# Data centric wireless sensor networks

S. Dulman and P. Havinga

Computer Architecture Design and Test for Embedded Systems Department
University of Twente, the Netherlands
{dulman, havinga}@cs.utwente.nl

## ABSTRACT

While having the simple task of gathering the most basic information, wireless sensor networks can pose very complex design challenges because of the limited quantity of resources available. The (often too complex) protocols needed to assure the quality and the amount of needed data are usually hard to implement in the target hardware.

The aim of this paper is to present a new architecture more suited for the wireless sensor networks than what is already available. The new architecture is designed to suit the dynamic environments in which these systems will be deployed.

***Keywords***: sensor networks; data-centric architecture

## 1 Introduction

Embedded systems have gained a lot of attention lately. Due to technological advances, building small-sized, energy-efficient reliable devices, capable of communicating with each other and organizing themselves in ad-hoc networks has become possible. These devices have brought a new perspective to the world of computers as we know it, pushing us into what can be called the third era of computing: intelligent devices can be embedded into the environment, assisting the user in performing various tasks while being invisible to him. The need for reconfiguration and maintenance disappears as the networks organize themselves to adapt to the continuously changing environment and requirements.

While having the simple task of gathering the most basic information, wireless sensor networks can pose very complex design challenges because of the limited quantity of resources available. Although sensor nodes will be equipped with a power supply (battery) and an embedded processor that makes them autonomous and self-aware, their functionality and capabilities will be very limited. Therefore, collaboration between nodes is essential to deliver smart services in a ubiquitous setting. New algorithms for networking and distributed collaboration need to be developed. These algorithms will be key for building self-organizing and collaborative sensor networks that show emergent behavior and can operate in a challenging environment where nodes move, fail and energy is a scarce resource. The question that rises is how to organize the internal software and hardware components in a manner that will allow them to work properly and be able to adapt dynamically to new environments, requirements and applications. At the same time the solution should be general enough to be suited for as many applications as possible.

This paper presents a new architecture to cope with the dynamic environment in which these systems are deployed. The proposed architecture was the foundation for a novel operating system and a simulation framework for sensor networks, and was shown to be able to support basic building blocks for such systems (as media access control, routing, position finding, timing and synchronization algorithms). The theoretical results are confirmed in practice by several working prototypes.

The paper is organized as follows: Section 2 motivates the need for a new architecture. The proposed architecture is presented in Section 3. The paper finishes with conclusions and directions for future work. The work described in this paper was partly supported by the EYES European Project (IST-2001-34734) [1].

## 2 Motivation

A large amount of effort has been put into the field of sensor networks lately. Large numbers of prototypes have already been developed and deployed, and various numbers of architectures and operating systems have been and still are under research (e.g. the TinyOs related projects [2]).

Before discussing the arguments for developing a new architecture for wireless sensor networks, let us take a look at the scenarios where these devices will fit in. The authors of [3] suggest a classification based on the complexity of the network involved (they distinguish between: small scale applications - intelligent warehouse, medium-large scale applications - environmental monitoring and very large scale applications - a city scenario). On the other hand, the authors of [4] present a classification of sensor networks based on their area of application. It takes into consideration only the military, environment[5], health[6], home[7] and other commercial areas and can be extended with additional categories such as space exploration, chemical processing, etc.

## 2.1 Diversity and dynamics

The sensor networks are dynamic from many points of view. Continuously changing behaviors can be noticed in several aspects of sensor networks, some of them being:

- *Sensing process* - the natural environment is dynamic by all means (the basic purpose of sensor networks is to detect, measure and alert the user of the changing of its parameters). The sensor modules themselves can become less accurate, need calibration or even break down.
- *Network topology* - one of the features of the sensor networks is their continuously changing topology. There are a lot of factors contributing to this, such as: failures of nodes or the unreliable communication channel, mobility of the nodes, variations of the transmission ranges, clusters reconfiguration, addition/removal of sensor nodes, etc.
- *Available services* - mobility of nodes, failures or availability of certain kinds of nodes might trigger reconfigurations inside the sensor network. The functionality of nodes may depend on existing services at certain moments and when they are no longer available, the nodes will either reconfigure themselves or try to provide them themselves.
- *Network structure* - new kinds of nodes may be added to the network. Their different and increased capabilities will bring changes to the regular way in which the network functions. Software modules might be improved or completely new software functionality might be implemented and deployed in the sensor nodes.

Most wireless sensor network architectures currently use a fixed layered structure for the protocol stack in each node. This approach has certain disadvantages for wireless sensor networks. Some of them are:

- *Dynamic environment* - sensor nodes address a dynamic environment where nodes have to reconfigure themselves to adapt to the changes. Since resources are very limited, reconfiguration is also needed in order to establish an efficient system (a totally new functionality might have to be used if energy levels drop under certain values). The network can adapt its functionality to a new situation, in order to lower the use of the scarce energy and memory resources, while maintaining the integrity of its operation.
- *Error control* - it normally resides in all protocol layers so that for all layers the worst case scenario is covered. For a wireless sensor network this redundancy might be too expensive. Adopting a central view on how error control is performed and cross-layer design will reduce the resources spent for error control.
- *Power control* - it is traditionally done only at the physical layer, but since energy consumption in sensor nodes is a major design constraint, it is found in all layers (physical, data-link, network, transport and application layer).
- *Protocol place in the sensor node architecture* - an issue arises when trying to place certain layers in the protocol stack. Examples may include: timing and synchronization, localization and calibration. These protocols might shift their place in the protocol stack as soon as their transient phase is over. The data produced by some of these algorithms might make a different protocol stack more suited for the sensor node (eg. a localization algorithm for static sensor networks might enable a better routing algorithm that uses information about the location of the routed data destination).
- *Protocol availability* - new protocols might become available after the network deployment or at certain moments, in specific conditions, some of the sensor nodes might use a different protocol stack that better suits their goal and the environment.

These examples already suggest that dynamic reconfiguration of each protocol as well as dynamic reconfiguration of the active protocol stack is needed.

## 3 Architecture description

The system we are trying to model is an event-driven system, meaning that it reacts and processes the incoming events and afterwards, in the absence of these stimuli, it spends its time in the sleep state (the software components running inside the sensor node are not allowed to perform blocking waiting).

Let us name a higher level of abstraction for the event class as *data*. Data may encapsulate the information provided by one or more events, have a unique name and contain additional information such as deadlines,

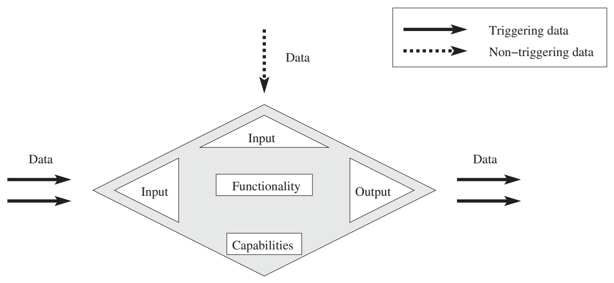

Figure 1: Entity description

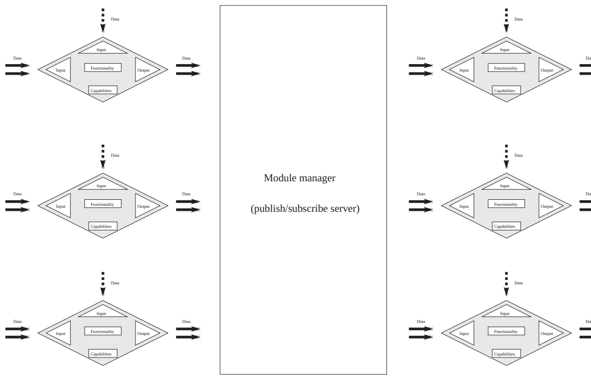

Figure 2: Data-centric architecture

identity of producer, etc. Data will be the means used by the internal mechanisms of the architecture to exchange information components.

In the following we will address any protocol or algorithm that can run inside a sensor node with the term *entity* (see Figure 2). An entity is a software component that will be triggered by the availability of one or more data types. While running, each entity is allowed to read available data types (but not wait for additional data types becoming available). As a result of the processing, each software component can produce one or more types of data (usually on their exit).

An entity is also characterized by some *functionality*, meaning the sort of operation it can produce on the input data. Based on their functionality, the entities can be classified as being part of a certain protocol layer as in the previous description. For one given functionality, several entities might exist inside a sensor node; to discern among them, one should take into consideration their *capabilities*. By capability we understand high-level description containing the cost for a specific entity to perform its functionality (as energy, resources, time, etc.) and some characteristics indicating the estimated performance and quality of the algorithm.

In order for a set of components to work together, the way in which they have to be interconnected should be specified. The architectures existent up to this moment in the wireless sensor network field, assume a fixed way in which these components can be connected, which is defined at compile time (except for the architectures that for example allow execution of agents). To change the protocol stack in such an architecture, the user should download the whole compiled code into the sensor node (via the wireless interface) and then make use of some boot code to replace the old running code in it. In the proposed architecture we are allowing this interconnection to be changed at run time, thus making on-line update of the code possible, the selection of a more suited entity to perform some functionality based on the changes in the environment, etc. (in one word allowing the architecture to become dynamically reconfigurable).

To make this mechanism work, a new entity needs to be implemented; let us call this the *data manager*. The data manager will monitor the different kinds of data being available and will coordinate the data flow inside the sensor node. At the same time it will select the most fitted entities to perform the work and it will even be allowed to change the whole functionality of the sensor node based on the available entities and external environment (see Figure 3).

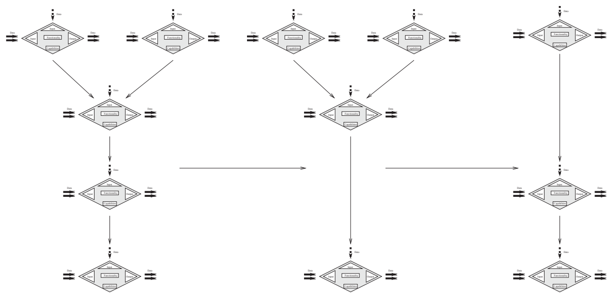

Figure 3: Architecture transitions

The implementation of these concepts can not make abstraction of the small amount of resources each sensor node has (as energy, memory, computation power, etc.). Going down from the abstraction level to the point where the device is actually working, a compulsory step is implementing the envisioned architecture in a particular operating system (in this case maybe a better term is system software). A large range of operating systems exist for embedded systems in general [8][9]. Scaled down versions with simple schedulers and limited functionality have been developed especially for wireless sensor networks [10].

Usually, the issues of system architecture and operating system are treated separately, both of them trying to be as general as possible and to cover all the possible application cases. A simplistic view of a running operating system is a scheduler that manages the available resources and coordinates the execution of a set of tasks. This operation is centralized from the point of view of the scheduler that is allowed to take all the decisions. Our architecture can also be regarded as a centralized system, with the data manager coordinating the data flow of the other entities. To obtain the smallest overhead possible there should be a correlation between the function of the central nucleus from our architecture and the function of the scheduler from the operating system. This is why we propose a close relationship between the two concepts by extending the functionality of the scheduler with the functionality of the data manager. The main challenges that arise are keeping the size of the code low and the context-switching time.

## 3.1 Requirements

As we mentioned earlier, the general concept of *data* is used rather than the *event* one. For the decision based on data to work, there are some additional requirements to be met.

First of all, all the modules need to declare the name of the data that will trigger their action, the name of the data they will need to read during their action (this can generically incorporate all the shared resources in the system) and the name of the data they will produce. The scheduler needs all this information to take the decisions.

From the point of view of the operating system, a new component that takes care of all the data exchange needs to be implemented. This would in fact be an extended message passing mechanism, with the added feature of notifying the scheduler when new data types become available. The mapping of this module in the architecture is the constraint imposed to the protocols to send/receive data via, for example, a publish/subscribe mechanism to the central scheduler.

An efficient naming system for the entities and the data is needed. Downloading new entities to a sensor node involves issues similar to services discovery. Several entities with the same functionality but with different requirements and capabilities might co-exist. The data centric scheduler has to make the decision which one is the best.

## 3.2 Extension of the architecture

The architecture presented earlier might be extended to groups of sensor nodes. Several Data Centric Schedulers together with a small, fixed number of protocols can communicate with each other and form a virtual backbone of the network. Entities running inside sensor nodes can be activated using data types that become available at other sensor nodes (for example, imagine one node using his neighbor routing entity because it needs the memory to process some other data). Of course, this approach raises new challenges. A naming system for the functionalities and data types, reliability issues of the system (for factors such as mobility, communication failures, node failures, security attacks) are just a few examples. Related work on these topics already exists (for example: [11][12]).

## 4 Conclusions

In this paper we have outlined the characteristics of wireless sensor networks from an architectural point of view. As sensor networks are designed for specific applications, there is no precise architecture to fit them all but rather a common set of characteristics that can be taken as a starting point.

The combination of the data centric features of sensor networks and the need to have a dynamic reconfigurable structure has led to a new architecture that provides enhanced capabilities than the existing ones. The new architecture characteristics and implementation issues have been discussed, laying the foundations for future work. The presented architecture has already been used in practice, the DCOS operating system [13] being designed on top of it.

This area of research is currently in its infancy and major steps are required in the fields of communication protocols, data processing and application support to make the vision of Mark Weiser a reality.

## REFERENCES

[1] Eyes: (http://eyes.eu.org)

[2] TinyOS: (http://www.tinyos.net)

[3] Estrin, D., Govindan, R., Heidemann, J., Kumar, S.: Next Century Challenges: Scalable Coordination in Sensor Networks. (1999)

[4] Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks (Elsevier) Journal **38** (2002) 393–422

[5] Polastre, J., Szewczyk, R., Culler, D.: Analysis of Wireless Sensor Networks for Habitat Monitoring. In: Wireless sensor networks. Kluwer Academic Publishers (2004)

[6] Society, I.C.S.: Pervasive computing vol. 3 nr. 2 - successfull aging (2004)

[7] T.Basten, M.Geilen, Groot, H.: Omnia fieri possent. In: Ambient Intelligence: Impact on embedded system design. Kluwer Academic Publishers (2003) 1–8

[8] VxWorks: (http://www.windriver.com)

[9] Salvo: (http://www.pumpkininc.com)

[10] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D.E., Pister, K.S.J.: System architecture directions for networked sensors. In: Architectural Support for Programming Languages and Operating Systems. (2000) 93–104

[11] P.Verissimo, Casimiro, A.: Event-driven support of real-time sentient objects. In: Proceedings of WORDS 2003. (2003)

[12] Cheong, E., Liebman, J., Liu, J., Zhao, F.: Tinygals: a programming model for event-driven embedded systems. In: Proceedings of the 2003 ACM symposium on Applied computing, ACM Press (2003) 698–704

[13] Hofmeijer, T., Dulman, S., Jansen, P., Havinga, P.: Dcos, a real-time light-weight data centric operating system. In: IASTED International Conference on Advances in Computer Science and Technology. (2004)