Multifunk: Self-Organizing Sensor Networks for Industrial Process Monitoring

Gokul Balakrishnan, Michael Geisinger, Christian Buckl
Fortiss GmbH – An-Institut der TU München
Guerickestr. 25, 80805 München, Germany
balakris@in.tum.de, {geisinger, buckl}@fortiss.org

Abstract

The technology and processes associated with manufacturing have undergone a major change during the past decades. However, what has stayed the same is that availability of the production system and quality of the produced goods play a key role. Both aspects can only be met by adequate monitoring. Monitoring itself should be easily configurable and allow addition, removal and replacement of sensors without interruption of production. Sensors must at least partially be mobile, namely wireless. The research project Multifunk aims at development and prototypical implementation of software and hardware components for monitoring of industrial production processes using model-driven development, a service oriented architecture, network self-organization and wireless components. This paper presents current results achieved in design of the system architecture, implementation of an adequate transport protocol, compliance to standards as well as a prototypical use case.

1. Introduction

Automated monitoring in industrial environments improves the efficiency of the production process significantly, because it allows detection of problems at an early stage, hence increasing productivity. It is implemented by deploying heterogeneous sensors into the production area and checking the acquired data against predefined values to detect discrepancies. Besides observation of the process, monitoring data may also be stored and used as quality assessment for the produced goods.

Nowadays, most monitoring systems are static: Sensors are not mobile, which is partially due to the fact that they are quite large and have relatively high power consumption. Distributed processing of the sensor data is often not implemented and detection of discrepancies is usually "hard-coded" and not specified in a platform independent way.

In the past, this approach has been motivated by the fact that specialized systems are cost-effective and reconfiguration is not needed. However, technologies available today allow to build very cost-effective monitoring systems while retaining flexibility and reconfiguration of the system.

In the Multifunk[1] project, research is done to improve application of state-of-the-art monitoring systems in industrial processing. The project aims at developing new hardware and software technologies for self-organizing, wired and wireless sensor networks. The system architecture will be prototypically applied to industrial production scenarios involving processes with high temperatures and pressures as well as radiation. The goal is to maximize reliability and minimize human interaction as well as energy consumption. This is realized by application of model-driven design, self-organization and a service-oriented architecture (SOA).

Model-driven design is used to ease conception, programming and configuration of monitoring systems. The system model, which is specified with *domain specific languages* at a high level of abstraction, captures all relevant information for the system. *Model transformation* and *automatic code generation* [8] is used to create the concrete system from the model. This approach allows non-experts to develop and adapt the system.

An important aspect in (wireless) sensor networks is **self-organization**, which can be used to strengthen reliability and robustness of the system regarding adaptation to varying conditions. For example, a self-organizing sensor network may have the ability to reconfigure itself automatically in response to changes in the number of devices (e.g., due to failed nodes or addition of new sensors). Typical methods employed are automatic device detection, adaptation of routing and optimization of network traffic.

Application of a **service oriented architecture** aids in a modular and scalable system design. Services can be deployed to individual nodes in the network and bind to other services. The concrete placement of a service can be transparent to the user and the other nodes in the system.

In this paper, we will focus on the system architecture and communication aspects of the Multifunk project and

[1] http://www.multi-funk.de/

present first results. The development process allows a seamless integration of new nodes and follows a programming paradigm that aims at a *single system illusion*.

Section 2 provides an overview of related work followed by an outline of our current work in Section 3. Section 4 introduces the system architecture, components and related tools. Section 5 shows the application of the already implemented parts of the system in a simplified use case and Section 6 summarizes the results achieved so far.

2. Related Work

Akyildiz et al. [1], Townsend et al. [9] and Miao [4] present a good introduction to sensor networks and applications. Distributed microsensor networks are used both in civil and military applications. Microsensors also provide a more flexible approach to industrial applications and enable better quality control. Riedel [7] describes self-organizing sensor networks that may operate autonomously over long periods of time in remote environments which also requires a very low power design as covered in Park et al. [6]. Examples for wireless system architectures, network topologies and real-time environment testing are discussed by Hill [2].

Jeong et al. [3] analyze the suitability of sensor networks for industrial processes with the conclusion that a sensor network cannot meet all requirements of each application because of different characteristics of heterogeneous components.

3. Present Work

The Multifunk project is currently only at an early development phase. The present work involves implementation of the system architecture with a focus on communication aspects. The base of the system is a generic protocol for the sensor network that is optimized for energy saving on wireless nodes. The self-organizing architecture is built upon this base. Furthermore, a gateway component is currently being developed to make the system accessible according to the *Device Profile for Web Services* (DPWS) [5].

4. System Architecture

The software architecture is divided horizontally into four layers (compare Figure 1):

1. The *machine layer* consists of wired and wireless sensor nodes for physical data acquisition.
2. The *network layer* is responsible for data transport between sensor nodes. A common transport protocol abstracts from the heterogeneous communication protocols (e.g., wireless communication with or without encryption, wired field buses).
3. The *management layer* provides mechanisms for self-organization and centralized data processing. However, data may already be preprocessed by specific services at the machine or network layers (e.g., for reduction of data volume).
4. Finally, the *factory layer* takes care of data storage and presentation. For quality assessment, data are associated to produced goods by RFID tags or barcodes. Model-driven system design and modification of the running system is also performed at this layer.

4.1. Sensor Nodes

In the Multifunk project, we consider wired and wireless devices as sensor nodes. Typical data acquired in context of the project are temperature, pressure, camera images and identification tags. Each node offers a set of services that abstract from the physical world and the concrete implementation. For services to be placed on a sensor node, the node requires a certain computation ability. Since services build upon a middleware layer with a common interface, their source code is portable across different platforms.

We consider three classes of sensor nodes according to Table 1. The "OS" column names typical operating systems. If an operating system is present, the common middleware layer is realized on top of it. The "DLS" column specifies whether dynamic loading is supported. Dynamic loading is used to add new services during runtime. Nodes without dynamic loading support have to be completely reprogrammed when the firmware has to be adapted. A suitable bootloader must be present in any case.

Class	Architecture	Memory	OS	DLS
A	x86	large	Linux	yes
B	ARM, AVR	small	Contiki	yes
C	AVR	very small	*(none)*	no

Table 1. Sensor Node Classes

4.2. Data Transmission in the Network

In contrast to the wired sensors, wireless sensors need to be power aware. Network activity and thus power consumption is minimized by local preprocessing and efficient data compression. This is realized by only transmitting service parameters instead of network packets containing a lot of overhead. The common *service model* (see Section 4.3) ensures that the signature of a service call is known in advance.

Data transmission is based on a binary protocol at the *presentation layer* of the ISO/OSI model, which allows it to be used on Ethernet, WLAN, IEEE 802.15.4 or serial

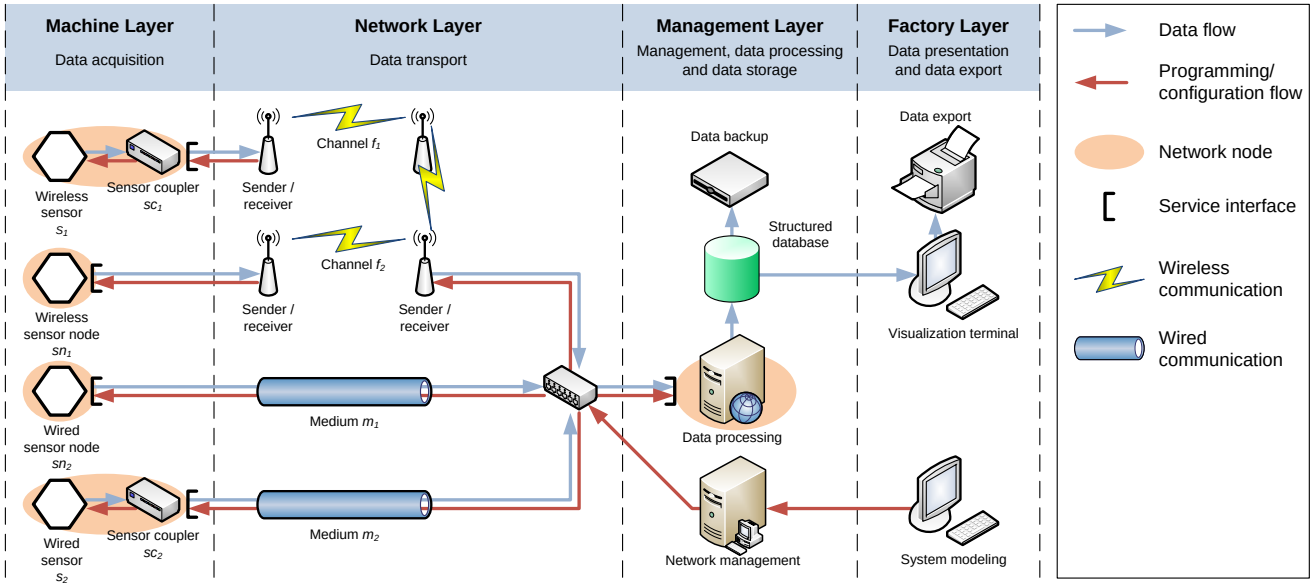

Figure 1. Multifunk System Architecture

connections. We call that protocol *Multifunk protocol*. In addition, protocol components for efficient data transport at lower levels (e.g., a network layer component for IEEE 802.15.4) will be implemented as required.

The structure of a packet for a specific service is defined in a shared C language header file that is currently generated manually, but will be generated using the model-driven development process in the future as follows:

The required properties of the service (name, input and output parameters with their data types and range of values) are specified in the service model (compare Section 4.3). A model-to-model transformation optimizes representation of the parameters so that the number of bytes for a message is reduced. The resulting mapping is used to generate the header file for service definition and the marshaling and de-marshaling routines for each target platform. Maximum compression is not enforced in all cases, since computations needed to "unpack" the data are expected to waste even more energy.

Each service is assigned to a service routine, a function that is called when the service is executed. The function receives the de-marshaled input parameters and returns the output parameters by value or by reference. Since the service specification must be platform independent, templates are used to specify the service functions that are based on a middleware that provides a *hardware abstraction layer* (HAL).

On a node, services are exposed by a *service broker* that directs service requests to local or remote services and calls the respective service functions. De-marshaling of the query and marshaling of the reply is handled at this level.

4.3. Model-Driven Development

An aspect of the project is to use a model-driven approach to develop the system. For this purpose, a special *system modeling* entity is present at the *factory layer*. The system model consists of five parts: The *hardware model* specifies the supported hardware platforms and compatible operating system and middleware layer implementations. The *service model* lists all available service templates that can be instantiated. The *network model* denotes the structure of the network including all physical network links and their properties. The *application model* contains all service instances and their logical dependencies and links against other services without consideration of the physical representation on concrete hardware. Finally, the *error model* specifies rules the behavior of the system is checked against as well as mitigation mechanisms, for example transition to a safe system state. The sub-models are augmented with additional information (e.g., placement of service instances) in a model transformation phase, which is followed by automatic code generation.

5. Use Case

Since testing a prototypical implementation in a real plant is not feasible due to potential interruption of production, we have built a demonstration setup for testing purposes. The demonstration system can also be used to train workers in using the system without potential harm to the production facility. The demonstration platform is based on

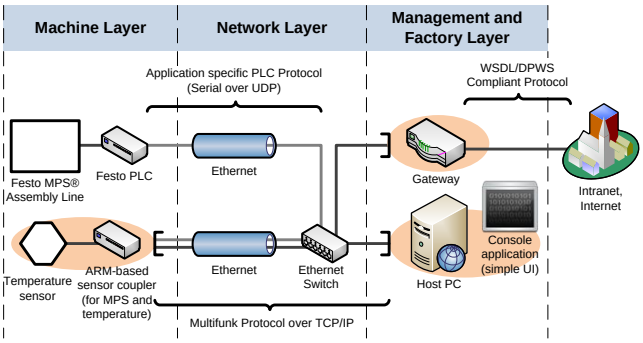

Figure 2. Network Structure in the Use Case

a small Festo MPS® assembly line[2] that allows simulation of common industrial automation scenarios.

In our scenario, two MPS stations and conveyor belts have been connected to form a circular assembly line. The stations allow execution of storage and processing tasks. Each station uses a PLC to achieve its tasks.

Figure 2 shows the structure of the network: the PLCs are connected over Ethernet to an ARM-based microcontroller of node class B according to Table 1 and emulates a *sensor coupler* (compare Figure 1): it exposes a service interface over IP using the uIP network stack and translates queries from other network components into commands that are sent to the PLCs. If the call yields a result, it is processed and sent back as a service reply. This allows us to monitor the state of the PLC's I/O in a service-oriented manner. Furthermore, we can trigger execution of commands on the PLC, for example to bring the system to a safe state when an error is detected. The microcontroller also offers a service for reading the ambient temperature from a digital temperature sensor (e.g., to detect overheating of a component in the production line).

The current setup does not contain any wireless components. Integration of such components will be a subsequent step. Due to the Multifunk protocol, integration will follow the same structure as for wired components.

The initial implementation of the system features a C++ application on a normal PC (node class A) that binds to the services on the microcontroller and makes them available to the end user. The user can monitor the current system state and may issue queries or commands. The system will soon be extended by a *gateway* component that translates between the Multifunk protocol and a *web services* oriented approach according to DPWS [5]. This will allow access to the system from external networks (e.g., the Internet) in a compliant way. Initially, the gateway will be implemented on the same PC than the host application. Later on, a reconfigured commercially available router device may be used.

[2] http://www.festo-didactic.com/int-en/learning-systems/

6. Conclusion

The presented work will improve structure and maintenance of wired and wireless sensor networks in production environments through model-driven design and self-organization. Communication is designed in a platform and medium independent way. Tool support for model-driven development and monitoring of sensor networks will be provided. Current results are the implementation of an optimized transport protocol for service invocation and prototypical implementation in an automation scenario.

Future work focuses on the generation of the prototypical system from a model. In addition, wireless network components will be added to form a heterogeneous sensor network with wired and wireless components. The system will finally be applied to three different industrial production scenarios where automatic processes should be monitored and documented for later reference. Some of these processes require minimal human interaction because of harsh environmental influences.

Acknowledgments

This work is partially funded by the German Ministry of Education and Research (BMBF) under grant no. 16SV3883.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Comm. Mag.*, Aug. 2002.

[2] J. L. Hill. *System Architecture for Wireless Sensor Networks*. PhD thesis, University of California, Berkeley, 2003.

[3] W. Jeong and S. Y. Nof. Performance evaluation of wireless sensor network for industrial applications. In *Journal on Intelligent Manufacturing*, volume 19, pages 335–345, 2008.

[4] Y. Miao. Application of sensor networks. Seminar on wireless self-organization networks, Friedrich-Alexander-Universität Erlangen-Nürnberg, Feb. 2005.

[5] G. Moritz, E. Zeeb, S. Prüter, F. Golatowski, D. Timmermann, and R. Stoll. Devices profile for web services in wireless sensor networks: Adaptations and enhancements. In *IEEE Conf. Emerg. Tech. and Factory Autom. (ETFA)*, pages 1–8, 2009.

[6] S. Park et al. Optimized self organized sensor networks. In *Sensors*, 2007. ISSN 1424-8220.

[7] T. Riedel. Self-organizing, wireless sensor network. *Remote Site & Equipm. Managem. Mag.*, Apr. 2004. Millennial Net.

[8] A. Singh, J. Schaeffer, and M. Green. A template-based approach to the generation of distributed applications using a network of workstations. *IEEE Transactions on Parallel and Distributed Systems*, 2(1):52–67, Jan. 1991.

[9] C. Townsend and S. Arms. *Sensor Technology Handbook*, volume 1, chapter 22: Wireless Sensor Networks: Principles and Applications, pages 575–588. Newnes, 2004.