WSN and P2P: a self-managing marriage

Gustavo Gutiérrez, Boris Mejías, Peter Van Roy,
Département d'ingénierie informatique, Université catholique de Louvain – Belgium
{firstname.lastname}@uclouvain.be
Diana Velasco, Juan Torres,
Pontificia Universidad Javeriana - Cali
{dlvelasco,juantorres}@puj.edu.co

Abstract

Wireless sensor networks are designed for a very wide, yet specific, purpose. Their components have processing and power limitations. Due to these limitations, decisions by running complex algorithms with the information collected by sensors must be done in components external to the WSN. This document presents a combination of WSN and p2p *networks to ease the development of systems that rely on WSN functionality. As a result, we propose the creation of a programing abstraction that allows developers to concentrate on the functionality of the developing system. We also propose the use of feedback loops as a way to design and develop the components of the abstraction and to define self-managing behavior for them. Those components should be also lowly coupled, interchangeable and extensible.*

1 Introduction

Wireless sensor networks (WSN) have been used to develop solutions for a wide range of problems, ranging from military applications to environmental and habitat monitoring [1]. WSN are suitable for problems with harsh environmental conditions that need unattended operation. Although they are mostly used to *collect* fine grained environmental or ambient information (i.e., pressure, temperature, etc.), they can also be used to *actuate* on their surrounding environment (e.g., by controlling mechanical actuators). Several issues need to be taken into account when designing WSN-based solutions; most notably, the fact that nodes are low power devices in which both computational resources and communication capabilities are limited.

Nowadays, we find that WSN-based applications are very difficult to develop, deploy and maintain. Moreover, there are situations in which is necessary to collect information coming from different, even geographically distant sites; WSN nodes, because of their limited communication capabilities, are not able to perform such collection tasks. Moreover, the processing power of network devices is not usually enough to process the collected information and take decisions from it. As a a consequence, resulting WSN-based applications are ad-hoc, costly, and difficult to maintain, scale, and extend (e.g., to adapt for new environment conditions).

P2P networks bring a lot of flexibility to the application design, and could overcome most of the WSN problems mentioned above. Nevertheless, distributed systems design is, in general, a very difficult task. Most of the problems are related to the interaction among its components. In this sense, *feedback loops*, as introduced in [5], appears as a promising design pattern to reason about all system elements, and, in conjunction with a suitable software abstraction, could be integrated to ease the design and development of WSN-based applications.

We propose the development of a programming abstraction to ease the design and construction of WSN-based applications in conjunction with *p2p* networks. This abstraction will provide implementation of communication protocols to support communication between peers and indirectly between WSN attached to the system.

The rest of the paper is structured as follows. In Section 1 we give a short introduction about why the development of WSN-based applications is difficult in some circumstances; we then describe our proposal in Section 2 on combining them with *p2p* networks. Section 3 gives an introduction to feedback loops and the way we plan to include them in our proposal. Finally, Section 4 present an example of a case study and conclusions on what we expect from the proposed abstraction are presented in Section and 5.

2 Combining *p2p* and *wireless sensor networks*

As hinted in the introduction, *p2p* networks can be a good complement of WSN in application development. On one side, *p2p* are well suited to communicate high-end nodes with considerable computational power. On the other side, WSN are best suited to capture environmental or ambient information in extreme conditions. The problem then consists in integrating both architectures to cooperate for a given functionality.

To build an abstraction that allows for the construction of systems on its top, two aspects should be considered. First, the distribution support built in the abstraction must be orthogonal to the functionality provided by the intended application. In this way, we expect to provide a generic platform for application development. This is why we do not go into details of the WSN nor the *p2p* system. Second, the abstraction should offer different modules for general purposes. For instance, as communication between nodes is done by message passing, different broadcast implementations should be available.

Every component implementation will be designed using the feedback loop abstraction (to be discussed in Section 3). This should favor a number of desirable features: low-coupled design among them, reuse, extensibility and extensive use of multi-core hardware architectures. More importantly, it will provide autonomous control to the system, being able to monitor and correct itself, increasing its self-management.

By doing minor assumptions on the interaction between *p2p* and WSN nodes, it is possible to guarantee that the abstraction provides the enough flexibility to be adaptable to different WSN functionalities. Indeed, there is no restriction in the way the WSN is set up, for instance, nodes in it might run different software [3, 2]. These assumptions comprise:

- Nodes in the *p2p* network may have, or not, a WSN attached.
- They communicate with its WSN by asynchronous message passing.

Figure 1 depicts an example of the structure of a running application, in which a *p2p* network (with nodes labeled *A, B, C, M, N*) cooperates with three WSN (labeled *1, 2, 3*). Nodes *A, B,* and *C* are connected to WSNs *WSN 2, WSN 1*, and *WSN 3*, respectively. Nodes *M* and *N* are not attached to any WSN but are intended to process the information collected through *A, B* and *C*. Connections between nodes represent communication by asynchronous message passing.

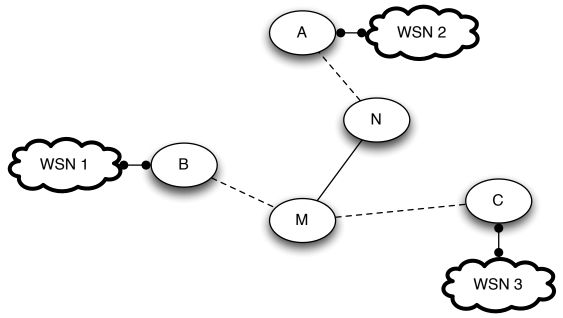

Figure 1. Structure example

In particular, we use dashed lines to represent that communication between both nodes is not constant. This means, for instance, that node *A* may join or leave the network at any given time.

Assumptions previously listed comprise, for instance, the case when a given *p2p* node, based on processed information, changes the way in which nodes inside the WSN behave. This feature is very interesting since allows the complete system to adapt by itself to environment changes. For example, suppose that *WSN 1* monitors temperature on its surrounding and sends the resulting value to *B*, after some processing node *B* determines that given the value reported for the temperature is crucial to measure humidity. *B* does this by sending a message back to *WSN 1* asking it to measure the new variable. This case is possible only if the underlying hardware in *WSN 1* is able to do that.

No constant communication is possible because the resulting *p2p* network should be open. That means, nodes can join the network at any time. The case in which mobile sensors, like described in [2], is also supported. One sensor node can migrate from one WSN to other, just because it is in some vehicle (e.g., in a bird, in the water, etc.). Disconnection of nodes may happen suddenly without respect any protocol.

3 Feedback loops

One of the advantages of the proposed approach is that it provides two levels of abstraction to simplify system development. The first one is at the WSN level, in which hardware decisions play an important role for electronic engineers. All the problems related with wireless nodes and communication at this level are not propagated to other levels in the design. The second level is at the high level computing interface, which consists of a *p2p* network that allows to process the information collected at the WSN level, possibly by taking actions. One purpose of this abstraction is narrowed towards unattended operation. The initiative is to design a self-managing system capable to maintain itself despite environment changes by using feedback loops.

Feedback loops allow to model different kinds of systems. They have been introduced in [5] as a way to model software and, in particular, self-managing systems. A system description is made up of four parts: (1) a *subsystem*, which is the main component; (2) an agent that *monitors* the subsystem; (3) a *correcting* agent, which depending on the information received from monitor agents can calculate the corrective actions to take if needed; (4) an agent that *applies* the mentioned correcting actions back to the *subsystem*. This abstraction is highly concurrent because each component may run as a separate process or agent. Abstraction in feedback loops is also an example of a loosely-coupled system in which components play a well defined role and can be added or replaced to adapt or change system behavior.

Every component of the feedback loop can be a feedback loop itself. For instance, a simple distributed application modelled as a feedback loop would need a *failure detector* as the monitoring component. When the monitor triggers a failure event, the application will need to trigger a corrective action to tolerate the failure and continue working. The failure detector can also be modelled as a feedback loop. First of all, the detector needs a *keep alive* message. In order to do this, a *ping* message is sent to the other node participating in the communication session. This *ping* message is one actuator of the loop. Once the message is sent, the detector waits for a *pong* message that will confirm that the other node is alive. If no *pong* message is received within a certain amount of time, a timeout will be triggered, which is also monitored by the system. According to the policy of the failure detector, two things could happen. A new *ping* message can be sent as a reaction to the timeout, or a failure event can be triggered to the larger loop, being the distributed application.

We believe that feedback loops constitute a convenient way of describing software systems at different levels of abstraction. Such levels favor simplifications both on the reasoning about distributed systems and in their development. This is because we can think of every model as an interaction between components that can be interchanged or specialized and affect the behavior of the system. Interestingly, feedback loops can be used to design the WSN networks but also to design the *p2p* network. In fact, they can be used to model from different message passing protocols to failure detector agents.

4 Case Study

An interesting application for the proposed programming abstraction arises in agriculture. Cultivating crops requires special conditions such as temperature, humidity of the ground, etc. Figure 2 shows a diagram of an extensive land with crops monitored by WSN. Suppose sensors at every zone are continuously monitoring air and ground humidity (*sensing phase*). When ground becomes too dry it is necessary to start water irrigation. However, if in a near zone the conditions of pressure and air humidity indicates high chances of rain, then, irrigation can be delayed in order to save resources.

To decide whether to start irrigation or not, it is necessary to have information from adjacent zones. In this way, the node needing this information will communicate with the others peers to ask for it. The *p2p* network will deliver this information to the site needing it. Irrigation is not the most complicate scenario, there can be situations in which all peers should execute some action without exception.

Figure 3 shows a feedback loop for the example described. WSN are in charge of monitoring the agriculture area, providing information about temperature, humidity and other related values such pressure. The information is given to a *p2p* network that analyze the data in order to take a decision using the actuators to affect the state of the subsystem. Such action could involve start water irrigation or change the behaviour of the sensors to monitor different parameters, or the frequency for providing information.

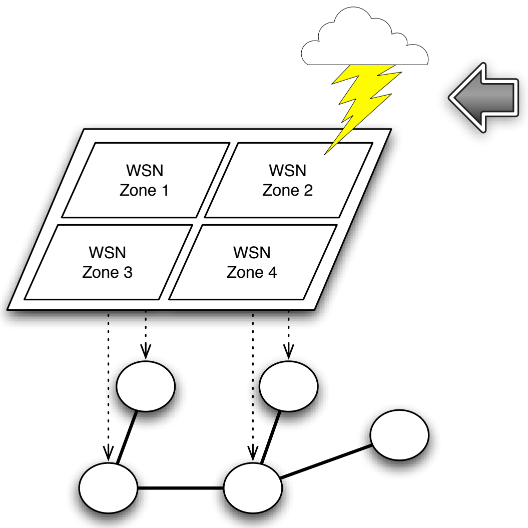

Figure 2. Field with 4 zones monitored by one WSN per zone.

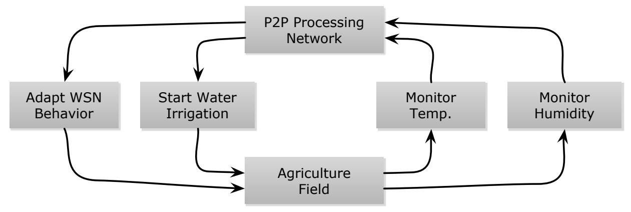

Figure 3. Feedback loop for the system described in Figure 2.

Note that nothing is said about the amount of sensors or peers. As long as the *p2p* network is large enough to offer computer power to process the data provided by the WSN, the system should scale without problem. It should also scales independent of the functionality of the peers. You can add nodes to the *p2p* network just to control how other nodes behave, or to monitor them. This is one of the advantages of *p2p* networks: nodes do not require to share the same, or a given, functionality. This fact encourages the construction of hybrid systems that can be bug prone: the larger the system, the difficulty to maintain it free of errors. However, feedback loops specification of nodes can be adapted (i.e., by adding monitoring and correcting agents) to monitor the behavior of entire components and take the appropriate corrective actions, in this way, we call the result a self managed system.

Our position is that this example can be an starting point to devise the applicability of the proposed abstraction. It is simple and can be used with success. However, this is neither the only possible application nor the most complex or the one that needs every possible feature from the programming abstraction, for other applications you can refer to [4].

5 Conclusions

WSN-based combined with *p2p* systems might lead to the development of robust applications. The use of feedback loops as a design tool make it easier to develop an reason about those, possibly large, resulting systems. A programming abstraction that provides a *p2p* platform to communicate and let developers to take care of functionality is a step towards the simplification of the development of distributed applications. To provide basic communications protocol implementations using feedback loops may help the development of new and possibly more sophisticated implementations by extending the existing ones. We plan to have a prototype of the mentioned abstraction along with the proposed study case implemented, using the Mozart programming system, by the end of this year.

References

[1] D. Culler, D. Estrin, and M. Srivastava. Guest editors' introduction: Overview of sensor networks. *Computer*, 37(8):41–49, 2004.

[2] J. Dedecker, T. Van Cutsem, S. Mostinckx, W. D. Meuter, and T. D'Hondt. Ambienttalk: Language support for mobile computing. *FUMCA '06: Proceedings of the International Workshop on System Support for Future Mobile Computing Applications*, 0:35–42, 2006.

[3] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. In *Ambient Intelligence*. Springer Verlag, 2004.

[4] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, New York, NY, USA, 2002. ACM.

[5] P. Van Roy. Self management and the future of software design. *Electron. Notes Theor. Comput. Sci.*, 182:201–217, 2007.