE Transactions on Industrial Electronics* 56, 10 (2009).

[9] Liang He, Yu Gu, Jianping Pan, and Ting Zhu. On-demand Charging in Wireless Sensor Networks: Theories and Applications. In *Proceedings of the 2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems.*

[10] Tian He, Sudha Krishnamurthy, John A. Stankovic, Tarek Abdelzaher, Liqian Luo, Radu Stoleru, Ting Yan, Lin Gu, Jonathan Hui, and Bruce Krogh. 2004. Energy-efficient Surveillance System Using Wireless Sensor Networks. In *Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services (MobiSys '04).*

[11] C. M. Hsieh, Z. Wang, and J. Henkel. 2013. DANCE: Distributed application-aware node configuration engine in shared reconfigurable sensor networks. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE).*

[12] Antinisca Di Marco, Francesco Gallo, and Franco Raimondi. 2012. PROTEUS: A Language for Adaptation Plans. (2012).

[13] M. D Nguyen. 2007. Reconfigurable Wireless Sensor Networks: A Survey and Future Works. (2007).

[14] Stefano Pace. 2015. Development Framework for Adaptive Wireless Sensor Networks Applications. (2015).

[15] V. N. P. Prasuna, A. Valarmathi, and J. A. V. Selvi. 2016. Parametric analysis of a novel reconfigurable Wireless Sensor Network architecture. In *2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS).*

[16] Natheswaran S and Athisha G. 2014. Remote reconfigurable wireless sensor node design for Wireless Sensor Network. In *2014 International Conference on Communication and Signal Processing.*

[17] Edgar M Silva, Pedro Maló, and Michele Albano. 2016. Energy Consumption Awareness for Resource-Constrained Devices: Extension to FPGA. *Journal of Green Engineering* 6, 3 (2016), 229–256.

[18] Amir Taherkordi, Frederic Loiret, Romain Rouvoy, and Frank Eliassen. 2013. Optimizing Sensor Network Reprogramming via in Situ Reconfigurable Components. *ACM Trans. Sen. Netw.* 9, 2 (April 2013).