

# Designing Industry-grade Middleware Services for Wireless Sensor Networks

Martin Fenne, Christian Scholz, and Thomas Wieland

University of Applied Sciences Coburg  
Department of Electrical Engineering and Computer Science  
96450 Coburg, Germany  
{fenne,scholz,wielandt}@fh-coburg.de

**Abstract.** Wireless sensor networks could be used for sensing tasks in industrial automation and process control. For becoming attractive to industrial developers, however, more abstract and more powerful programming models are required to provide standardized and portable system abstractions. This programming model is best realized as a middleware layer shielding the operating system basics and offering advanced services to the application. In this paper, we first identify several decisive requirements for such a middleware in an industrial context. These requirements are then compared with numerous existing middleware approaches that we have analyzed. Finally we propose key design elements based on these considerations that could be guidelines for future investigations on the way to an industrial adoption of sensor networks.

## 1 Introduction

Today wireless sensor networks (WSNs) are in a transition state from research prototypes to practical and commercial usage. A lot of research has been conducted to make those networks more flexible and reliable. The WSN paradigm assumes that a WSN consists of a dozen to hundreds of individual nodes (called "motes"), connected wirelessly, sharing their data in a multi-hop manner.

At the same time there is the huge market of automation and industrial production where a plethora of different sensor types is currently in use. Most of these sensors are wired to a machine or factory floor network; some of them are already connected wirelessly. Of course, the wireless sensors in automation do not follow the WSN paradigm at the moment. Industrial sensors can reside in nodes with own processors and memory, but are typically connected in star or bus topology. Each of them has one designated task, whereas the power of a wireless sensor network is only unleashed in mesh networks combining measurements from several nodes.

The contemporary research activities in WSN focus on gathering and forwarding data, usually allowing just one application for the entire network. In these cases, the protocols and applications are built by the same team, following a common and thus closely coupled design idea. The fundamental services that the TinyOS [1] operating system provides, for

example, are just enough to set up a very simple network. Any more sophisticated routing procedure or energy-saving strategy must be selected individually and added as a module. However, if wireless sensor networks should be attractive for industrial users, an out-of-the-box readiness has to be achieved. We believe that the practical and wide-spread adoption of sensor networks need powerful and robust software services running on the nodes. This software should offer a much more abstract programming model such that the industrial developer can concentrate on his own application, without needing to know about all the hardware and protocol details of the mote.

There are some of the specific characteristics of sensor networks that are less important in industrial contexts. One of them is *mobility*. As production equipment is generally stationary, the sensors are not supposed to move around in a large radius. They may be mounted on a moving part of a machine, but these normally have just a range of a few meters. In some cases, however, like item tracking between production lines, mobility can be an issue.

Factories are well-guarded buildings or sites to which only authorized personnel has access. So *data access control* and encryption on the air link is also of minor importance. Transmissions of automation data must usually not fear hacker interference, apart from acts of sabotage.

Moreover, the production equipment needs lots of electrical power. Hence it is in most cases no problem to provide the sensor nodes with a power supply. The classical concern of mote design, *energy efficiency*, is thus uncritical for most industrial applications.

## 2 Requirements on Middleware Systems

As discussed earlier [2]-[4], middleware systems for wireless sensor networks must meet several functional and non-functional requirements. Most of them are not really specific to WSNs, but apply to any highly distributed computing system. Some applications may do without one or another condition, but as a middleware should as generic as possible, an all-purpose view is desirable.

We list only the most important requirements here, dividing them in three groups. The first group comprises general requirements that a distributed system should fulfill to make it usable in professional environments:

- **Performance:** In general, the middleware should introduce a delay to the overall communication chain that is in the same order of magnitude as the operational performance without this software layer.
- **Adaptability:** The middleware system should be adaptable to changes in applications' and users' requirements or changes within the network (e.g. a different transport protocol).
- **Maintainability:** The middleware shall make it possible to improve it and correct errors or software failures as easy as possible.
- **Scalability:** The system should be scalable to the number of motes and number of users/traffic.

The second group lists some requirements that lead to additional functionality, but are only considered as optional:

- **Location awareness:** It means the ability of the middleware (and the programs running on a particular mote in general) to determine its location. This can be done topologically, i.e. relative to the other nodes of the network, or geographically, i.e. in terms of an absolute geographical position.
- **Addressing:** The middleware should be aware of the addressing scheme employed in the network. If there is not enough support by the underlying base functions or the employed concept is insufficient, the middleware may introduce its own alternative addressing scheme and use it consistently on all participating motes.

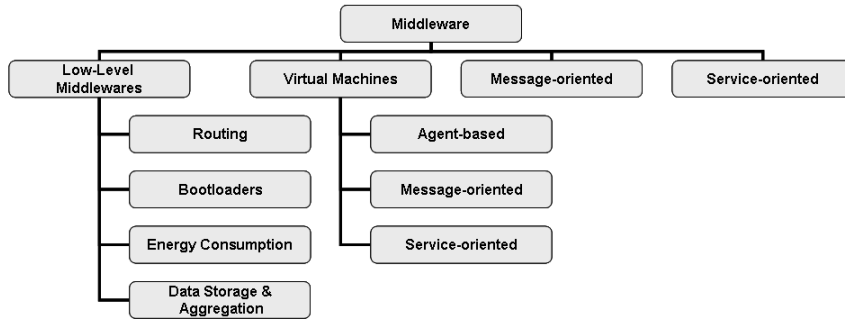
Moreover, there are some requirements that are usually not brought in connection with WSNs, but which we regard as crucial to gain industrial acceptance.

- **Monitoring/Logging:** The middleware should provide services that enable an effective monitoring of the application. This surveillance should be possible in real-time, i.e. during operation, and a posteriori, i.e. after operation by creating and analyzing log files. The monitoring should not be limited to any individual node nor should it require to investigate each node separately; it should instead allow to look at the operation of network as a whole.
- **Updatability:** Installation and replacement of application software over the air, taking the available applications and versions into account. The update should be able to be initiated from a gateway node and to be resumed if the connection to a mote has been interrupted.
- **Task sharing/network partitioning:** The motes in the network should be able to switch from one application to another if necessary. After deployment the network could work as a whole, but should also support partitioning in the sense that some parts of the network perform different tasks than the others.
- **Support of role concept:** Various roles can be assigned to the nodes, either for establishing hierarchies for a more efficient node management and networking (vertical role assignment) or for supporting different tasks of the motes in terms of sensor equipment or software components (horizontal role assignment).

### 3 Middleware Classification

The research in wireless sensor networks led to various approaches of middleware software, but only a few are really applicable. We analyzed more than 50 middleware systems from nearly ten years of research. Most works were only for educational or research purposes, hence they are only publicly available in paper form. We classified the different approaches in four main categories (figure 1).

The family of **low-level** middleware only capsules certain functionalities of the underlying operating system or plays the role of an operating system. These middleware programs simply try to make common functionalities like routing, energy awareness, data storage, as well as aggregation



**Fig. 1.** Categorization of middleware systems.

easier and offer well-defined interfaces. They can be used in nearly any system, but they are very hardware specific. Since they are designed on a low level, the functionality of a higher middleware (e.g. abstraction of objects) never can be reached. Such middleware can also be the lowest tier of a bigger middleware system. Current projects of low-level middleware are for example *Deluge* [5], *CoCo/Mo* [6] and *MoteWorks* from Crossbow Inc.

*Virtual Machines* give the user the ability to replace code on the motes and therefore they are predestined for **agent-based** approaches. Nevertheless message- or service-oriented applications are also possible. The main benefit is the use of an interpreted language, so the code can be improved on a running system. This is a big difference to the low-level approach of the bootloader, where only compiled images can be updated. As mentioned before, an agent-based middleware takes most advantage of this behavior. This architecture is most applicable in homogeneous, but unstable networks with moving nodes, supporting more than one task. By the distribution of the agents, applications can move to the subnetworks where they are needed. *Maté* [7] and its TinyOS port *Bombilla*, developed at UC Berkeley, can be seen as the standard virtual machine for wireless sensor networks. *Agilla* [8] is the agent-based approach of a middleware built upon *Bombilla* and *Deluge*. It is a very active project with first application examples.

The classic **message-oriented** approach is also represented. Here the middleware provides interfaces for the underlying system and message structures. It also can help to do some automated actions like changing routing algorithms. This approach bears some problems within highly distributed or strongly collaborating networks. Here the vast amount of messages that need to be sent can lead quickly to communication blockings. Examples of message-oriented middleware are *Hood* [9] and *Dynamo* [10].

The last family consists of the **service-oriented** middleware. Here the middleware provides functions to publish and subscribe to services. Sensor readings, data storage and aggregation can then be implemented in services to which any mote in need can subscribe. This is a quite con-

servative way of a middleware which has formerly been developed for wired distributed systems. Therefore this approach is well known, but it does not take the whole advantages of a WSN. Static heterogeneous networks would be the best environments for such a middleware. The service-oriented paradigm is followed by *Spatial Views* [11], *TinyLime* [12], *TinyCubus* [13], and the *GREEN* middleware [14] of the RUNES project. Most commercial projects also use this strategy (*Octavex* by Octave Technology, *PLASMA* by IHP, and *eZeeNet* by Meshnetics). There are also some hybrid middleware programs which belong to more than one class providing more functionality and flexibility. Summarizing it can be said, that most of the work is done for research and educational reasons. Only a very very small part has reached a commercial state yet.

## 4 Design Challenges

To design and implement middleware services that fulfill all or most of the requirements stated in sec. 2 is still a lot of work – even when starting with one of the open available systems mentioned above. We sketch some concrete issues here that are crucial for architecture and design (see [3] for related aspects).

- **Partition control and role concept:** To support a flexible network architecture with dynamic task sharing, distributed self-configuring cluster mechanisms are necessary. Each node should be able to switch between several roles, depending on the current network configuration. Protocols for cluster forming have to be developed that activate sleeping nodes, distribute the characteristics of the cluster, and perform "load balancing" in terms of task sharing.
- **Versioning of applications:** For dynamic updates and multiple applications, formal descriptions have to be specified representing the capabilities and status of an individual node.
- **Resource allocation and management:** As each application needs different resources, an automatic way of detecting and reserving available resources like processing power, energy, or memory has to be found. In addition, the partitions/clusters must be able to coordinate each other, as they ultimately might use the same communication resources on their connections to gateways and other central units.
- **Monitoring:** For a precise monitoring, different levels of on-line and off-line surveillance have to be supported on each node. The forwarding of monitoring data must not interfere the network in such an extent that the operation is completely different to the normal case. Moreover, the memory on the nodes is very limited, so the efficiency of the logging has to be very high. Both issues require a lot of theoretical work on protocols and operating system scheduling.

## 5 Conclusion

For leveraging the usage of wireless sensor networks in industrial automation and process control, the development of respective middleware

systems is absolutely necessary. But the requirements arising from this field call for much more abstract, flexible, and adaptive services than available today. The virtual machine model, followed by systems like Bombilla, is a very promising approach for application switching and role models. The challenges in designing industry-grade middleware as mentioned above still mean significant work in solving all conceptual and technical issues before a commercial roll-out of the WSN technology can commence.

## Acknowledgements

The authors want to thank Christoph Niedermeier and his team at Siemens Corporate Technology for financial support and fruitful discussions leading to the results reported here.

## References

1. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D.E., Pister, K.S.J.: System architecture directions for networked sensors. In: ASPLOS, Cambridge, MA (2000) 93–104
2. Römer, K., Kasten, O., Mattern, F.: Middleware challenges for wireless sensor networks. *ACM Mobile Computing and Communication Review* **6**(4) (2002) 59–61
3. Yu, Y., Krishnamachari, B., Prasanna, V.K.: Issues in designing middleware for wireless sensor networks. *IEEE Network Magazine* **18**(1) (2004) 15–21
4. Mascolo, C., Hailes, S., Lymberopoulos, L., Picco, G.P., Costa, P., Blair, G., Okanda, P., Sivaharan, T., Fritsche, W., Karl, M., Rónai, M.A., Fodor, K., Boulis, A.: Survey of middleware for networked embedded systems. Technical report, EU project 'RUNES', IST-004536-RUNES (2005)
5. Hui, J.W., Culler, D.: The dynamic behavior of a data dissemination protocol for network programming at scale. In: Proceedings of the 2nd international conference on Embedded networked sensor systems, ACM Press (2004) 81–94
6. Winter, R.: Coco/mo (communication cooperation/mobility). Technical report, FU Berlin (2006) <http://www.inf.fu-berlin.de/inst/agt-tech/projects/SPP1140/index.htm>.
7. Levis, P., Culler, D.: Maté: A tiny virtual machine for sensor networks. In: ASPLOS. (2002) 85–95
8. Fok, C.L., Roman, G.C., Lu, C.: Mobile agent middleware for sensor networks: An application case study. In: Proc. of the 4th Int. Conf. on Information Processing in Sensor Networks (IPSN'05), IEEE (2005) 382–387
9. Welsh, M., Mainland, G.: Programming sensor networks using abstract regions. In: Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04), San Francisco, CA, USA (2004) 29–42

10. Mohapatra, S.: DYNAMO: Power Aware Middleware for Distributed Mobile Computing. PhD thesis, Donald Bren School of Computer Sciences, University of California, Irvine (2006)
11. Ni, Y., Kremer, U., Iftode, L.: Spatial views: Space-aware programming for networks of embedded systems. In: 16th International Workshop on Languages and Compilers for Parallel Computing, TAMU, College Station (2003)
12. Curino, C., Giani, M., Giorgetta, M., Giusti, A., Murphy, A.L., Picco, G.P.: Tinyline: Bridging mobile and sensor networks through middleware. In: 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005), IEEE Computer Society (2005) 61–72
13. Marrón, P.J., Minder, D., Lachenmann, A., Rothermel, K.: Tinycubus: An adaptive cross-layer framework for sensor networks. *Information Technology* **47**(2) (2005) 87–97
14. Sivaharan, T., Blair, G.S., Coulson, G.: Green: A configurable and re-configurable publish-subscribe middleware for pervasive computing. In: OTM Conferences. (2005) 732–749