Bringing Context Awareness to IoT-Based Wireless Sensor Networks

Shashank Gaur

CISTER-TR-150106

2015/03/23

Bringing Context Awareness to IoT-Based Wireless Sensor Networks

Shashank Gaur

CISTER Research Center

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: sgaur@isep.ipp.pt

http://www.cister.isep.ipp.pt

Abstract

Wireless Sensor Networks (WSN) are already enabling and enhancing a large number of pervasive computing applications in homes, offices, production facilities, and vehicles, just to name a few. Despite the tremendous evolution achieved in terms of robustness, reliability, maintenance costs, interoperability and other areas, WSN are still remarkably hard to program. This work addresses specifically the case of IoT-based WSN, and motivated by the need to facilitate the development of context-aware WSN applications, this research proposes to develop a framework that allows the user to focus on specifying the behavior of the application, and offloading the concerns with reconfiguration, adaptation, resource management, code deployment and interoperability to the framework itself.

Bringing Context Awareness to IoT-Based Wireless Sensor Networks

Shashank Gaur
CISTER Research Institute
ISEP, Polytechnic Institute of Porto, Portugal
Email: sgaur@isep.ipp.pt

***Abstract*—Wireless Sensor Networks (WSN) are already enabling and enhancing a large number of pervasive computing applications in homes, offices, production facilities, and vehicles, just to name a few. Despite the tremendous evolution achieved in terms of robustness, reliability, maintenance costs, interoperability and other areas, WSN are still difficult to program. This work addresses specifically the case of IoT-based WSN, and is motivated by the need to facilitate the development of context-aware WSN applications. This research proposes to develop a framework that allows the user to focus on specifying the behavior of the application, and offloading the concerns with reconfiguration, adaptation, resource management, code deployment and interoperability to the framework itself.**

I. Introduction

Nowadays, wireless sensor networks (WSN) have penetrated into the daily life of common user(s). In past, WSNs were designed to perform a single task again and again, often in very controlled environments. Nowadays they are stepping towards becoming more and more pervasive, powerful, reliable, easy to maintain, and incorporating an increasing number of functionalities. Such advancement enables the development of self-adapting WSN, which respond to the environmental conditions (or context, such as location, activity, time). This work is motivated by such applications, where the context adaptation can be made by performing in-network processing of simple rules (e.g. detection of presence, changes in temperature, light), but processing outside the WSN, in more powerful nodes such as mobiles or in the Cloud, is also supported.

The work is supported on the Internet of things (IoT) paradigm. IoT is allowing to make progress towards globally interoperable WSN, by introducing standards upon which applications can be built, such as the 6LoWPAN [1] protocols, RPL [2], and CoAP [3]. There is also recent work that enabled reconfigurable in-network processing based on these standards[4]. Even with all the research on facilitating the programming of WSN[5], programming IoT-based WSNs is still a difficult task, and no framework available allows to program a WSN without considering the resources of each node individually.

We propose to develop a framework which provides ease of access for a programmer, while providing support for various necessary components of a WSN, such as resource management, in-network processing, and application deployment. To do so, first we present a programming model to write applications without worrying about the details of each node. Secondly, we introduce a resource management framework which can take advantage of the programming model. At last, we intend to add the capability to detect various context changes in the system and according to that deploy new applications across complete or part of the sensor network.

II. Context Aware Framework

The idea of context-aware applications involving WSN has been around for long[6], [7], but these doesn't provide user the capability to write context aware applications. We propose such a framework, which exempts the user from many liabilities while programming context based functionalities. The programmer can write application code for different context, and provide a set of rules for the activation of a context. These rules use inputs (sensor readings) from the WSN, and are usually processed in the network. In the case the rules require substantial processing, the framework includes provisions to deploy the functionality in more powerful nodes, or in the Cloud.

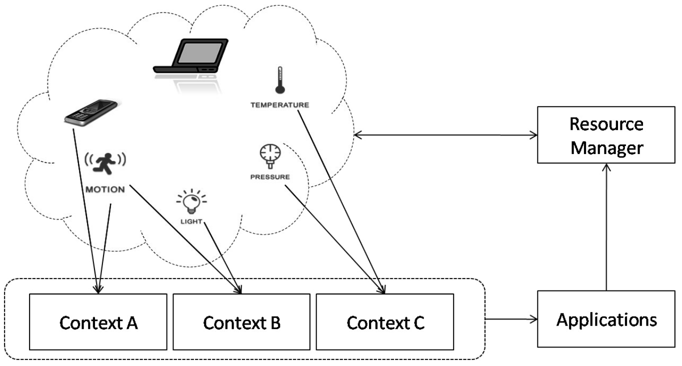


Fig. 1. Framework for Context Aware Sensor Network

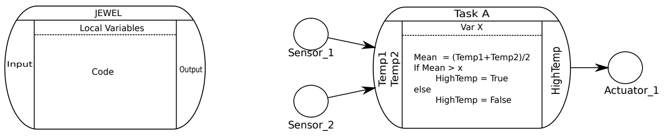
The rules will dictate the active context, which defines the set of applications that should be executed in the WSN. When a new context is activated, this will trigger the deployment of the respective application code. This process is overseen by the resource manager, which decides how the application code will use the resources (processing, communication, sensing) of the WSN. Both the context and resource manager will be programmed following a rule based approach. Figure 1 depicts how WSN contributes to various context(s) and resource manager delivers appropriate application to the WSN.

III. Programming Paradigm

To achieve our goal of easing the development of IoT-based context-aware WSN, we want to keep the user free from responsibilities other than expressing the functionalities for desired task(s), i.e. to follow the application goal. On the other hand, the code must be expressed in a modular way, so that the resource manager can easily make decisions about usage of the available resources in the WSN. Hence we propose a declarative programming model, where the user can write the application in self-contained blocks of code, and these blocks will process only the values received as input(s) and send a resulting output(s). We call these blocks of code Jewels and the user builds applications by defining and connecting one or many Jewels.

A. *Jewels*

A Jewel has four parameters, *Input*, *Output*, *Code*, and *Local Variables* as shown in Figure 2(a). The *Code* for a Jewel defines its functionality. It is conceived in such a way that it does not depend on which device or what data is given as input and/or output. That makes the code isolated from any changes that may occur in the devices associated with it. The *Local Variables* helps the code execute locally and they are extra information which is not obtained from elsewhere. A Jewel can have more than one *input* from or *output* to other Jewels.



(a) Jewel Architecture (b) Example

Fig. 2. Jewel

The Jewel in Figure 2(b) is written to detect if temperature in a room is high enough. it takes two temperature sensor values as input and calculates the mean. If the mean is more or less than a local variable, the jewel provides true or false respectively as output.

Due to architecture of Jewels, the programmer is free to write tasks in form of jewels or may be reuse and build upon existing ones. Currently we are developing the support for writing, executing and deploying jewels using an IoT protocol stack (namely 6LowPAN, and CoAP).

IV. Resource Management

The resource manager of the framework, is able to collect information about all resources (sensors, actuators, embedded devices, etc.) required for the jewels from the WSN. This includes the capabilities of a node such as physical parameters it can measure, processing capabilities available on the hardware, energy parameters of the node, its neighbors etc. Using all this information, and by analyzing the jewels, it selects how to deploy the application.

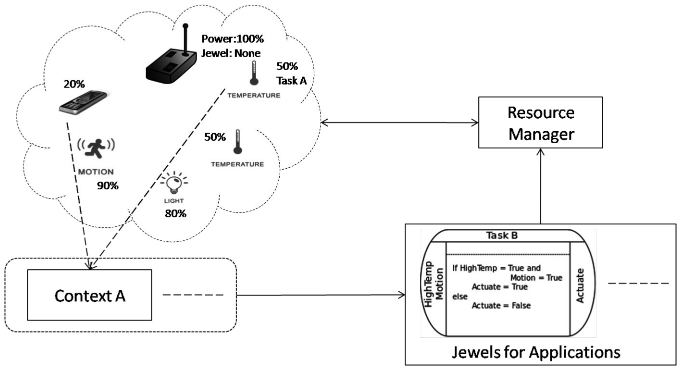


Fig. 3. Resource Management for Context Aware Sensor Network

A Jewel is written only considering its inputs and outputs, and includes simple definition of the resources required for its execution. The binding between the Jewel and the resources for its execution only happens at runtime. Consider the simple example of the Jewel specified in Figure 2(b), where "Task A" is completing a single application. Assume due to a new context, before providing output to the actuator, it is necessary to check motion in the room. This requires another jewel to complete the application. As shown in Figure 3, the resource management acquires the knowledge such as Power and current jewel on nodes and then deploys new jewel "Task B" in the WSN.

At runtime, it is a responsibility of the resource manager to decide which sensors of the room to use (assuming the room has more than two). Also due to the independence of jewel, the resource manager is free to execute the jewel on any node in the network. The resource manager decides how to deploy a jewel based on the various costs associated with the resources (communication, computation, sensors). Deploying the jewel on a sensor node which provides input locally may save communication cost, but deploying the jewel on a powerful node may save energy.

V. Conclusion

With the increase in number of IoT-based smart devices, the need of simplifying the programming of such systems is obvious. This work in progress is a contribution to future of IoT-Based WSNs. The immediate goal is to implement the programming model described earlier, which allows the programmer to be free from responsibilities other than expressing the functionalities for desired task(s). To support this, the framework will require solutions for distributed resource management, in-network processing, and code deployment.

Acknowledgment

The work mentioned above is done jointly with my advisors Prof. Eduardo Tovar, Prof. Nuno Pereira and Dr. Vikram Gupta at CISTER research institute. Especial thanks to Prof. Nuno Pereira for help in editing and reviewing this document. This work was partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology) and by ERDF (European Regional Development

Fund) through COMPETE (Operational Programme 'Thematic Factors of Competitiveness'), within project(s) FCOMP-01-0124-FEDER-037281 (CISTER) and [FCOMP-01-0124-FEDER-020312] (SMARTSKIN)

References

[1] Z. Shelby and C. Bormann, *6LoWPAN: The wireless embedded Internet*. John Wiley & Sons, 2011, vol. 43.

[2] T. Winter, "Rpl: Ipv6 routing protocol for low-power and lossy networks," 2012.

[3] C. Bormann, A. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes," *Internet Computing, IEEE*, vol. 16, no. 2, pp. 62–67, March 2012.

[4] D. Alessandrelli, M. Petraccay, and P. Pagano, "T-res: Enabling reconfigurable in-network processing in iot-based wsns," in *Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on*. IEEE, 2013, pp. 337–344.

[5] G. P. P. Luca Mottola, "Programming wireless sensor networks: Fundamental concepts and state of the art," 2011.

[6] A. Wood, J. A. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru, "Context-aware wireless sensor networks for assisted living and residential monitoring," *Network, IEEE*, vol. 22, no. 4, pp. 26–33, 2008.

[7] D. Salber, A. K. Dey, and G. D. Abowd, "The context toolkit: aiding the development of context-enabled applications," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 1999, pp. 434–441.