Middleware Challenges for Wireless Sensor Networks*

Kay Römer, Oliver Kasten, Friedemann Mattern
Department of Computer Science, ETH Zurich, Switzerland
{roemer,kasten,mattern}@inf.ethz.ch

Middleware for sensor networks aims to support the development of applications for large populations of wirelessly connected nodes capable of computation, communication, and sensing. We examine the purpose, functionality, and characteristics of such middleware.

I. Introduction

Wireless sensor networks (WSN) [1] are an increasingly attractive means to monitor environmental conditions and to bridge the gap between the physical and the virtual world. A WSN consists of large numbers of cooperating small-scale nodes capable of limited computation, wireless communication, and sensing. By correlating sensor output of multiple nodes, the WSN as a whole can provide functionality that an individual node cannot. Application areas for WSNs include geophysical monitoring (seismic activity), precision agriculture (soil management), habitat monitoring (tracking of animal herds), transportation (traffic monitoring), military systems, business processes (supply chain management), and in the future, possibly cooperating smart everyday things.

The basic mode of operation of WSNs is significantly different from traditional computer networks, due to their tight integration with the physical world. Additionally, sensor networks have some unique characteristics that make the development of applications non-trivial. This short paper summarizes our initial investigation of the potential for middleware in wireless sensor networks, as discussed at the Advanced Topic Workshop on Middleware for Mobile Computing. We first give an overview of WSNs and their characteristics, and then discuss middleware issues for WSN.

II. Wireless Sensor Networks

Basic Operation Deployment of a sensor network in a target area can be a continuous process, for example to replace nodes with depleted batteries or nodes that have been destroyed due to environmental influences. In general, deployment establishes an association of sensor nodes with objects, creatures, or places in order to augment them with information-processing capabilities. Deployment can be as diverse as establishing one-to-one relationships by attaching sensor nodes to specific items to be monitored [2], covering an area with locomotive sensor nodes [7], or throwing nodes from an aircraft into an area of interest [18]. Due to their large number, nodes have to operate unattended after deployment.

Once a sufficient number of nodes has been deployed, the sensor network can be used to fulfill its task. This task can be issued by an external entity connected to the sensor network, such as a user with a PDA, an aircraft flying by, or some device on the Internet. Also conceivable are isolated, self-contained sensor networks which are programmed to fulfill a certain sensing task, whose result controls actuator nodes that are also part of the network. In hybrid architectures, the sensing results control the sensors to trigger more detailed monitoring of a certain phenomenon, which is then reported to the external task issuer.

The sensing tasks are usually rather high level, such as "report size, speed and direction of vehicles over 40 tons moving through a certain area". On the other hand, individual sensor nodes typically provide very simple functionality, such as determining movement at a certain place. In order to solve complex high-level sensing tasks, the limited sensor nodes have to coordinate and split the task among themselves, taking into account the individual characteristics of the nodes (e.g., attached sensors, location, residual energy). The readings of the individual sensors then have to be merged in order to obtain a high-level sensing result.

WSN Characteristics WSN middleware should support the implementation and basic operation of a sensor network as outlined above. However, this is a non-trivial task, as WSNs have some unique characteristics: First, sensor nodes are small-scale devices with volumes approaching a cubic millimeter in the near future [8]. Such small devices are very limited in the amount of energy they can store or harvest from the environment. Furthermore, nodes are subject to failures due to depleted batteries or, more generally, due to environmental influences. Limited size and energy also typically means restricted resources (CPU performance, memory, wireless communication bandwidth and range) [1].

Node mobility, node failures, and environmental obstructions cause a high degree of dynamics in WSN. This includes frequent network topology changes and network partitions. Despite partitions, however, mobile nodes can transport information across partitions by physically moving between them [5]. However, the resulting paths of information flow might have unbounded delays and are potentially unidirectional.

Communication failures are also a typical problem of WSN [4]. Another issue is heterogeneity. WSN may consist of a large number of rather different nodes in terms of sensors, computing power, and memory. The large number raises scalability issues on the one hand, but provides a high level of redundancy on the other hand. Also, nodes have to operate unattended, since it is impossible to service a large number of nodes in remote, possibly inaccessible locations.

Design Principles In order to deal with the characteristics outlined above, some software design principles for WSN have already been proposed [3]. *Localized algorithms* are distributed algorithms that achieve a global goal by communicating with nodes in some neighborhood only. Such algorithms scale well with increasing network size and are robust to network partitions and node failures. *Adaptive fidelity algorithms* allow to trade the quality of the result against resource usage and are thus a key element for resource efficiency. As an extreme case, the application can choose from a whole range of different algorithms which solve the same problem with different quality and resource requirements. *Data-centric communication* introduces a new style of node addressing by focusing on the data produced by nodes, since applications are unlikely to request the current sensor reading such as temperature at a specific node, but instead ask for locations where temperature exceeds a certain value. This allows for more robust-

*This work is partially supported by the Swiss National Science Foundation (NCCR-MICS, grant number 5005-67322).

ness by decoupling data from the sensor that produced it. Finally, *application knowledge in nodes* can significantly improve the resource and energy efficiency, for example by application-specific data caching and aggregation in intermediate nodes.

III. Middleware Challenges

Middleware sits between the operating system and the application. On traditional desktop computers and portable computing devices, operating systems are well established, both in terms of functionality and systems. For sensor nodes, however, the identification and implementation of appropriate operating system primitives is still a research issue [6]. In many current projects, applications are executing on the bare hardware without a separate operating system component. Hence, at this early stage of WSN technology it is not clear on which basis future middleware for WSN can typically be built.

Scope and Functionality The main purpose of middleware for sensor networks is to support the development, maintenance, deployment, and execution of sensing-based applications. This includes mechanisms for formulating complex high-level sensing tasks, communicating this task to the WSN, coordination of sensor nodes to split the task and distribute it to the individual sensor nodes, data fusion for merging the sensor readings of the individual sensor nodes into a high-level result, and reporting the result back to the task issuer. Moreover, appropriate abstractions and mechanisms for dealing with the heterogeneity of sensor nodes should be provided. All mechanisms provided by a middleware system should respect the design principles sketched above and the special characteristics of WSN, which mostly boils down to energy efficiency, robustness, and scalability.

The scope of middleware for WSN is not restricted to the sensor network alone, but also covers devices and networks connected to the WSN. Classical mechanisms and infrastructures are typically not well suited for interaction with WSN. One reason for this are the limited resources of a WSN, which may make it necessary to execute resource intensive functions or store large amounts of data in external components. This may result in a close interaction of processes executing in the WSN and a traditional network. One example of such "external" functionality are so-called virtual counterparts, components residing in the Internet which augment real-world objects with information-processing capabilities [9]. Thus, middleware for sensor networks should provide a holistic view on both WSN and traditional networks, which is a challenge for architectural design and implementation.

Another unique property of middleware for WSN is imposed by the design principle *application knowledge in nodes*. Traditional middleware is designed to accommodate a wide variety of applications without necessarily needing application knowledge. Middleware for WSN, however, has to provide mechanisms for injecting application knowledge into the infrastructure and the WSN.

Data-centric communication mandates a communication paradigm which more closely resembles content-based messaging systems than traditional RPC-style communication. Moreover, event-based communication matches the characteristics of WSN much better than traditional request-reply schemes. In general, communication and application specific data processing is much more integrated in WSN middleware than in traditional systems.

The design principle *adaptive fidelity algorithms* requires the infrastructure to provide appropriate mechanisms for selecting parameters or whole algorithms which solve a certain problem with the best quality under given resource constraints.

In traditional systems, each computing device belongs to someone who is responsible for configuration, maintenance, and error handling. In contrast, WSN nodes must operate unattended, which means that middleware for WSN has to provide new levels of support for automatic configuration and error handling.

Since WSN process real-world data, the concepts of physical time and location play a much more important role than in traditional computing systems. Time and location of sensed real-world events are key elements for fusing individual sensor readings in order to obtain a high-level sensing result. Some application areas might even pose real-time requirements on WSN. Therefore, support for time and location management should by tightly integrated into a middleware infrastructure for sensor networks.

Research Activities There are already some projects underway that aim to develop middleware for WSN, such as [10, 11, 13, 14, 15, 16, 17, 19]. Cougar [11], for example, adopts a database approach where sensor readings are treated like "virtual" relational database tables. An SQL-like query language is used to issue tasks to the WSN. The Smart Messages Project [17] is based on agent-like messages containing code and data, which migrate through the sensor network. NEST [10] provides so-called microcells as a basic abstraction. They are similar to operating system tasks with support for migration, replication, and grouping. SCADDS [14] is based on a paradigm called Directed Diffusion, which supports robust and energy-efficient delivery and in-network aggregation of sensor events.

However, most of the projects are in an early stage focusing on developing algorithms and components for WSN [12], which might later serve as a foundation for middleware. Moreover, most of the current results are based on simulations or small-scale experiments in laboratory settings. The suitability for large scale networks still has to be proven. First concrete experiments [4] show that even very simple protocols and algorithms can exhibit surprising complexity at scale. After all, there is still a long way to go for successful WSN middleware, both in terms of design concepts and system implementations.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks*, 38(4):393–422, March 2002.

[2] M. Beigl, H.W. Gellersen, and A. Schmidt. MediaCups: Experience with Design and Use of Computer-Augmented Everyday Objects. *Computer Networks, Special Issue on Pervasive Computing*, 25(4):401–409, March 2001.

[3] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *MobiCom 99*, Seattle, USA, Aug. 99.

[4] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks. Technical Report CSD-TR 02-0013, UCLA, February 2002.

[5] N. Glance, D. Snowdon, and J.-L. Meunier. Pollen: using people as a communication medium. *Computer Networks*, 35(4):429–442, 2001.

[6] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Networked Sensors. In *ASPLOS 2000*, Cambridge, USA, Nov. 2000.

[7] A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. In *DARS 02*, Fukuoka, Japan, June 2002.

[8] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Mobile Networking for Smart Dust. In *MobiCom 99*, Seattle, USA, August 1999.

[9] K. Römer, F. Mattern, T. Dübendorfer, and J. Senn. Infrastructure for Virtual Counterparts of Real World Objects, April 2002. www.inf.ethz.ch/vs/publ/papers/ivc.pdf.

[10] A Network Virtual Machine for Real-Time Coordination Services. www.cs.virginia.edu/nest.

[11] Cougar Project. www.cs.cornell.edu/database/cougar.

[12] Mobile Information and Communication Systems. www.nccr-mics.ch.

[13] Power Aware Distributed Systems. pads.east.isi.edu.

[14] Scalable Coordination Architectures for Deeply Distributed Systems. www.isi.edu/div7/scadds.

[15] Sensorwebs Project. basics.eecs.berkeley.edu/sensorwebs.

[16] Smart-Its Project. www.smart-its.org.

[17] Smart Messages Project. www.rutgers.edu/sm.

[18] The 29 Palms Experiment: Tracking Vehicles with a UAV-Delivered Sensor Network. www.eecs.berkeley.edu/∼pister/29Palms0103.

[19] Webdust Project. www.cs.rutgers.edu/dataman/webdust.