Service-Oriented Software Architecture for Sensor Networks

Frank Golatowski, Jan Blumenthal, Matthias Handy, Marc Haase,
Hagen Burchardt, Dirk Timmermann
Institute of Applied Microelectronics and Computer Science
University of Rostock
Richard-Wagner-Str. 31, 18119 Rostock, Germany
{frank.golatowski, jan.blumenthal, matthias.handy, marc.haase, hagen.burchardt,
dirk.timmermann}@etechnik.uni-rostock.de

Abstract

This article describes the new concept of service oriented software architecture for mobile sensor networks. By the use of this architecture, a flexible, scalable programming of applications based on an adaptive middleware is possible. The middleware supports mechanisms for cooperative data mining, self-organization, networking, and energy optimization to build higher-level service structures. The development of applications speeds up and is simplified by the use of optional administration components.

1 Introduction

In the future, the increasing miniaturization of electronic components and advances in modern communication technology make the development of powerful spontaneously networked and mobile systems possible. In the next 15 years, *wireless sensor networks* have an enormous economical potential, if the research issues stated in this work can be solved satisfactorily.

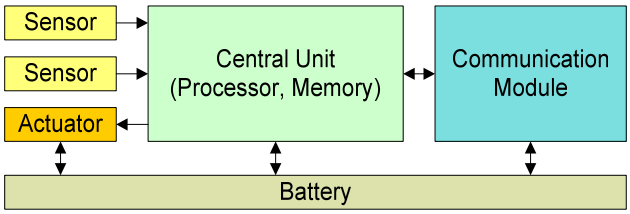

Figure 1: Structure of a Sensor Node

Sensor networks consist of a huge number of small *sensor nodes*, which communicate wirelessly. These sensor nodes can be spread out in hard accessible areas by what new applications fields can be pointed out.
A sensor node combines the abilities to compute, communicate and sense. The aim is to fit all mentioned features in a one single chip solution. In principle, controlling of an *actuator* is possible, too. Figure 1 shows the structure of a sensor node.
The development of sensor nodes is influenced by

- increasing device complexity on microchips,
- high performance, wireless networking technologies,
- a combination of digital signal processing and sensor data acquisition,
- progress within the development of micro-electromechanical systems (MEMS), and
- availability of high performance development tools.

The use of energy-efficient components and procedures in sensor nodes is important for the life time of network nodes. The application fields of sensor nodes are effected by processor

performance, transmission range, radio sensitivity, power consumption, weight, and size. Possible applications are medical monitoring of different health parameters (intra and extra-corporal), environment monitoring, control of industrial machines and devices.
This paper is organized as follows. The requirements of sensor networks are discussed in detail in Section 2. Section 3 explains aspects of software development and points out existing solutions. In Section 4, a new concept of a service-oriented software architecture for sensor networks is presented. Section 5 concludes this paper with an outlook on future research activities.

2 Sensor Networks Requirements

Besides application-specific tasks of a node, the entire network requires a conformance to dynamical system requirements. The development focus changes from the single result of a sensor node to the cumulative result of the network. Consequentially, the following requirements for the design and implementation process of sensor networks arise:

1. Sensor networks have to be **self-organizing**.
2. Sensor nodes must perform tasks of network maintainance.
3. **Cooperative processing** of tasks should lead to more precise results and new application fields.
4. Sensor networks require **security mechanisms** that are adaptive to environmental conditions.
5. All algorithms and protocols must be **energy optimized**.

These requirements are described in detail in the following subsections.

2.1 Self Organization

The enormous number of nodes in sensor networks requires sophisticated solutions for the automatic organization of the network. A manual boot procedure by the administrator is nearly impossible. Therefore, the software of the node has to autonomously setup an operating network infrastructure by interaction with the neighboring nodes. For self-organizing networks, the knowledge of the current context (*context awareness*) is important. During the organization phase, the infrastructure context (perception of the network bandwidth and reliability) and the domain context (relations between the network participants) are primarily important. The current system context, e.g. the knowledge of data sinks of the system, is necessary for a sensor node to operate correctly. This context can change permanently because of the mobility of nodes, so therefore update mechanisms have to be considered.

2.2 Network Functionality

Software development for wireless sensor networks requires novel programming paradigms and technologies. Conventional principles of communication are mostly inapplicable due to the dynamic topology and the need for cooperative task processing in sensor networks.
Traditional wired networks are usually based on the client-server-principle, with the client sending a request and the server replying. However, communication in sensor networks should be event based: Exceeding a threshold value at a specific node triggers an event that is forwarded to data sinks. Contrary to conventional communication via *request-reply*, energy-dissipating polling can be avoided [5].
Node addressing in wireless sensor networks differs from the approach in wired networks. For sensor networks, explicit node addressing via ID or IP is unfavorable since random node distribution and mobility impedes assignment of node addresses to the position of a measurement. Since we do not need to communicate with a specific node described by its

address, we use an addressing mechanism called *attribute-based-naming* [6]. A satisfying request processing requires a node's position and its context. An example for a network request could be: „What is the temperature at location (x,y)?".
In large sensor networks, packets have to be routed from data sources to data sinks over intermediate nodes, i.e. besides the measurement task, all nodes have to perform additional tasks to maintain network integrity. These additional tasks can considerably affect the lifetime and functionality of a sensor node. For the implementation of sensor network software, special techniques have to be used to find trade-offs between task processing, network functionality and lifetime of nodes.
A potential approach could be the strategy of *communication avoidance* resp. *communication depletion*, e.g. context-dependent routing. For static networks, a proactive routing algorithm is applicable, i.e. routes are discovered prophylactically without a request for a specific route. A network with mobile nodes needs different algorithms since routes change frequently. Hence, reactive routing protocols are used for dynamic network topologies. Routes are only discovered on demand. An optimal routing algorithm for wireless sensor networks should adapt its strategy to the mobility of nodes.

2.3 Cooperative Algorithms

Compared to networked macrosensors, an advantage of wireless sensor networks is the possibility to implement cooperative algorithms. A potential application for these algorithms is the reduction of network traffic by data preprocessing and aggregation. For a sensor application, it is not important whether a data aggregation is performed within the node itself or by a neighboring node. However, communication directed to data sinks has to be minimized since data sinks are usually located far away. An example for a cooperative algorithm is *location determination* by triangulation. This algorithm needs at least measurements from three different nodes. Computed positions then can be used for addressing or routing.

2.4 Security Mechanisms

The selection of proper security mechanisms for wireless sensor networks depends on the network application and the environmental conditions. Additionally, the resources of sensor nodes (processor performance, memory capacity and energy) have to be taken into account.
Besides the standard security requirements, like availability, confidentiality, integrity, authentication, and non-repudiation, special security requirements for wireless sensor networks like message freshness, intrusion detection, intrusion tolerance, or containment exist.
To fulfill the safety requests mentioned above, adapted security mechanisms for wireless sensor networks have been developed and tested by several research groups [13]. The task of the middleware is the selection and the management of security mechanisms dependent on the security policies defined by the operator of the sensor network.

2.5 Low-Power Approach

Sensor nodes are typically battery-driven, however too small and too numerous to replace or recharge batteries. Moreover, microsensor networks are often deployed in remote or dangerous environment. Hence, the increase of the lifetime of sensor nodes is a main design and implementation challenge.
Microcontroller hardware used for sensor nodes provides manifold power saving techniques. One of these is Dynamic Power Management (DPM). DPM switches off hardware components that are not needed and uses clock scaling.

An Operating System (OS) for sensor nodes should implement a low-power task-scheduling. A scheduling algorithm for example could take advantage of nonlinear battery effects to reduce energy consumption [15]. Furthermore, a proper selection of communication protocols additionally reduces energy consumption. The most promising layers for energy savings are physical-, link- and network layer.

3 Software Engineering

For the development of application software for sensor networks, the use of a component based framework is desirable. The components of the framework provide the functionality of single sensors, sensor nodes and the whole sensor network. According to these components, applications are classified into *sensor applications*, *node applications* and *network applications*.

The *sensor application* contains the entire measurement and readout of a sensor as well as the local storage of the data. It has full access to the hardware and is able to access the operating system directly. Network access is not possible. The sensor application provides essential basic functions of the local sensor node, that may be used by the sensor node application.

The *node application* contains all application specific tasks and functions of the middleware to build up and maintain the network, e.g., routing, looking for nodes, discovering services, and self localization.

The *sensor network application* describes the main tasks and required services of the entire network without assigning any tasks or services to individual nodes. It represents an interface to the administrator to evaluate the network results.

The purpose of our research activities is the development of a framework, which radically simplifies the development of software for sensor, sensor node, and sensor network applications. It provides support for distribution, configuration, scalability and portability. The framework should contain a middleware that allows machine-intimate programming of embedded systems. Thereby, the programming effort is reduced.

3.1 Current Software Architectures of Small Distributed Devices with Wireless Network Connection

For embedded systems, solutions already exist, which support service architectures and context awareness. In [11] the distributed middleware infrastructure GAIA is presented. It features coordination of software units and heterogeneous networks. The network appears to the environment as a single enclosed device. Because of the use of CORBA, XML, SQL and JAVA, GAIA is not an efficient choice for sensor networks because of its resource requirements.

Another approach is Tiny-OS that already provides an advanced framework for sensor networks [3]. It is optimized in terms of memory usage and energy efficiency. Tiny-OS [12] provides mechanisms (events and components) to define statically linking between the layers. The predefinition of the needed instances at compile time prevents dynamical memory allocation at runtime which achieves additional advances in speed. Tiny-OS supports the execution of multiple threads and provides a variety of additional extensions like the virtual machine Maté [2] and the database TinyDB for cooperative data acquisition. Services are currently not provided. Programs for Maté are precompiled and, because of the minimal instruction set, very short. They can be carried in a single Tiny-OS data packet of a maximum length of 24 bytes. Thus, Maté allows a dynamical adaptation of the node application at runtime.

The open and platform independent architecture OSGI is suggested in [10] for services in embedded systems. However, OSGI is not appropriate for sensor networks due to its very

high resource demands. Therefore, it is mandatory to develop a service architecture that features minimal resource consumption.
As a general rule, services have, at least partly, to be executed locally to start communication to the service provider. The execution can take place in native code or in an adapted virtual machine.

3.2 Characteristics of a Middleware for Sensor Networks

The term *middleware* refers to the software layer between operating system and sensor application on the one hand and the distributed application which interacts over the network on the other hand. The primary objective of the middleware layer is to hide the complexity of the network environment by isolating the application from protocol handling, memory management, network functionality and parallelism [14]. A middleware for sensor networks has to be:

- scalable
- generic
- adaptive
- reflective

The resource constraints (memory, processing speed, bandwidth) of available node hardware requires the optimization of every node application. The optimization is performed at compile time. Thereby, the application is reduced to all essential components and datatypes as well as interfaces are customized (*scalable middleware*).
The components of the middleware require a generic interface in order to minimize the customization effort for other applications or nodes. The use of identical middleware components in different applications leads to a higher number of complex interfaces. Reducing this overhead is the objective of a *generic middleware*. It is important to customize the interfaces to the application in contrast to customize the application to common interfaces. As example, a middleware function SetBaudrate(int transmitter, long baudrate) identifies the network interface with the first parameter. However, a node that has only one interface, does not need this parameter. Consequently, the knowledge of this information at compile time can be used for optimization.
Another possibility is to change the semantics of data types. A potential use case is the definition of accuracy of addresses that results in a change of data type's width. The width of a data type has vital influence on the network traffic. Besides hardware-oriented optimization, an application specific data type optimization exists.
The mobility of nodes and changes in the infrastructure requires adaptations of the middleware at runtime depending on the sensor network application. The middleware must be able to dynamically exchange and run components (*adaptive middleware*).
Reflection covers the ability of a system to understand and influence itself. A reflective system is able to present its own behaviour. Thereby, two essential mechanisms are distinguished – the inspection and the adaptation of the own behaviour [1][4]. Inspection covers ways to analyze the behavior, e.g., with debugging or logging. The adaptation allows modifying the internal layers to change the behaviour presented to the application. In contrast to an adaptive middleware, a *reflective middleware* does not exchange components but changes the behaviour of some components. An example of reflective behavior is the modification of the routing strategy depending on mobility. The interface between the software layers remains constant.

3.3 Services in Sensor Networks

Besides the native network functions, such as routing and packet forwarding, future service architectures are required enabling location and utilization of services. A service is a program which can be accessed about standardized functions over a network. Services allow a cascading without previous knowledge of each other, and thus enables the solution of complex tasks.
A typical service used during the initialization of a node is the localization of a data sink for sensor data. Gateways or neighboring nodes can provide this service. To find this service, the node uses a service discovery protocol.
JINI is an emerging technology for desktop applications, but for sensor networks unsuitable due to resource requirements. Sun Microsystems suggests the surrogate host architecture for embedded systems [9]. This is primarily suitable for systems that are controlled by an IP based network. The client can access non-standardized services in a sensor network by inquiring a proxy server. The surrogate host translates the standardized protocol to the proprietary protcol and vice versa. It acts as the service provider to the IP based network. Service architectures for sensor networks are part of the sensor application and operate in contrast to the event-driven node application on the client-server principle.

4 Concept of a Service-oriented Software-Architecture for Wireless Sensor Networks

Figure 2 shows an example of a simple service architecture applicable to a sensor network. In this special case, the client wants to acquire information about the surface conditions in the area of interest. First, the client requests the surrogate proxy via standardized protocols for the surface profile of a part of the observed area. The proxy communicates with the distributed nodes using a proprietary protocol. The nodes located in the target area try to determine the surface profile using cooperative algorithms and send it to the proxy. The proxy translates the information into standardized protocols and sends them back to the client.

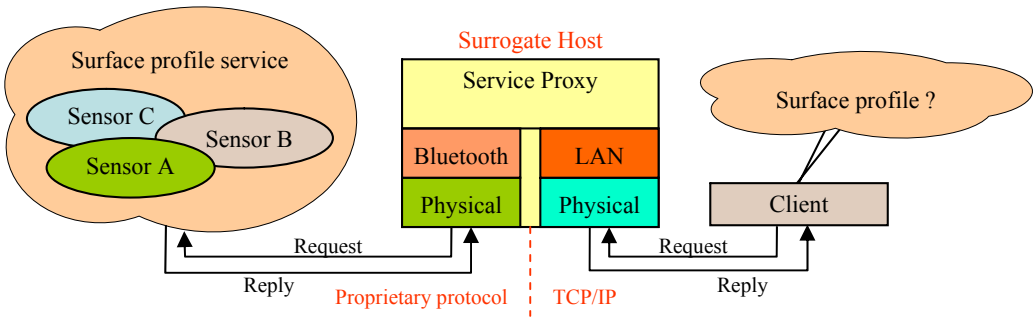

Figure 2: Example of a Surrogate Architecture in Sensor Networks

The different requirements and objectives for sensor networks described in Sections 2 and 3 can be achieved only by using a flexible architecture of the node software. Therefore, a node software is divided into three parts according to the main tasks (Figure 3).
The *Operating System* handles the device-specific tasks. This contains bootup, initialization of the hardware, scheduling, and memory management as well as the process management. The OS consists of special tailored parts only needed by the specific application of the node.
The second part is the *Sensor Driver*. It initalizes the sensor hardware and performs the measurements in the sensor. It encapsulates sensor hardware and provides an optimized Application Programming Interface (API) to the middleware.

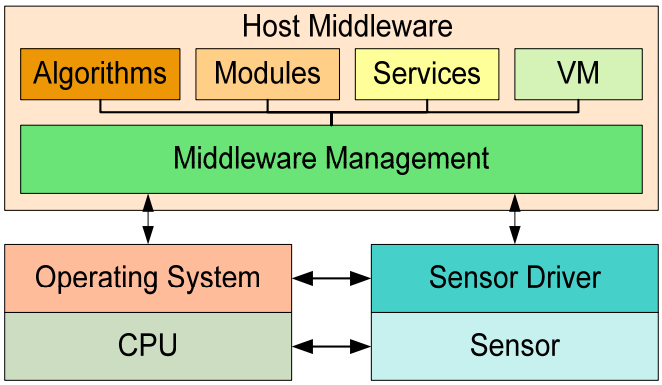

Figure 3: Structure of a Node Application

The *Host Middleware* is the superior software layer. Its main task is to organize the co-operation of the distributed nodes in the network. The *Middleware Management* handles four optional components, which can be implemented and exchanged according to the node's task. *Modules* are additional components that increase the functionality of the middleware. Typical modules are routing modules or security modules. *Algorithms* describe the behavior of modules. For example, the behavior of a security module can vary if the encryption algorithm changes. The *services* component contains the required software to perform local and cooperative services. This component usually cooperates with other nodes to fulfill its task. *Virtual Machines (VM)* enable an execution of platform independent programs.
The software components in a node can be linked together statically or dynamically. Statical linking facilitates an optimization of interfaces between several components within a node. This optimization is called *software scaling*. It performs in faster and smaller programs. The dynamic link process is used for components exchanged during runtime, e.g. algorithms downloaded from other nodes. This procedure results in system-wide interfaces with significant overhead.

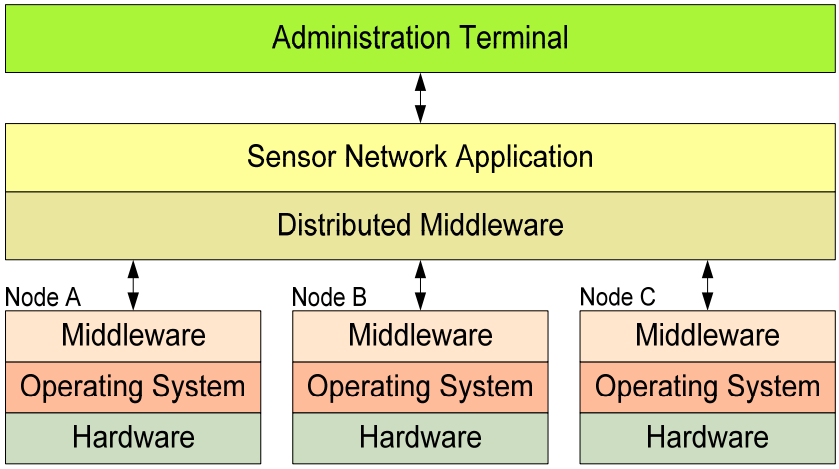

Figure 4: Structure of a Sensor Network

Figure 4 shows the logical view on a sensor network application. The nodes can be contacted only through services of the middleware layers. They do not perform any individual tasks. The *Distributed Middleware* coordinates the cooperation of the services within the network. It is logically located in the network layer but it exists physically in the nodes. All layers together in conjuction with their configuration compose the *sensor network application*.
The *Administration-Terminal* is an external entity to configure the network and evaluate the results. It can be connected to the network at any location.

5 Conclusion

Based on the requirements of sensor networks, this article describes aspects of software engineering. The main objective is the simplification of development of service applications in sensor networks. A key issue is to separate the software from the underlying hardware. The presented service-oriented software concept facilitates the programming on high abstraction layers.

Our current research activities concentrate on the realization of the proposed architecture embedded in a framework. It simplifies the development of sensor-, node-, and sensor network applications. Besides that, it provides functionalities to configure and manage the whole network, whereby the scalability and portability of applications increases.

6 References

[1] G. Coulson, „What is reflective middleware?“, URL: http://dsonline.computer.org/middleware/RMarticle1.htm, DS Online, 2003.

[2] P. Levis, D. Culler, „Maté: a tiny virtual machine for sensor networks“, *Proc. of ACM Conference on Architecture Support for Programming Languages and Operating Systems (ASPLOS)*, Oktober 2002.

[3] J. Hill et al., „System architecture directions for networked sensors“, *Proc. of the Ninth International Conference on Architectural Support for Porgramming Languages and Operating Systems*, November 2000.

[4] P. Meas, „Concepts and experiments in computational reflection“, PhD Thesis, Vrije University Brüssel, 1987.

[5] K. Römer, O. Kasten, F. Mattern, „Middleware challenges for wireless sensor networks“, *Mobile Computing and Communications Review*, Volume 6, Number 2, 2002.

[6] P. Rentala, R. Musunuri, S. Gandham, U. Saxena, „Survey on sensor networks“, *Proc. of International Conference on Mobile Computing and Networking*, 2001.

[7] D. Chess, C. Harrison, A. Kershenbaum, „Mobile agents : are they a good idea ?“, *IBM Research Report*, 1995.

[8] J. Waldo, *„The Jini™ specification“*, 2nd Edition, Addison-Wesley, 2001.

[9] *„The Jini™ technology surrogate architecture overview“*, Sun Microsystems, 2001.

[10] Open Service Gateway Initiative, URL: http://www.osgi.org/about/mission.asp

[11] M. Román et al., *„Gaia : a middleware infrastructure to enable actives spaces“*, Digital Computer Labs, University of Illinois, 2002.

[12] D. Culler, „Tiny OS – a component-based OS for the networked sensor regime“, URL: http://webs.cs.berkeley.edu/tos/, 2003.

[13] A. Perrig, D. Culler et al., „SPINS: Security protocols for sensor networks.”, *Proc. of the of the Seventh Annual International Conference on Mobile Computing and Networking,* Juli 2001.

[14] K. Geihs, „Middleware Challenges Ahead”, IEEE Computer, Juni/2001, S. 24-31.

[15] D. Rakhmatov, S. Vrudhula, C. Chakrabarti, „Battery-concious task sequencing for portable devices including voltage/clock scaling”, *Proc. of the 39th Design and Automation Conference*, New Orleans, 2002.