Adaptable Middleware for Heterogeneous Wireless Sensor Networks

Edison Pignaton Freitas[1,2], Per Söderstam[1], Wagner Ourique de Morais[1], Carlos Eduardo Pereira[2,3], and Tony Larsson[1]
[1] School of Information Science, Computer and Electrical Engineering, Halmstad University, Halmstad, Sweden
[2] Instituto de Informática, Universidade Federal do Rio Grande do Sul, Brazil
[3]Dep. Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Brazil
{edison.pignaton, per.soderstam, wagner.demorais, tony.larsson}@hh.se, cpereira@ece.ufrgs.br

Abstract. The use of sensor networks in different kinds of sophisticated applications is emerging due to several advances in sensor technologies and embedded systems. However, the integration and coordination of heterogeneous sensors is still a challenge, especially when the target application environment is susceptible to changes that the system must track and adapt itself to in order to fulfil the users' requirements. These changing scenarios require services being provided in different places during the system runtime, and to fulfil this, a support for adaptability is needed. In this paper we present some initial ideas to use multi-agents in a middleware that aims to provide the necessary support to sophisticated sensor network applications.

Keywords: Agent-based Adaptable Middleware, Wireless Sensor Networks, Heterogeneous Sensor Networks.

1 Introduction

Sensor network applications are becoming more useful with the possibility to use different kinds of mobile sensors to provide more sophisticated functionalities [1] and be deployed also in complex scenarios, like where context-awareness is needed [2]. However, to support those emerging applications, an underlying infrastructure is necessary, and the current proposal is the use of middleware, such as TinyDB [3] and COUGAR [4]. The main drawbacks of these state-of-the-art middleware's that make them not very suitable for future envisaged applications, are the assumptions that the network is composed only by a homogeneous set of basic or meagre sensors, and thus a lack of the intelligence in the network to provide adaptability required to face changing operation conditions. Adaptability is a major concern that needs to be mainly supported for two reasons. The first is that long deployment time of wireless sensor networks may require flexibility in order to make changes according to the user requirements that may probably change within the usage life time of the network. The second is the fact that wireless sensor networks are being deployed in highly

dynamic environments, implying that applications have to be flexible enough in order to continue being useful in changing scenarios.

This paper presents a work in progress related to the development of an adaptive middleware to support sophisticated sensor network applications that must change their behaviour according to influences from the environment and the application demands. The idea is to use a multi-agent approach to help in the following issues: adaptation by code migration and decision planning.

The remaining of the text is organized as follows: Section 2 presents the overview of the proposed middleware. Section 3 describes how the use of agents will help in providing middleware adaptation. Section 4 presents some related works and finally, section 5 ends the paper with some concluding remarks and future works directions.

2 Overview of the Proposed Approach

The general idea is to develop a flexible middleware that can be used to support applications in heterogeneous sensor networks. By heterogeneity we mean that nodes in the network may have different sensing capabilities, computation power, and communication abilities and running on different hardware and operating system platforms. The goal is that this middleware fits both meagre and rich sensors. Meagre sensors are those with limited resources capabilities, such as piezoelectric resistive tilt sensors, with limited processing support and communication capability. Rich sensors

comprehend powerful devices like radar, cameras or infrared sensors that are supported by moderate to high computing and communication resources. In order to achieve this goal, it must thus be lightweight, while being scalable in order to provide the demands of more sophisticated sensors. The proposed middleware might handle a node's resource usage, in order to assist in distributing tasks among different nodes that are capable to accomplish them. Another feature that the middleware may provide is the quality of the data required to reply a certain user's demand. Better results can be achieved by choosing the correct set of sensors to perform the measurements and collect data. These sensors can be static or moving, on the ground or flying over the target area in which the observed phenomenon is occurring. The mobility characteristic is also related to the heterogeneity that the middleware will address.

The input to the sensor network system, coordinated by the proposed middleware, will be seen as a "mission" that the whole network has to accomplish. In order to allow that, a high-level Mission Description Language (MDL) is being formulated based on the C/ATLAS test language [5]. This language will allow the user to specify - at a high level of abstraction - the data in which he/she is interested, including constraints regarding timing and location limits, as well as the measurement rate or accuracy desired. The proposed language will also allow hierarchical description of the mission goals, with establishment of priorities and other refined details, for instance, an application of comparison metrics to evaluate the how well the mission is being accomplished.

Given that the middleware must perform its actions also in very dynamic and changing scenarios, one has to take into account that a set of sensors chosen in the

beginning of a mission may not be the most adequate during for the whole mission. As an example, an area surveillance system receives the mission to survey an area that may not allow traffic of certain kinds of vehicles. Ground sensors are set to alarm in the presence of undesired vehicles, and unmanned aerial vehicles equipped with visible-light cameras are set to fly to the area where a ground sensor has issued an alarm to verify the occurrence. However, a sudden change in the weather, for instance becoming cloudy and foggy, turns the employment of a visible-light camera useless. The adaptation to this type of change in operational conditions will be supported by the middleware in order to choose a better alternative, among a set of options, for instance by choosing an infrared camera instead.

The middleware is divided in three parts or layers indicating that they are partly using each other in a specific order. Figure 1 presents the overview of the layers of the proposed middleware, and a description of each layer is provided in the following.

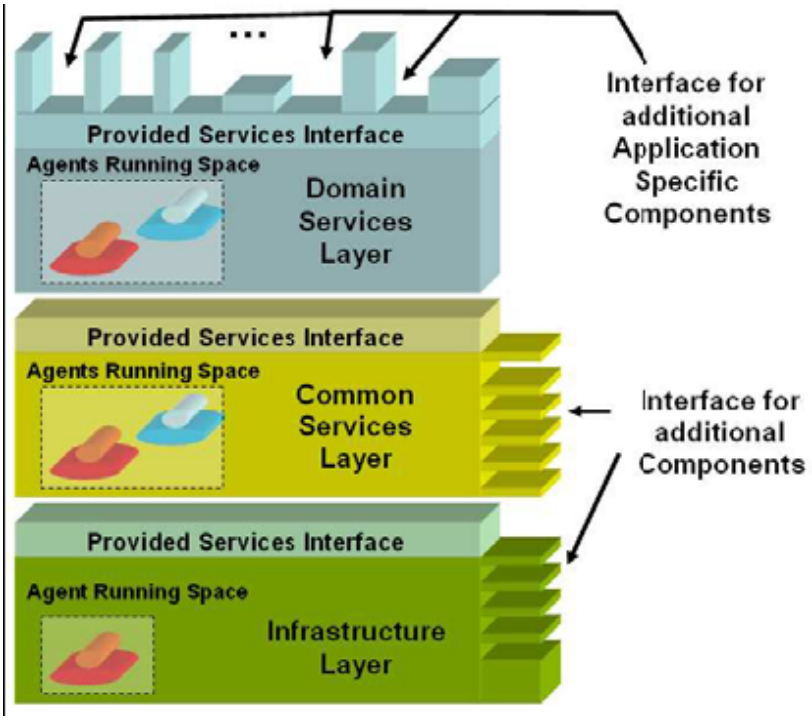

Fig. 1. Overview of the Middleware Layers.

The bottom is called Infrastructure Layer. This layer is responsible for the interaction with the underlying operating system and for the management of the sensor node resources, like available communication capacities, remaining energy, and sensing capabilities. This layer also coordinates the resource sharing based on application needs passed through the upper layers.
The intermediate layer is called Common Services Layer. This layer provides services that are common to different kinds of applications, such as QoS negotiation and control, and quality of data assurance.
The top layer is called Domain-Services Layer and has the goal to support problem-domain specific needs, such as data fusion support and specific data semantic support.
Multiple applications can run concurrently in the network. The middleware handles resource sharing and provides data sharing among applications that need the same type of data, allowing a better energy use in resource constrained nodes. In powerful

nodes, the middleware can provide more complex services aimed to handle rich data, like those related to image processing, and pattern matching. This also means that such nodes can take some of the burden from more resource constrained nodes.

3 Agent-Based Adaption

As stated before, the use of agents in the proposed middleware should help in the task of providing adaptive behaviour. To achieve this goal these agents are divided in two classes. The first is responsible for adaptation features in the services provided by the middleware, using code migration and updating. The second handles reasoning involved in the decision planning concerning the entire network. In the following these two ideas are explained, as well as an example of adaptation is provided.

3.1 Adaptation by Code Migration

As the network are composed by heterogeneous sensor nodes, those nodes that have constraints about their energy consumption, memory space or processing power,

should run a minimum number of tasks as possible. However, as they are supposed to be used in dynamic environments, with changing conditions, requiring different kinds of handling, different task set allocations must be used in order to fulfil the actual needs in a certain time interval. We propose to use the idea of multiple mobile agents to provide services that can be used by the node if the agent is allocated in that node. We call them service-agents. This technique is not new; there are related works like Agilla [6] that use the same approach. However, our approach uses also multi-agents to decide which service-agent is needed in each node of the network at a certain time. We also consider other technologies to help in the middleware adaptation, like aspectorientation and component-based design, although not discussed further in this paper. We refer to [7] for more details about these two technologies.

The service-agents can be allocated in the Common Services Layer or in the Domain Services layer, according to the type of the services they provide. Several service-agents can run in a single node, depending on the computation and memory limitations of the hardware platform. Meagre platform nodes usually need to allocate simpler service-agents, but more powerful nodes can host more sophisticated serviceagents providing more complex services.

Service-agents can migrate from one node to another, clone and move to another node, or simply be unallocated, leaving space for another service-agent. The simple migration is used when its services are needed in other locations, and thus no more in the present one. The clone of a service-agent occurs when its services also are needed in another node, and it is unallocated when there is no node that needs its services. This flexibility of service-agents allows the desired adaptation of the network face changes in the environment that require different runtime services availability.

3.2 Adaptation Planning

We propose the use of a multi-agent approach to distribute intelligence over the network in order to provide a distributed way to decide several key issues about the network setup and configuration during its runtime. As our target applications are heterogeneous wireless sensor networks deployed in changing environments, and so, demanding system adaptations, we need a way to reason about the necessary changes and how to implement them.

The basic idea is that each node attached to the network receives a set of tasks (*Node-Missions*) to perform, and a type of agent, called planning-agent that is in charge of help the node in the accomplishment of its node-missions. A node-mission characterizes the node contribution to the whole system mission achievement, which is called *Global-Mission*. The node's internal tasks are called *Node-Tasks*, and they represent the finer-grained tasks that must be performed by a node to accomplish its node-missions. Node-tasks are related to the individual nodes concerns, for example power consumption handling and sensor capabilities, as well as the management of the services provided by the node.

The planning-agent is responsible for monitoring information about the current node's and assigned node-tasks state during the system runtime, including available resources, e.g. communication bandwidth and energy. As some condition changes in the environment, like when the weather changes or if one kind of measurement is no more possible. For instance, the planning agent has to reason about the node-mission assigned to its node's, and the service-agent(s) that perform the necessary services that the node has to provide. They also have to take in account the available resources to accomplish the node-mission. The planning-agents in all nodes perform this reasoning and, by a consensus, agree in a new distribution of the service-agents in order to accomplish the global-mission assigned to the network or in the employment of a different set of sensors due to a certain change in the network or environment conditions.

3.2 Example of Adaptation

As we stated before, the intention in using both planning-agents and service-agents is to help in the adaption of the system, face changes in the operational conditions due

to environments or user requirements changes. In Figure 2, an example of a change in operation conditions followed by an adaptation is provided.
In the scenario presented in Figure 2, the network receives a mission, which is partitioned in four sub-missions, one for each sensor node. Service-agents are distributed around the sensor nodes according to the initial conditions to accomplish their sub-missions. In the left part of the Figure 2, t_1 represents the initial configuration established for sensor node 1 and 2 to accomplish their respective submissions. However, a change in operating conditions occurs, due to environment or user requirements change. The initial configuration does not meet anymore the system needs to accomplish the mission. So, planning-agents allocated in the nodes exchange information and agree in a new configuration, what occurs in time t_2, in the middle of
21
the Figure 2. After the agreement, a service-agent from node 1 is migrated to node 2 and one service-agent from node 2 is unallocated, as shown in t_3.

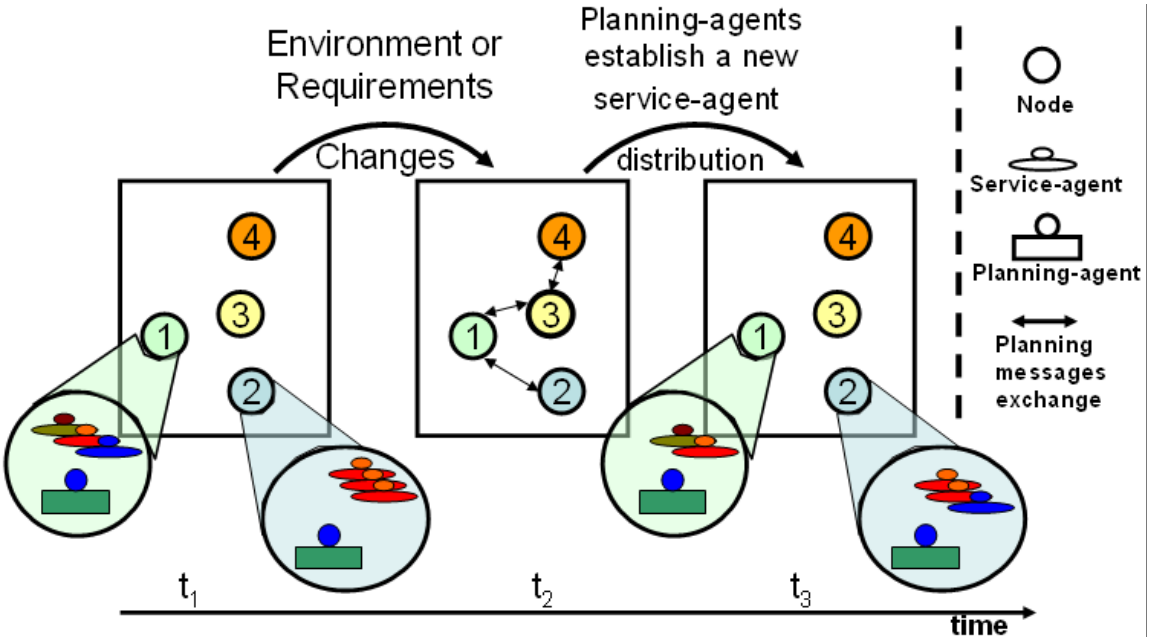

Fig. 2. Changing Scenario and Adaptation Example

4 Related Work

Code replication and migration are ideas that have inspired parts of the present proposal. Some state-of-the-art middleware for sensor networks uses this kind of ideas, as discussed in the following.
Impala [8] is a middleware used in applications dedicated to the study of wildlife. It has some strength in relation to adaptabilities, update and fault-tolerance mostly due to its event-based programming with highly modularized code. The drawbacks of this approach are that it has no support for data fusion and that the domain application is rather simplistic. Comparing with our proposal, we use multi-agents not only to move functionalities within the system, but also to provide intelligent behaviour and reasoning to support mission and environment adaptations.
Agilla [6] is the first mobile agent middleware of WSN implemented in TinyOS [9]. This approach uses agents that can move from one node to another, and it also allows multiple agents to run in the same node. These characteristics provide the desired features of energy saving, as the agents can run near to the data avoiding unnecessary communication, and it allows multiple applications to run concurrently in the network. The major drawbacks are related to the lack of an authentication policy

to monitor agents' activities, and the difficult maintainability due to its programming model. We have some ideas in our work that go close to the ones presented in Agilla. However, the same comments made regarding Impala fits also in a comparison with Agilla. Another drawback is that Agilla runs only over TinyOS, and we propose not to be restricted to one operating system. Agilla use the mobile agents only to run application-specific tasks, while in our approach, a broader idea is proposed, using the agents to provide services that can be more specific to a certain domain application

(being hosted in the Domain Services Layer), or more general, supporting different kind of applications (agents hosted in the Common Services Layer).
Maté [10] consists of a virtual machine which runs atop of TinyOS hiding asynchrony and race conditions. Its strength is represented by the division of the program into small self-replicating capsules that are self-forwarding and selfpropagating. The major drawbacks are that it has high energy consumption and that its programming model is not flexible enough to support a wide range of applications; an update of a capsule can not be done only in a single node, it is spread over the whole network. On the other hand, we propose to support the flexibility to run a broader range of applications, adapting, updating or modifying only specific nodes in the network, and like this also save energy, as only the nodes that need a certain adaptation are reached. This feature also focuses the support for heterogeneity.

5 Concluding Remarks and Future Work

This paper reports about an ongoing project that is in its initial stage. The study of literature in the area of wireless sensor networks indicated that there is a need for research in the area of adaptable middleware for heterogeneous sensor networks [1] [11]. Wireless Sensor Networks - composed both by small and simple resource constrained sensors as well as more sophisticated sensor nodes - need an approach that find the best trade-off to handle the different capabilities in order to provide meaningful information based on the gathered data.
An ongoing study is being performed in order to find the best suited partitioning of services to be hosted in each of the layers of the proposed middleware. In relation to that, the group is also discussing the best possible distribution of functionalities among components and services provided by agents. It is an important issue as we intend to address heterogeneous nodes with different capabilities that can accommodate different sets of services. Another important topic under discussion is the formal definition of the MDL (Mission Description Language). Some directions are under analysis in which a promise one is the use of scripts, such as those used in SensorWare [12], but with a higher abstraction level approach.
However, as this work is in its initial phase, we did not discuss about underlying details like how to achieve consensus among the planning-agents yet. It is certainly a major concern that we have to address during the development of the planning strategy and the methods for system coordination. Related works in this area, such as [13], are being analysed. Besides that, we are aware about the energy issues related to this concern. The message exchange in order to achieve the consensus among planning-agents must be minimal. Another important topic that we are analysing is how to implement the agents running space and the coordination among these spaces. Two ideas considered are the use of tuple space, as used by Agilla and a virtual machine like proposed in Maté [10]. The advantages and drawbacks of each are being studied as well as the search for other alternatives.

References

1. D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," IEEE Computer, vol. 37, no. 8, pp. 41–49, 2004.
2. K. Henricksen and J. Indulska. A software engineering framework for context-aware pervasive computing. In 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom), pages 77–86. IEEE Computer Society, March 2004.
3. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An acquisitional query processing system for sensor networks. ACM Transactions on Database Systems,

30(1):122–173, 2005.
4. P. Bonnet, J. E. Gehrke, and P. Seshadri. Towards sensor database systems. In 2nd International Conference on Mobile Data Management (MDM), volume 1987 of Lecture Notes, 2001.
5. IEEE Std 716-1995, 1995. IEEE standard test language for all systems-Common/Abbreviated Test Language for All Systems (C/ATLAS), IEEE, Inc.
6. C.-L. Fok, G.-C. Roman, and C. Lu, “Rapid development and flexible deployment of adaptive wireless sensor network applications,” in Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS’05), 2005.
7. A. Tesanovic, et al. “Aspects and Components in Real-Time System Development: Towards Reconfigurable and Reusable Sofftware”, Journal of Embedded Computing, IOS Press, v.1, n.1, 2005.
8. T. Liu and M. Martonosi, “Impala: A middleware system for managing autonomic, parallel sensor systems,” in ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2003.
9. TinyOS website. [Online]. Available: http://webs.cs.berkeley.edu/tos/
10. P. Levis and D. Culler, “Maté: A tiny virtual machine for sensor networks,” in International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, USA, Oct. 2002. [Online]. Available: citeseer.ist.psu.edu/levis02mate.html
11.K. Romer, O. Kasten, and F. Mattern, “Middleware challenges for wireless sensor networks,” ACM SIGMOBILE Mobile Communication and Communications Review, vol. 6, no. 2, 2002.
12. A. Boulis, C.-C. Han, and M. B. Srivastava. Design and implementation of a framework for efficient and programmable sensor networks. In MobiSys ’03: Proceedings of the 1st international conference on Mobile systems, applications and services, pages 187–200, New York, NY, USA, 2003. ACM Press.
13. Y. Elmaliach, N. Agmon and G. Kaminka, “Multi-Robot Area Patrol under Frequency Constraints”, in Proc. of 2007 IEEE International Conference on Robotics and Automation, IEEE Computer Society, 2007, pp. 385-390.

Acknowledgments

We would like to acknowledge Kristoffer Lidström and Andreas Persson for the value contributions in discussions about this paper.